UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

School of Electronics and Computer Science

Topologies for Programmable Analogue Circuits

by

David Varghese

Thesis for the degree of Doctor of Philosophy

December 2005

To My Parents...

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

TOPOLOGIES FOR PROGRAMMABLE ANALOGUE CIRCUITS

by David Varghese

Modern VLSI design has always tried to shorten the design cycles for digital and analogue circuits. This effort has been very successful over the past years for digital integrated circuits, through the development of FPGAs (Field Programmable Gate Arrays) and appropriate CAD methodologies to suit the design flow. Although a very low paced development, for the analogue circuits due to complexities in the analogue domain, the role of Field Programmable Analogue Arrays (FPAAs) to enable rapid prototyping and testability of an analogue design solution has been realised by the industry. Hence, different types of programmable analogue circuits, have been developed to suit specific applications. But none of these FPAAs have actually addressed the problems of choosing an appropriate analogue building block, an architecture and the routing strategies between them. Hence, this research focuses on the *"Topologies for Programmable Analogue Circuits"* and also in the development of a target FPAA chip primarily for Instrumentation and Data Acquisition Systems.

This thesis, reviews previously developed FPAAs and also summarises the key strategies for choosing a Configurable Analogue Block (CAB) and the architecture for the target FPAA. This document includes the conceptual design and tests for the chosen CAB called the Differential Difference Amplifier (DDA) integrated with the second generation Current Conveyor (CCII), to form the hybrid DDACCII CAB. The implementation of a cluster based hierarchical architecture, following a similar concept to the HFPGAs (Hierarchical Field Programmable Gate Arrays), for connecting the CABs in the target FPAA, is also included in this thesis. This thesis covers the design and fabrication of an FPAA prototype and the decisions and design trade-offs made. A methodology for interconnectivity analysis and establishing relationship between the routability of the FPAA and the flexibility if its hierarchical interconnection structures, not relying on the quality of partitioning/routing algorithms as in digital circuits, is also discussed in this thesis.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Dr.J.N Ross for his valuable guidance and involvement in this research. I am also thankful to him and Prof Andrew D Brown for granting a full scholarship throughout my PhD and also funding the expenses incurred in this research.

I would like to thank Dr. Mark Zwolinski for his involvement and suggestions in this research. I would also like to thank Dr. Peter Wilson for technical assistance in using the CAD facilities.

I am thankful to my parents for providing me the strength and support throughout my career till now. I would also like to thank my friends Apurva, Arun, Bhaskar, Biswajith, Chincha, Jasmine, Muge, Narasimhan, Sachin, Vinu, Sabu & family, Prashanth & family, Shashindran & family, for making my stay in Southampton memorable.

Chapter 1

Introduction

Field Programmable Devices have proved to be an important development in modern VLSI design, by reducing the design cycles and hence enabling rapid prototyping and testability of a design solution, before the mass volume production of an ASIC chip. A good example is the *Field Programmable Gate Array (FPGA)*, which has drastically reduced the design cycles for digital integrated circuits, when used with the developed modern CAD methodologies for testability, optimization and synthesis from a behavioural description of the digital circuit. Over the past several years, similar developments have been going on, in an effort to reduce the design cycles for analogue circuit design too. The greater complexities to be considered in designing analogue circuits when compared to digital circuits, are the main cause for the slow development of CAD tools that can accomodate the complexity of analogue signals. But the role of analogue circuits in modern VLSI solutions is significant, as they form the interface between digital electronics and the real world analogue signals. Hence analogue circuits have been playing an important role in applications ranging from signal processing and conditioning to control systems and biomedical instrumentation.

Gradually the demand for analogue systems has been increasing in the market for low-power and high-speed applications. Hence faster design and test methodologies for analogue circuit design had to be researched and developed. As a result, research on the possibility of developing high performance analogue circuits that could suit the CAD methodologies bridging the gap between behavioural description and synthesis of the analogue circuit, and which can be mapped on to a target technology has been going on for several years.

The *Field Programmable Analogue Array (FPAA)*, was one of the significant developments in analogue and mixed-signal circuit design that could provide low-cost rapid prototyping techniques. But again there has not been any universal solution or any array of configurable analogue blocks (CABs), that could satisfy all the analogue functions required in electronic systems. The developed FPAAs have been specific to a certain range of applications, dependent on the nature of the CABs and the routing architecture between them. They have not proved to be flexible with respect to switching between configurations and also for reducing the number of external parasitic components (resistors, capacitors) at the same time.

This research reconsiders possible *topologies for programmable analogue circuits*, evaluates them on the basis of applications covered and the ability to handle with good precision the complexity of analogue signals with respect to frequency, signal levels, time and parasitics. This project aims at the development of a CAB with an appropriate routing network for an array of CABs, *primarily targeting instrumentation and signal conditioning circuits.* The possibility of covering *more target applications will also be researched.* The project aims at focusing on the flexibility achieved in switching between possible configurations with *minimum number of external passive components.* None of the developed FPAA architectures (except the FPAA using switched-capacitor technology) focuses on this aspect of reducing the number of external components and hence the off-chip signal wires. Thus the developed FPAA boards tend to be more complex due to the large number of signal wires forced to come out off the chip. Hence the routing architecture between the CABs is also significant in reducing the complexities of the signal routing between CABs and also the number of signal wires carrying analogue signals to the CABs and vice-versa. The project considers the routing architecture of the FPAA and also a possible method of discussing these routing strategies, because there has not been much research work done on these issues for programmable analogue circuits.

This thesis describes the initial investigation done on FPAAs, the possible architectures proposed by various research teams and their target applications. A brief discussion of all the proposed topologies for FPAAs till now, describing their architectures and targeted applications, is covered in Chapter 2. Chapter 3 includes a discussion of the key expected features and design strategies for a CAB. The design of the CAB as a result of the research work along with appropriate simulation results is described in Chapter 4. These results verify the intended target functionalities and also the flexibility in achieving the same circuit configuration from one to another, with minimal number of external components compared to the traditional circuits and existing FPAAs. It also covers the efforts in improving the versatility of the designed CAB by additional stages of circuitry that makes it suitable for a wider range of analogue applications. The proposed design of the FPAA was fabricated in CMOS $0.6\mu m$ technology. The layout and other design issues of the whole FPAA chip are discussed in Chapter 5. Chapter 6 includes a discussion on a few tested applications, the corresponding measured results and a comparison with the simulated results. Chapter 7 discusses the need for interconnectivity analysis of the proposed hierarchical interconnection architecture for the HFPAA Hierarchical Field Programmable Analogue Array. A description of a novel methodology adopted for such an interconnectivity analysis using a set of identified analogue benchmark circuits is also included in this chapter. Chapter 8 covers a summary of the progress in this project, conclusions and briefly the future work.

Chapter 2

Literature Review

An initial investigation on the existing FPAA topologies developed by various researchers and the industry was done. The following sections include a brief discussion and review of these topologies.

2.1 Switched Capacitor based FPAA

2.1.1 Architecture

This FPAA architecture in [10], is a two-dimensional array based architecture of 4x5 programmable analogue cells, as shown in Figure 2.1. The connectivity within the ar-

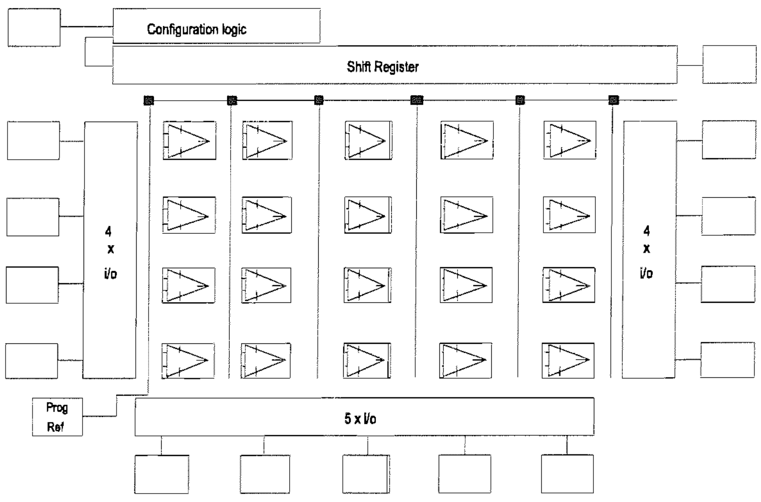


FIGURE 2.1: The FPAA Architecture MPAAx20, [10]

ray is controlled by configuration logic, placed at the upper periphery of the chip. The function of each cell is controlled by RAM local to the cell. A band-gap voltage reference is provided on the chip, which has an 8-bit programmable output voltage for each analogue cell. The periphery of the chip also has buffer cells arranged, which may be configured into unity gain buffers or smoothing/anti-aliasing filters with external components. The array consists of 41 operational amplifiers (opamps), 100 programmable capacitors and 6809 electronic switches. The control over the circuit connectivity and capacitor values is through the electronic switches. The chip is configured through a shift register that runs along the full-width of the array (619 bit wide), loaded with the required bits through the serial port or an Erasable Programmable Read Only Memory (EPROM). The data is loaded from this shift register in a parallel fashion into the row selected from the array, followed by the loading of the configuration shift register with next set of bits for the next row. There exists an interface multiplexer (MUX), which controls the inter-cell connections within the array. This process of loading operation continues until the entire array is fully configured. The FPAA architecture in [10], does not specify any relation between the architecture and the technology or any analysis of the maximum interconnectivity achieved for maximum routability of circuits mapped on this FPAA.

2.1.2 Technology

This FPAA uses switched capacitor technology, achieved through the opamps, the programmable capacitors and appropriate control over the circuit connectivity through internally programmable switches. The clock distribution and phasing of the clocks for the charging/discharging of the internal capacitances and the achieved bandwidth/frequency range of operation are not specified in [10].

2.1.3 Software Support

In order to make the whole design procedure from schematics to place and routing more user friendly and flexible, a CAD support was also developed for this FPAA architecture in [10]. The CAD support for the FPAA was known as the *MIDT (Motorola Interactive Design Tool)*, designed to run on PC machines running Windows OS. The MIDT displays a simplified view of the chip (similar to the FPGA editor from Xilinx tools) and allows the user to select and place "macros" (pre-packaged circuits such as gain stages, filters, full-wave rectifiers etc) from the Motorola library.

2.1.4 Summary

The switched capacitor technology for an FPAA in [10], has proved to be a better solution when compared to other technologies such as the Transconductors in [61] and Switched current methods in [94]. This is accounted for by the following:

- The use of parasitic insensitive architectures allows the arbitrary routing of signals in an array with minimal loss of signal integrity, thus preventing the parasitic capacitances from each node to ground from affecting the targeted transfer function.
- The matching of capacitors in close proximity is very good, thus enabling precise analogue functions to be generated without the expensive and cumbersome techniques of calibration.
- As long as the settling time requirements are met, the parasitic resistances in the switches and the signal path do not affect the desired transfer function.

The precision obtainable in this system has been proved suitable for use in feedback systems [10] that requires accurate control of the phase and magnitude characteristics. Results from a phase compensator built in the FPAA, with a measured 3dB frequency point of 29.5 Hz for a target of 30Hz is included in [10]. Also the other possible limitations suspected in this FPAA are:

- the large area required for the programmable capacitor arrays
- the requirement of continuous-time filters for anti-aliasing and reconstruction
- the requirement of the switched capacitor filter cutoff frequency to be at least one order of magnitude lower than the sampling frequency, to minimise continuous-time filtering

This technology was later migrated to another company Anadigm, with Adrian Bratt and Ian Macbeth as the originators of this FPAA [16] design. Anadigm is still successful in providing commercial FPAA based solutions [3].

2.2 Current-Mode based FPAA

2.2.1 Architecture

This FPAA, in [75], uses a two-dimensional array based architecture, which consists of a modular array of cells. Each cell is composed of three sub-cells, (a)Configurable Analogue Cell (CAC), (b)Programming Register (PR) and (c)Programming Logic (PL), as shown in Figure 2.2. This architecture targets applications in the area of analogue signal processing such as filtering and data conversion, through the configuration of the CACs to implement the basic building blocks such as integrators, amplifiers, attenuators etc. The configuration and the connection between the CACs is performed using the circuit topology developed for the CAC, by, (a)a Function Word (FW) , through which the function of the CAC is digitally programmed, (b)a Parameter Word (PW), through which the characteristics of the configured function on the CAC is digitally programmed, and (c)a Routing Word (RW), through which the desired connectivity of the CAC is configured. The corresponding digital control words for each CAC are stored in the

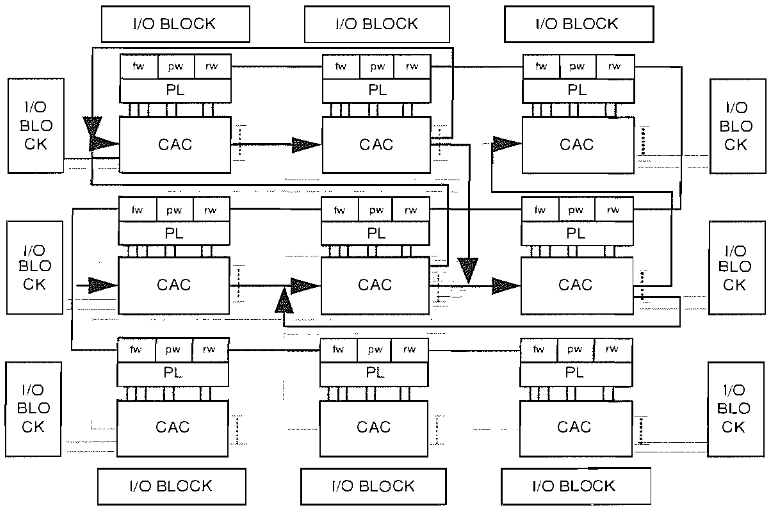


FIGURE 2.2: The Current-Mode based FPAA Architecture, [75]

respective PR, and all these PR's are connected in series to form a single shift register, which can be loaded with the desired programming bit stream through one external pin. The I/O blocks that provide the input and output interface for the chip, are arranged in the periphery of the chip, surrounding all four sides of the array. The architecture of the CAC is designed using the simple current-mode approach, making it suitable for low voltage and high frequencies, to perform, (a)an Integrator, (b)an Amplifier or (c)an Attenuator. The architecture of the CAC is based on the active cascode integrator, that provides high DC gain, operates at high frequencies and low voltages and is also programmable. The proposed architecture of the CAC consists of n identical slices of structurally similar sized n-channel and p-channel transistors. These slices are connected in parallel and can can be configured by activating certain branches. All branches are by default, deactivated, where all the transistors are turned off with no floating gates.

2.2.2 Technology

The FPAA in [75], is based on simple current-mode CMOS circuits targeting signal processing applications, which allows for operation at high frequencies and low voltage, and can be implemented in a digital CMOS process. The architecture tries to avoid excessive number of programming devices (switches) in the signal path, thus avoiding significant performance degradation. The merging of this FPAA architecture with a FPGA to provide Field Programmable Mixed-Signal Array (FPMA), is stated in [75].

2.2.3 Summary

A biquad implementation having three integrators and two attenuators was used to demonstrate the application of this FPAA in [75]. The operating range of frequencies for this FPAA has been claimed in [75], from several tens of kHz to a few MHz. The implementation of programmable filters is the main demonstrated application of the FPAA in [75]. The amount of interconnectivity available between the analogue cells is not discussed with respect to maximum routability of circuits. It is only stated that the output of each CAC is connected to its closest neighbouring CACs. The claim in [75], is mainly on a novel approach of programmability, which is based on hardwiring most of the signal path and minimising the number of programming switches used in the path. Also the whole FPAA is based on two simple sub-circuits, namely the integrator and the current mirror.

2.3 FPAA based on Sub-Threshold techniques

2.3.1 Architecture

The analogue array architecture in [60], is based on *sub-threshold circuit techniques* and consists of a collection of Configurable Analogue Blocks (CABs) and an interconnection network. Interconnection between the CABs and the analogue functions to be implemented in each block are defined by a set of configuration bits loaded serially into an on-board shift register by the user. This architecture targets the simulation of various neural network applications. In addition to introducing CABs based on sub-threshold techniques, a discussion on the interconnection networks is also included in [60].

The interconnection networks [38, 57, 73] are generally categorised into the following two types:

- Crossbar:- It provides full interconnection capability between any two connected elements, with less delay on data transfer, known as non-blocking. It does exhibit an area growth of O(N^2) for connecting N inputs to N outputs.

- Hierarchical(multistage):- A few examples of this network include the Banyan, Omega and Delta. They have a cost growth of O(N log N) at the expense of longer delay between inputs and outputs. They allow data transfer or routing between two layers or elements, but not within the same layer.

Hierarchical networks are preferred for the design of the array because of the area requirement. But since the area is always of concern from a VLSI point of view, it is required that the network possess the property of *"area universality", which is defined as the ability of the network for a given area to efficiently embed any circuit whose size is only slightly smaller.* A *fat tree*[62] is one such *area universal* network, that has a hierarchical network, which has optimum number of elements, data transfer delay and growth rate of routing channel size. Based on this network, the FPAA is designed with CABs (as the leaves of the fat tree) connected by switch blocks (SBs) different levels of the fat tree. The switch blocks can be realised by using crossbar switches. The connections made by the SB is determined by the contents of a 10-bit shift register specified by the user. The SB also allows for polarity changes of the differential signals. Therefore addition or subtraction among current-mode signals is possible. The significance of such an area universal hierarchical interconnection network would be applicable for a similar efficient interconnectivity between CABs in other FPAA technologies too.

2.3.2 Technology

For the implementation of high-order neural network applications, the basic functions required are *addition,threshold operation*, *coefficient multiplication* and *signal multiplication.* For those networks that are capable of learning, the values of the coefficient multiplier are very important for having the feature of adjust-ability. Thus these functions are realised in CMOS technology through sub-threshold circuit techniques, that offer the advantage of low power dissipation. The circuits utilise both differential current-mode and differential voltage mode signals. The swing of the voltage-mode signals is kept small to maximise the speed of operation. The operation of addition among signals is obtained by representing the signals in current-mode and then adding currents at the corresponding nodes. The threshold operation being similar in behaviour to that of an analogue comparator, has two states (low and high) and a gain region required for the transition between the two states. The design of the CAB of circuit primitives and a few transistors that act as analogue switches. The circuit primitives are, a pair of p-channel transistors, a pair of n-channel transistors, a p-channel long tail pair, a n-channel long tail pair and an analogue multiplier. The switches are configured by a local 3-bit shift register. Each shift register controls various switches and/or multiplexers within the CAB itself. Therefore the circuit topologies for different functions are similar, and these functions can be conveniently achieved by configuring the circuit primitives and the analogue switches. The configuration bits are set by shifting the bits into the chip serially. A control unit determines the required bit patterns for the configuration cycles and the clock phases of the shift registers to control the beginning and the end of the different cycles.

2.3.3 Summary

Demonstration of the FPAA is illustrated in [60], where one CAB was configured as an analogue multiplier (biquad Gilbert multiplier) and the other as a current buffer. The other experimental results illustrated in [60], include the configuration of the CAB as a current comparator. The application of FPAAs in neural networks with the neural network simulations of the macro models of the circuits is also illustrated in [60] . The macro models were developed by first simulating the DC and AC characteristics of the analogue operations and then modelling the operations with sets of resistors, capacitors and dependent sources according to the simulation results derived from the extracted layout. The prototype and the simulation results proved the suitability of the FPAA for neural network applications.

Sub-threshold techniques provides low-power dissipation and also reduces the effects due to the resistance of the switches on analogue operations. Most of the single chip analogue neural network implementations, are designed for a fixed network topology. The main advantage focused on here, is to devise an IC strategy, whereby various analogue networks can be realised, as determined by the user, in some type of reusable generic prototyping medium. Apart from the sub-threshold techniques used, differential circuitry is used for noise immunity, and current-mode signalling provides addition and subtraction operations. Memory functions such as integrators, coefficient storage and sampled analogue delay lines are also supported. Since the circuits in every programmable block are identical and also since the input and output of each block are accessible through the routing network, each programmable analogue function can be tested exhaustively, unlike analogue circuits.

2.4 Current-Conveyor based FPAA

2.4.1 Architecture

The architecture current-conveyor based FPAA in [36], contained four current-conveyor CABs and a transmission gate-based interconnection network. This architecture was named as a bipartite architecture, where the CABs are organised for connection between current-conveyors of different groups as shown in Figure 2.3. The typical two-dimensional array architecture, which includes two current-conveyors arranged in a feedback loop is stated as a disadvantage in [36], although it facilitates applications like gyrators for a second order filtering function. This is because a significant waste of silicon area occurs, as it is quite often that only one of the two current-conveyors is used when the other one must be turned off. The current-conveyor based implementation of a CAB here is capable of implementing, first-order filtering functions, amplification, log/anti-log, integration/differentiation and addition/subtraction/multiplication. The architecture of the CAB in [36, 72] includes the current-conveyor, two transconductors (programmable resistors), two programmable capacitors and a buffer. The current-conveyor used is based on a low input resistance class AB CMOS Design [71], with a low impedance on the node X so as to implement a fast comparator. The variable resistors are implemented using CMOS double pair of transconductors [72], were this transconductor is configurable as a programmable resistor. The variable capacitors are implemented using programmable capacitor arrays, that produce discrete capacitor values. The unity gain buffer is based on an operational amplifier in a unity gain feedback configuration. The transconductor chosen for the design is based on the CMOS double pair, due to its ability to be used as a grounded resistor and also avoid the mismatch problems between NMOS and PMOS transistors.

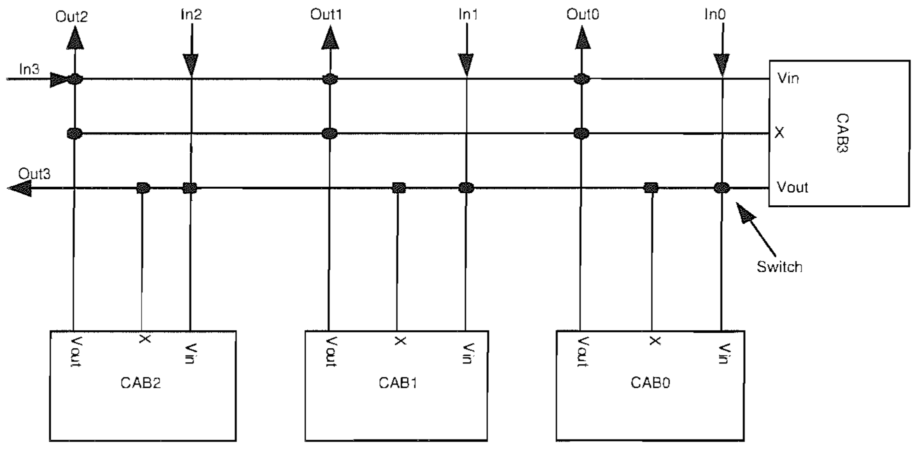


FIGURE 2.3: Test Chip Block Diagram, [36]

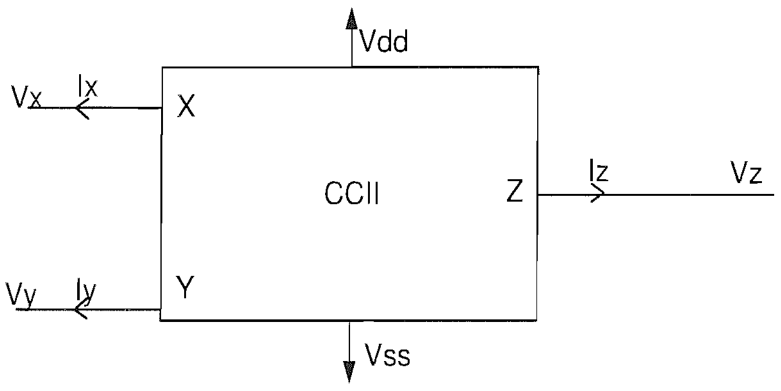


FIGURE 2.4: Current-Conveyor symbol, [36]

2.4.2 Technology

A second generation current-conveyor [94, 82, 103, 83] is shown in Figure 2.4. It is a three terminal device. The node Y is at high (infinite) impedance, with no current flowing into it. The node X is a very low impedance input. When a voltage is applied at node Y, the voltage gets replicated at node X. When a current is injected into node X, this current gets replicated at node Z. A positive Z output current-conveyor is denoted as *CCII+*.

2.4.3 Summary

This CMOS FPAA in [36], targets application frequencies greater than 1MHz, especially for video applications. The second-generation current-conveyor (CCII) is used for the design of this FPAA. This 0.8μm CMOS chip contains four CABs based on CCII and an interconnection network based on transmission gates. The current-conveyor based building blocks do not use feedback for simple functions. The gains in the current and voltage from one node to another is unity and thus stability is ensured and requires no compensation, unlike the opamp which requires a compensation through a dedicated capacitor that reduces the unity-gain feedback of the opamp to a point phase margin to ensure stable operation, which again makes the opamp incapable of producing significant gains at high frequencies. Also current-conveyor based amplifiers have a constant bandwidth, independent of gain, in contrast to the opamp based amplifiers having a constant gain-bandwidth product.

2.5 Transconductor based FPAA

2.5.1 Architecture

The architecture of an FPAA based on the transconductance technique is shown in the Figure 2.5. It consists of two cross-bar interconnection networks, multi-valued memory refresh circuits, an array of programmable differential voltage or current sources and an array of configurable analogue blocks. Each line represents a differential wire and the solid circles represent the MOSFET transconductors. The two interconnection networks are the VRIN (variable resistor interconnection network) and the SCRN (signal controlled resistor network). The gate voltages of the transconductance in the VRIN are controlled by the multi-valued memories [41, 59]. The gate voltages in the SCRN are controlled by the output voltages of the CABs or external input signals. Thus the SCRN can be used for the realisation of four-quadrant multipliers and dividers [52] , and time-varying circuits. The CABs comprises a fully differential opamp with switchable feedback capacitors. The MOSFET transconductors play the role of the resistors in a conventional opamp circuit. The combination of the MOSFET transconductors and different types of active elements, can be used for different analogue signal processing problems, like a continuous time filter in [63]. The values of the transconductors may be set using programmable sources. These sources can be constructed using the multi-valued memories with voltage or current amplifiers as buffers. The implementation of an opamp, a fully differential continuous time biquad filter circuit, and a voltage controlled oscillator (VCO) embedded into the FPAA are included in [61], to demonstrate the flexibility of the architecture. The CABs are configured as integrators and amplifiers. Different transfer functions can be programmed by controlling the gate voltages of the transconductors through the multi-valued memories. The parasitic capacitance of the transconductors and the connection wires gets cancelled in a first order approximation due to differential signalling.

2.5.2 Technology

In the earlier section, an architecture for the FPAA using sub-threshold current-mode techniques was considered, where the reconfiguration for different analogue functions and the connections between them were made by pass transistors activated by a set of configuration bits. But this approach is not appropriate for analogue circuit applications due to the non-ideal effects of the pass transistors. A new technique using four transconductors is proposed in [61] to eliminate the parasitic effects, and is shown in Figure 2.6. The parasitic capacitance in each transistor is cancelled due to transistor matching and differential signalling. By controlling the drive voltages Vg1 and Vg2, this four-transistor transconductor can be used as, (a)an on/off switch, (b)a polarity change switch and (c)a variable resistor.

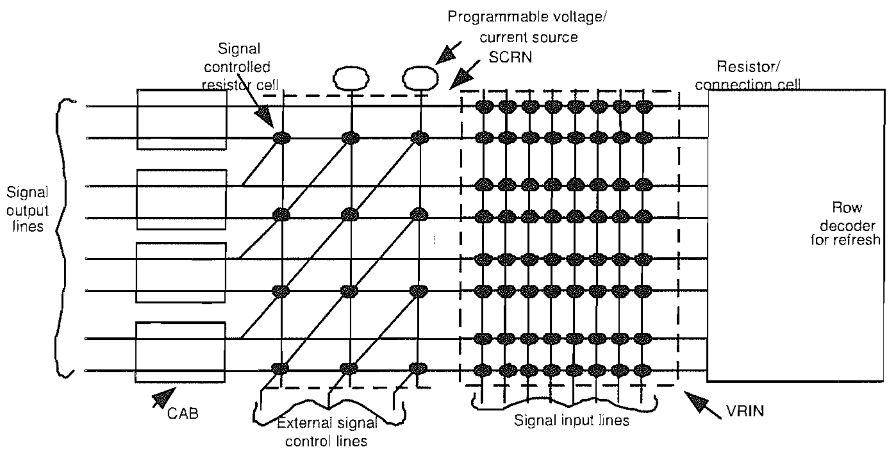


FIGURE 2.5: Architecture of the FPAA, [61]

2.5.3 Summary

An FPAA architecture which consists of crossbar interconnection networks formed by MOSFET transconductors, a multivalued memory system, arrays of CABs and programmable sources is proposed in [61]. It is a parasitic insensitive technique for the implementation of a FPAA. It is not area efficient as proposed, because the interconnection networks also operate as programmable functional elements. The parasitic effects and noise due to the interconnection networks is eliminated by differential signalling. Information regarding any software support and analysis of the flexibility of interconnection network of the FPAA is not mentioned in the literature.

2.6 Continuous-Time FPAA

2.6.1 Architecture

The architecture of this FPAA in [64], uses opamps combined with passive networks to form the functional blocks, which are programmable transfer functions with accuracy which is insensitive to variations in process parameters and environment. The routing method used in this FPAA achieves wide bandwidth due to continuous-time operation. It offers accurate signal transfer without calibration since the resistance of interconnection switches is nulled and the function accuracy is limited only by device matching.

CMOS switches being easy to configure digitally and having a simple dominant parasitic, the ON-resistance (R_{on})[19], are chosen for the routing mechanism. The R_{on} swamps the parasitics of any OFF switches attached to the same node. Elimination of error due to the current flow in the R_{on} of the switches routing the signal between the diverse sources and load impedances of active functional blocks, is important to achieve accuracy for the FPAA operation. In the *switch capacitor* approach, the signals are transmitted as charge

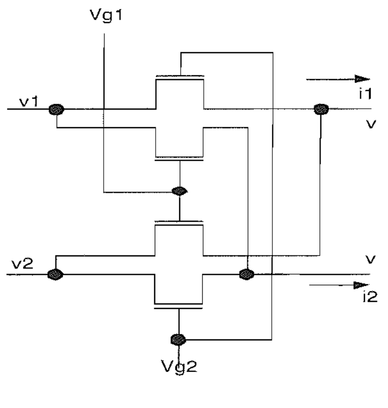


FIGURE 2.6: Four-transistor transconductor, [61]

not subject to any loss due to R_{on}. But its operation on sampled signals requires an anti-aliasing filter, that again constrains the bandwidth. The accuracy of the transfer function depends on the ratio of the capacitors. Thus the advantage of the matching devices available in IC technology is used in this design. In the *transconductor* based design of the FPAA, as seen earlier, the signal is continuous in time and the transconductor is a part of the transfer function. Here there is no loss due to R_{on}, but again the voltage range is limited by the transconductor linearity. The overall transfer function depends on the active device transconductance and passive device transconductance. Thus a combination of both the *switched-capacitor* and *transconductance* methods is used here which provides the following benefits:*(a)the function accuracy depends on the device ratios, (b)signals are continuous time* and *(c)the routing has its R_{on} nulled.*

The continuous-time voltage signals are routed into the high-impedance internal nodes though the CMOS pass-switches. These internal nodes at high-impedance are the inputs of the unity-gain amplifiers. Thus an opamp with unity closed loop gain, maximises the linear voltage range, input impedance and also eliminates the current flow in the pass-switches (cause of inaccuracy in the internal nodes), maintaining a low output impedance (insensitive to loading).

Redundant buffers for external signals are not used so as to avoid the inefficient FPAA utilisation. Thus each buffer is output-switched so that it can be fully switched to every route available. The output switches are provided inside the feedback loop as shown in Figure 2.7. Also note the opamp with unity closed loop gain in the figure. It is thus possible to configure pins to be inputs or buffered outputs, through this approach known as the *Force-Sense* approach, suitable for low impedance loads down to 5kΩ, due to the opamp gain ensuring negligible source resistance. The node buffers are designed to support excess bandwidth compared to the other elements in the signal flow. Using these techniques the inaccuracies associated with switch resistances are nulled.

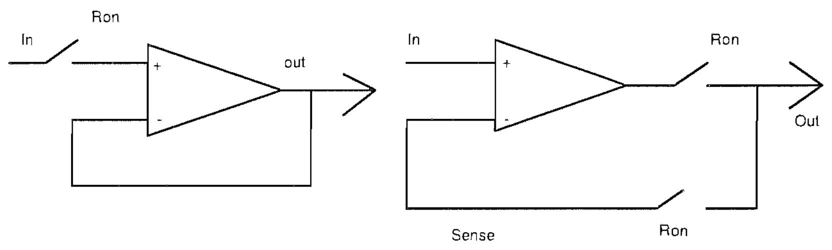


FIGURE 2.7: Buffer and Force-Sense nodes, [64]

2.6.2 Technology

The functional blocks are opamps combined with passive networks to form the functional blocks, which are programmable transfer functions with accuracy insensitive to variations in process parameters and environment. The use of ratios of passive devices as control parameters is more appropriate for achieving accurate transfer functions, rather than the absolute values alone. The use of opamps under passive feedback as functional blocks, is used in order to achieve this accuracy. Programmable parameters are obtained by the appropriate use of passive and switch networks.

2.6.3 Summary

The demonstration of the FPAA is illustrated in [64], where the FPAA combined with configuration routing elements and function blocks was arranged as a 2x2 array of the Adder and Subtractor modules in the fabricated chip. A crossbar network using multiplexer switch boxes, where each multiplexer consists of 8 switches and connected to buffered nodes, allows inputs and outputs of the function block array to be routed. The external analogue interface consists of six pins which connect to the *Force-Sense* nodes in the crossbar network. The configuration data is transferred serially as 118 bits and the configuration register is double buffered, that allows in-service configuration. The configuration stream has read-back facility too. The minimisation of the cross talk and common-mode effects is done by differential switch controls. Thus the opamp and programmable passive network was proposed as a solution to the programmable parameter circuit for a general purpose FPAA. The buffered switch has been examined as a way of eliminating the configuration error due to ON-resistance R_{on}. The measurement results proved the feasibility of this approach in realising a general purpose Field Programmable Analogue Array. Details of any software support and interconnectivity analysis is not mentioned in the literature.

2.7 Other miscellaneous FPAA topologies

Apart from the FPAA topologies based on the switched-capacitors, current-conveyors, sub-threshold circuit techniques, opamps and transconductors, researchers have developed many other FPAA platforms targeting specific applications.

One topology is a fine-grained Field Programmable Transistor Array (FPTA) for reconfigurable hardware and is presented in [87], as an initial effort towards Evolvable Hardware (EH). Automatic synthesis of electronic circuits to produce a desired functionality is demonstrated in the literature. The aspects of using EH for the design of unconventional circuits such as combinatorial circuits for fuzzy logic is also illustrated in the paper. This has addressed the benefits of EH in providing the flexibility and survivability of future space hardware, by showing that evolution can circumvent faults and recover functionality lost due to an increase in temperature.

The second topology is the first known radiation-tolerant FPAA targeting high-reliability areas such as space flight hardware, in [26]. This FPAA design was fabricated with a moderate to high level of connectivity and performance enable by the use of Actel Corporation's low resistance metal-to-metal anti-fuse technology. This FPAA had 12 analogue modules, where each module consisted a differential opamp, 4 programmable resistors (6 values in powers of 2, 1kΩ to 32kΩ), 8 programmable capacitor arrays (6-bit resolution, from approximately 0.5pF to 31.5pF) and 32 CMOS switches. Shift registers were provided at the periphery of the chip, which accepts 2 bits to address an individual fuse by horizontal and vertical position, allowing external access to both sides of the selected anti-fuse for programming. This literature also accounts for the use of low resistance anti-fuse interconnects, by stating that the permitted granularity of the FPAA is a function of the performance of the interconnect. Thus the RAM based analogue arrays tend to have a large granularity such as an entire filter section and allow interconnect programmability only at certain critical points. These RAM based interconnects having higher resistances in the vicinity of 1kΩ can degrade the analogue performance. The low resistance anti-fuse interconnects with a series resistance in the vicinity of 20Ω or lower, permits the granularity of basic analogue building blocks to be reduced to individual components: capacitors, resistors, amplifiers and analogue switches and is also faster for prototyping.

The third topology is a current-mode based FPAA in [74], supported with a high-speed architecture suitable for a wide class of circuits, including continuous-time filters, differential and algebraic equation solvers, dynamic systems and fuzzy and multi-valued logic circuits. The examples in the paper demonstrate an implementation of a ladder continuous-time filter, an analogue rank filter, a circuit for tracking a product of two matrices, a linear equations solver, and a circuit for solving a linear programming problem by the method of steepest descent. High performance is achieved here through the reduced use of global signal interconnections, in favour of local ones. The analogue cell of the FPAA process current-mode and differential signals. The core comprises the Gilbert multiplier circuit and a wide-band, current-mode amplifier, tunable in the range of 0-80 dB. The specific architecture here results from the general premise to use local signal interconnections whenever possible, and global signal interconnections only when absolutely necessary. The FPAA is implemented in bipolar technology.

The fourth topology is a continuous-time FPAA in [15], consisting of digitally configurable transconductors arranged in a hexagonal structure to provide signal routing to neighbour cells and self-feedback without transmission-gates in the signal path. The work presented mainly aims at high-frequency filters using transconductor amplifiers and capacitors are used. Each transconductor builds a G_m-C integrator together with the parasitic input capacitance of the neighbour cell. The literature claims the feasibility of the design for unity-gain frequencies up to 200MHz based on design calculations for 0.25μm CMOS process.

vendor	time-domain	bandwidth
Anadigm [3]	DT	2MHz
Lattice [2]	DT	1.5MHz
Zetex [1]	CT	12 MHz
Cypress [4]	CT	10 MHz

TABLE 2.1: Commercially available FPAA designs

There are also commercially available FPAAs adopting a continuous-time (CT) or discrete time (DT) mode of operation. A list of these available designs is shown in Table 2.1.

2.8 Summary of the available FPAA topologies

The earlier discussion on existing FPAA topologies, their target specifications and applications as in [10, 75, 60, 36, 61, 64], can be summarised as a table shown in Table 2.2. This table can be used to identify the possible design strategies for a programmable analogue circuit, which is discussed in Chapter 3. The comparison of the FPAA topologies is done considering the underlying technology, the mode of operation (continuous-time/discrete-time), bandwidth, targeted applications and their key features. The bandwidth is classified as low (<=50kHz), medium (10kHz-100kHz), wider(100Hz-1MHz) and high (>=10MHz) for the sake of comparison in the Table 2.2. The area consumption and effects of noise parameters cannot be included in the comparison, as these details are not mentioned in the corresponding literature. A comparison on the basis of architecture is also avoided here because they all tend to follow a two-dimensional array based architecture.

2.9 FPAA Interconnect Architecture

The flexibility in configuring an array of CABs depends on the interconnect architecture, which defines the routability between the CABs and the resources involved in the interconnectivity between the CABs. It also defines the amount of routability (throughput) achieved between the CABs and the different types of connection between CABs (direct, general purpose, lengthy) with minimum number of switches in the signal path.

The complexities involved in choosing an appropriate architecture were analysed by considering the various architectures proposed for FPGAs. It is quite common that the signal path delay exceeds the delay of the logic blocks. This is because of the longer path the signal may have to travel through interconnection devices, which possess higher parasitic resistances and capacitances than a regular metal wire. This results in the speed degradation of the FPGA. The high area consumption of interconnection devices

Technology	Continuous/Discrete time approach	Bandwidth	Applications	Key Features
Switched capacitor[10]	discrete	medium	linear analogue signal processing, integrator/differentiator, oscillators	avoids on-chip programmable passive components, requires smaller number of external passive components
Current-mode[75]	continuous	high	integrator, attenuator, amplifier, filtering, data conversion	low voltage, avoids excessive switches in the signal path, reduced parasitic effects
Sub-Threshold[60]	continuous	low	signal multiplier, current comparator, neural networks	low-power dissipation, less resistive switch effects, noise immunity, utilise both differential current-mode and voltage-mode signals
Current-Conveyor[36]	continuous	high	first-order filtering functions, amplifiers, log/anti-log, integrator/differentiator, adder/subtractor, multiplier	requires less number of external passive components, reduced parasitic effects(ground capacitance), rarely requires feedback connections, ensured stability and hence no compensation required, high precision
Transconductance[61]	continuous	medium	filters, integrators, multiplier, divider, adder/subtractor	configurable active/passive elements, the four transistor transconductance to reduce the non-linear and parasitic capacitances
Opamps with passive networks[64]	continuous	medium and wider	programmable adder/gain/subtractor blocks	programmable transfer functions with accuracy insensitive to variations in process parameters and environment

TABLE 2.2: Comparison of FPAA Topologies

in the FPGA is responsible for its lower density. To improve these shortcomings of low density and speed, a hierarchical interconnection structure based FPGA, called the HFPGA (Hierarchical Field Programmable Gate Array) has been proposed in [99, 55]. In these papers, the logical blocks in the HFPGA are connected to form "clusters" and a graphical representation for this hierarchical architecture is also presented in these papers. Also it has been proved in [55] that the total number of switches in an HFPGA is less than the conventional FPGA, under the requirement of equivalent routability.

The architecture of the existing FPAAs and the technology behind each architecture have been discussed in [10, 75, 60, 36, 61, 64]. We observe that a majority of these FPAAs architecture is based on a two-dimensional array based architecture, which consists of rows and columns of CABs. Additional resources are provided for both vertical and horizontal routing between these rows and columns of the CABs in the array. The architecture in [75], following a current-mode based signal processing, aims to minimise the number of programmable devices in the signal path. But all the existing FPAA architectures are dependent on the nature of the CAB and the targeted FPAA applications. There are still other issues to be clarified as to how certain constraints can affect the FPAA architecture and the choice of the CAB. The following sub-sections include a brief discussion on the considered strategies, which are crucial in choosing the target FPAA architecture.

2.9.1 Optimising Routing Resources

The choice of limiting or maximising the available routing resources in the FPAA does affect the maximum routability available between CABs. This again is related to the amount of functional flexibility that can be exploited from a CAB. It is clear that maximising the allowable routing between CABs will improve the efficiency of utilisation of the FPAA. But the optimum solution would be to achieve maximum routability with limited number of resources.

2.9.2 FPAA Routing

Efficient routing plays a very important role in reducing the parasitics introduced by the interconnects. The possible interconnect parasitics are: (a) stray resistances of any wire, (b) stray capacitances of wires to substrate, (c) cross coupling capacitances between parallel and crossing wires respectively, (d) series resistance and parasitic capacitance introduced by switches in the programmable architecture. The compatibility of the architecture to existing routing algorithms, for an efficient FPAA router keeping the performance degradation minimal, is also an important factor. The fact that no existing router can be used in its current form directly to solve the FPAA routing problem is discussed in [32]. The paper presents a new routing algorithm and also discusses most of the relevant issues in routing for array-based FPAAs (Motorola MPAA020 architecture [10]). The router is also demonstrated to be highly efficient, fast and keeps the degradation within acceptable limits.

A key difference as regards to routing between this analogue circuit and a digital circuit is the need for connections to external components to achieve the required behaviour. The other difference would be with respect to number of signal wires. In digital circuits an application is not generally achieved from a single logic block. There is generally parallel processing taking place in a collection of logic blocks. The complexity with respect to the number of switch connections is less for an array of analogue blocks than for an array of logic blocks. Also the analogue blocks require a less dense array compared to the larger cluster of logic blocks required to achieve an extensive range of applications.

The developed FPGA routing algorithms [17, 5], or any of the developed routers for HFPGAs [100], cannot be used directly for the FPAA routing problem. The FPAA routers that have been developed [32, 33], are mainly for single-segment array-based FPAAs, with acceptable levels of performance degradation and maximum routability. Similar issues about the requirement of modification/development of FPAA routing algorithm, and other FPAA interconnect architecture strategies is considered in Chapter 7.

2.9.3 Interconnectivity Analysis

We believe there has been relatively little work on the interconnection strategies for FPAAs. There has been prior work on the interconnection complexity for FPGAs and HFPGAs based on a deterministic method called the Rent's Rule [20]. Rent's rule describes first-order statistics of the terminal-count distribution for different partitions of the circuit. A stochastic model can be derived to describe higher-order statistics. Digital circuits that follow Rent's rule are considered to be statistically homogeneous in circuit topology and gate placement. This term "homogeneous" in digital circuits, implies quantities such as the average wire length per gate and the average number of terminals per gate are independent of the position within the circuit. Hence, the power-law form of Rent's rule can predict the number of terminals required by a group of gates for communication with the rest of the circuit. But at higher levels of the hierarchy mismatches may occur due to heterogeneity. This heterogeneity and its impact on the Rent's rule can be modelled as well.

The Rent's rule states that for different partitions of more or less equally sized modules of a circuit, obtained by a certain net cut minimising method, there exits a power law relationship between the average terminal count T_{avg} and the average module size B_{avg}:

$$T_{avg} = kB^{p}{}_{avg} \tag{2.1}$$

Here, k is the Rent *coefficient* which corresponds to the average number of ports, and p is the Rent exponent. The significance of this rule is due to the self-similarity that exists in real circuits. A high value for the Rent exponent implies more global interconnections and hence a high interconnection complexity. The value of p ranges from 0.47 for regular circuits (RAMs) to 0.75 for complex circuits (full custom VLSI circuits). For modules of larger sizes (in higher levels of the design hierarchy), the number of terminals appears to be lower than expected from equation 2.1. This deviation from the main power law of Rent's rule is known as region II in Rent's rule. This mostly due to technological constraints (pin limitations). A region III of the Rent's rule is also defined. The region III is for those circuits that exhibit a deviation from the Rent's rule for low values of T and B. This occurs due to a mismatch between the average number of block ports and the Rent coefficient k.

Rent's rule is extended in various ways. The module terminals consist of inputs and outputs, and similar empirical relationships can be observed for the module inputs and outputs individually. These extensions prove to be very useful to generate synthetic benchmark circuits to evaluate CAD tools for FPGAs [88]. This rule has been used for evaluating chip architectures before they are manufactured. This rule is also efficient for a prior wire length estimation, where these techniques use the Rent's rule for evaluating the partitioning behaviour of digital circuits. This is later used for predicting variance of the terminal count distribution using a set of benchmark circuits for estimating Rent parameters, typically the `intrinsic` Rent exponent to characterise the quality of the partitioning algorithms.

Methods for estimating the Rent exponent are explained in [20] and references contained there in. But similar methods may not be applicable for analysing the interconnection complexity for FPAA architectures, due to the absence of any established exhaustive set of analogue benchmark circuits. Also, a similar statistical homogeneity as in digital circuits, may not be definable in analogue circuits at a transistor level. Also existing circuit partitioning and gate level placement algorithms for digital circuits may not be useful for the permitted granularity in analogue circuits. This would depend on the chosen granularity of the FPAA to achieve the target application. Hence, the target analogue functionality could be partitioned at an analogue cell level, composed of some simple analogue circuits (opamps, current-conveyors, switched capacitors). Partitioning of an analogue application might be more relevant if target functionality can be separated as a set of individual analogue functions similar to the available granularity in the FPAA (integrator, adder/subtractor, comparator).

Hence, the Rents' rule method of analysing interconnectivity in analogue networks is complex, due to the inherent differences between analogue and digital circuits. Our methodology for establishing relationship between the routability of our FPAA and the flexibility of its hierarchical interconnection structures, not relying on the quality of partitioning/routing algorithms as in digital circuits, is discussed further in Chapter 7.

2.10 Conclusions

As a result of the review of the literature on FPAAs and the summary of them in Table 2.2, certain issues were considered for further investigation and these provide a prospective path to follow in this research project. It is evident from the previous work on FPAAs that, they have been designed by researchers and the industry primarily to target specific applications. This set of target applications is also related to the adopted technology. Hence, the possible reason for the difficulty in proposing a generic analogue building block that can suit different types of analogue applications and specific needs related to power consumption (as in ANNs) or bandwidth requirements (as in Filters). These FPAA implementations tend to follow the continuous-time approach rather than the discrete-time approach due to the better bandwidth performance of the former.

Operational amplifiers seems to be a very common and preferred analogue building block used both in the continuous-time and discrete-time. The use of operational amplifiers might also be due to the convenience of using a very familiar analogue block which has already been proved for implementing very efficient analogue circuits to suit the specific requirements of a target application. But the scenario changes when trying to use the same block as one of an array of blocks in an FPAA, together with a suitable architecture and which can be programmed for the very same target application. The designer now has to think how this array of building blocks can be topologically interconnected to meet the target applications by solving the complexities in analogue design. One significant complexity would be the dependence of an analogue circuit on passive components in order to achieve the desired analogue behaviour. This again might be the reason why previous research has explored relatively little the possibilities and complexities of adopting other architectures for FPAAs other than the simple two-dimensional array.

Having considered a plethora of design approaches for FPAAs, a need for deriving the FPAA design strategies was found to be essential in order to focus on the main objectives of this research project. This would be a better way to categorise the various attempts previously made by researchers and the industry to design FPAAs. A comprehensive discussion on these identified design strategies is included in the next chapter. The design of a prospective CAB, which is highly functionally flexible with minimum dependence on passive components is considered in more detail in the next chapter.

The need for exploring the strategies for interconnecting the prospective building blocks with a suitable interconnect architecture is also to be included in this project. This would include an interconnectivity analysis for a particular FPAA architecture which is to be explored in this research project. Hence, this will show the feasibility of adopting another architecture, for example a hierarchical architecture compared to two-dimensional architecture in FPAAs. This discussion is included in chapter7. In this thesis ideas for an ideal configurable analogue block and for a flexible and appropriate interconnect strategy are explored.

Chapter 3

FPAA Design Strategies

This chapter examines the approaches that may be adopted in designing the FPAA and to develop a strategy for them according to the expected target area of applications. This approach will be to scrutinise possible FPAA design strategies, leading to the choice of an appropriate design strategy suitable for the targeted area of applications. Here these will primarily be *Instrumentation and Data Acquisition Systems.* The expected frequency range of operation for the target FPAA will be at-least 1MHz. The following sections includes a discussion on the possible design strategies for a CAB to be used as a building block in the target FPAA.

3.1 Choice of Discrete-time vs Continuous-time Approach

The choice of operating the programmable analogue block either in discrete-time or continuous-time is one of the main issues. The existing FPAA topologies that follow a discrete-time approach in [10], and continuous-time approach in [75, 60, 36, 61], highlight the advantages and disadvantages of both the approaches. *Discrete-time* circuits are prone to the problems of distributing non-overlapping clocks, required for switches. The accompanied noise in clocks is another disadvantage. An input/output signal reconstruction is also required in most of the discrete-time circuits (smoothing and anti-aliasing filters). Another major limitation is the low bandwidth achieved in using discrete-time circuits. For example, in *Switched-Capacitor* techniques, the input signals are required to be band-limited to at-most one half of the sampling frequency, which causes the limited bandwidth. The attractive feature in discrete-time circuits is that they avoid the need for any on-chip tuning circuit in implementing programmable passive components. *Continuous-time* circuits provide a higher bandwidth of operation. Unlike discrete-time circuits, continuous-time circuits do not require clocks for circuit operation and hence avoid noise due to clocks and switching. The disadvantage of continuous-time circuit is the requirement of external stable and matched resistors (unless CMOS linear, stable and matched resistors are available). This is a major issue in some continuous-time circuits, as the required number of external matched resistors may be large for achieving the desired functionality. This problem can be observed in operational amplifier based circuits. Keeping these resistors external in continuous-time circuits is a good feature to avoid on-chip resistive effects dependent on temperature and process parameters varying gradient across the chip, but has implications for pin-out, parasitics and complexity for external connections.

3.2 Choice of Voltage-mode vs Current-mode Circuits

The choice of using voltage or current as the signal parameter in the implementation of the programmable analogue circuit is another important strategy to be considered. This choice has been considered in the existing FPAA topologies operated in voltage-mode [10, 60, 61, 64] and current-mode [75, 36]. *Voltage-mode* circuit techniques provide the advantage of a high fan-out due to the high input and low output impedance respectively. However the recent practise of using lower power supply voltages, has reduced the dynamic range available in voltage-mode circuits. Voltage-mode circuits are not very attractive for applications requiring a high-bandwidth (video applications). *Current-mode circuits* provide the advantages of good dynamic range. They are also attractive in implementing circuits operating at higher frequencies through current-mode amplifier circuits. Hence current-mode circuits are ideal for implementing filters. Current-conveyor circuits are good example for current-mode filter applications, as in [90, 30]. The beauty of handling analogue signals as currents in current-mode circuits is that it makes it easier to handle operations like addition, by just wiring the signals together. The flexibility that current-mode circuits like the current-conveyor based applications provide in [60, 94, 82, 103, 83, 71] is a significant advantage. This again reduces the complexity in *routing/interconnections* and the *number of internal/external passive components.*

3.3 Choice of Single-ended vs Fully-Differential Signals

The choice between circuit techniques using single-ended signals or fully-differential signals is another issue that relates to the number of internal nodes across which the signals are applied or taken out. Fully differential design technique has become popular for analogue integrated circuits. This technique has been applied in sub-circuits like opamps [14, 66, 70], current-conveyors [28, 27, 40], switched capacitor circuits [67, 79]. Functional circuits that process fully differential signals have the following advantages over sub-circuits using single-ended signals:-

- Cancellation of all the even harmonics if perfectly symmetric circuits are used (second order harmonics are often the dominant cause of distortion).

- They are insensitive to sources of disturbances (power supply coupling and electromagnetic interference) that tend to enter the circuit in a common-mode fashion.
- The signal to noise ratio is significantly larger than a single-ended circuit since the output signal is taken across two internal output nodes/ports.

The two main drawbacks of fully-differential circuit techniques are:-

- The requirement of the common-mode feedback, as only a differential output is obtained from every active component (opamp, transconductance), which is function of only the differential input [96]. There are no mechanisms in the circuit to set the common-mode voltages. This inevitably leads to other problems of: (a)increasing the chip area, (b)increasing the power dissipation, (c)instability caused by this extra feedback loop, (d)requiring a high gain and bandwidth equal to the differential-mode part of the circuit.
- Increasing the complexity of the target FPAA interconnect architecture. The use of fully-differential signals internally increases the number of internal switches and wires that carry signals within the circuit or between circuit blocks. This may also increase the total number of required output signal wires to be brought outside the chip for any external circuitry.

Incorporating a circuit feature for accepting differential input signals is definitely better than having only single-ended input ports. This is because processing of differential voltage signals is very common in applications such as automatic control and instrumentation systems. Single-ended input circuits can definitely reduce the complexity in the routing architecture when compared to fully-differential circuits. The dynamic range of the input signal decreases with DC offset and/or noise in the signal path. Single-ended circuits mostly require additional signal conditioning circuitry as they are more susceptible to coupled-noise and DC offsets.

3.4 Choice of circuits for the CAB Design

The following sections contain an investigation of the possible analogue circuit blocks and their *granularity* which may be suitable for the CAB design. This investigation on CAB granularity was done by considering the advantages and disadvantages of the strategies discussed in sections 3.1, 3.2 and 3.3. The approach involved identifying the target features and applications the FPAA should address and then inspecting the possibilities of achieving them through the common building blocks used in analogue circuit design (operational amplifiers, current-conveyors, transconductors). The area of target applications was chosen to be for Instrumentation and Data Acquisition Systems operating at frequencies up to at-least 1 MHz, with limited number of external components

Category	Functions
General Purpose circuits	Multiplexing, Filtering, Signal Conversion (I to V and vice-versa), Gain Control,Sample and Hold, Modulation and Demodulation, Comparator,Rectification and Schmitt Trigger
Signal Sources	Sine wave,Triangular/Sawtooth wave, Square Pulse and Monostable-Pulse Shaping
Control	Addition/Subtraction, Integrator/Differentiator
Miscellaneous	Multiplier/Divider, Log/Anti-log, Square and Square Root

TABLE 3.1: Target Applications

to provide accurate and stable operating characteristics. Table 3.1 illustrates some of the target functions and applications the FPAA may be expected to fulfil. The realisation of these functions using the common building blocks is considered in the following sub-sections. An integrator is a good example to discuss the suitability of the analogue circuits like operational amplifiers, transconductors and current-conveyors.

3.4.1 Opamps

The integrator circuit shown in Figure 3.1, is the classic implementation using an opamp. This circuit has capacitive feedback, which is not very attractive, as the parasitic capacitance affects the circuit behaviour. Feedback connections through passive components are prominent in closed-loop opamp configurations, to satisfy the target functionality. This ultimately leads to an increase in the number of internal/external passive components and thereby making it more complex to realise the target applications. Another possible solution would be to implement opamp circuits using the switched-capacitor techniques.

The switched-capacitor circuits using operational amplifiers, switches, capacitors and non-overlapping clocks provide a good solution for analogue signal processing with good linearity and dynamic range. But again the quick charging/discharging of the capacitances (on-chip/off-chip) for switched-capacitor circuits, with proper distribution and phasing of the clocks is complex. The requirement for switches also forces the designer to consider carefully the possible non-ideal switch effects (nonlinear capacitance and channel charge injection effects).

3.4.2 Transconductance

The integrator can be implemented using the transconductance both in current-mode and voltage-mode as shown in Figure 3.2 and Figure 3.3 respectively. The requirement of fewer passive components in these circuits when compared to the opamp based integrator circuit, is quite obvious. This reduction in the number of internal/external passive components is an attractive feature of using transconductance circuits. However the high output impedance in these circuits may lead to low fan-out problems. Hence they usually require additional current/voltage output buffer stages. The ability to have grounded capacitances in these circuits is a useful feature to reduce parasitic effects on the circuit behaviour. They also rarely require feedback connections through passive components to achieve the desired functionality. The use of transconductance elements in place of operational amplifiers to realise analogue circuits that perform at higher frequencies and dissipate lower power is an efficient technique [37]. There have been many approaches to realise linear transconductance elements, which employ transistors operating in the saturation region [93, 18, 69, 81, 84, 53, 54, 89]and MOS transistors operating in the triode region [95, 101, 86]. The transistors in these methods show a large second-order nonlinearity in the voltage-current relationship, with higher-ordered nonlinearities due to effects such as body effect, channel length modulation and mobility reduction. Fully-differential transconductors have proved to be better for analogue signal processing, with good linearity and high dynamic range [97]. The complex routing with more signal paths and switches within the CAB and between the CABs in fully-differential transconductances can be predicted. This makes the differential transconductor less suitable for use in an FPAA.

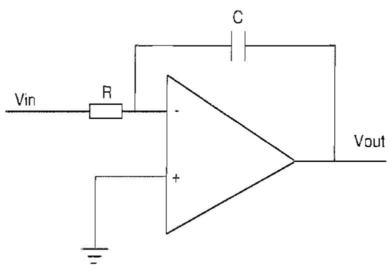


FIGURE 3.1: Operational amplifier based Integrator

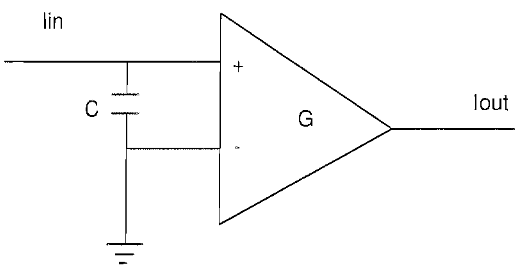


FIGURE 3.2: Current-mode Transconductance Integrator

3.4.3 Current-Conveyors

The implementation of the integrator using a current-conveyor is shown in Figure 3.4. The current-conveyor approach may be a more appropriate method to implement an integrator when compared to the opamp and transconductance circuits. This is because of the good dynamic range and the higher bandwidth current-conveyors provide. The reduction in the parasitic effects by having no capacitive feedback connections (mostly grounded capacitances) is another attractive feature. Current-conveyors have proved to be very efficient in implementing a range of circuits in [94, 82, 103, 83], requiring fewer feedback connections and external passive components to achieve the desired functionality.

The suitability of the circuits considered above and other possible candidates depends upon the flexibility with which they may be internally reconfigured to provide a range of analogue functions. Hence, the chosen granularity could even be a fine-grained transistor array presented in [87], to support evolvable hardware using evolutionary/genetic mechanisms on Field Programmable Transistor Array (FPTA), aimed for spacecraft based applications. This is a trade-off between the expected functional flexibility and the choice of the CAB granularity. Again, irrespective of the CAB granularity, its extent of dependency on matched external/internal passive components is crucial for designing the FPAA architecture.

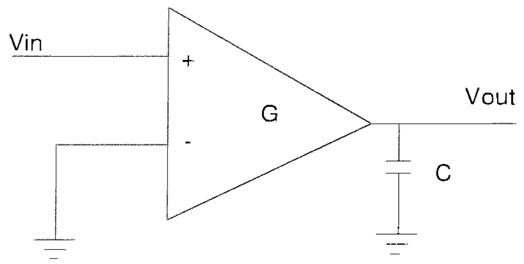


FIGURE 3.3: Voltage-mode Transconductance Integrator

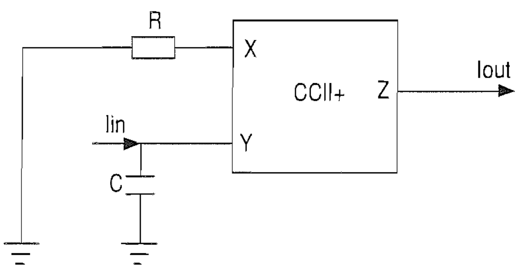


FIGURE 3.4: Current-mode Integrator using a second generation current conveyor

3.5 Internal/External passive components

The *minimum number of internal/external passive components* required for achieving the desired analogue behaviour from the CAB is an important issue not addressed in detail in previous FPAA topologies. The following are three possible options to consider: (a)switched internal passive components, (b)external passive components and (c)voltage/current controlled variable components.

The choice of keeping the required passive components internal to the circuit depends on whether it is resistive or capacitive in nature. Incorporating resistors within the circuit using a standard IC process is not a good practice due to the poor stability of IC resistors with changes in temperature and process parameters respectively. The design of internal resistors has to be handled with great care to minimise these effects. Those capacitances crucial to good circuit performance are best kept internal if they do not consume excessive chip area. This is generally true for switched-capacitors circuits that use small values of capacitances. Larger value capacitances or those whose value may need to be varied are sometimes made available external to the chip to save chip area. The other option would be to use external voltage/current controlled variable components if the required tuning range is quite limited.

Maximum flexibility in configuring a CAB, an array of CABs and switching between two configurations, can be achieved if the circuit behaviour of the CAB has minimal dependence on internal/external passive components to achieve the desired functionality. This simplifies the routing architecture for an array of CABs. It also reduces the area consumed by avoiding the embedding of passive components (programmable passive components in some circuits) into the programmable analogue circuit. A similar feature can be observed in current-conveyor circuits, where complex applications like filters [90, 94, 82, 103, 83], can be achieved with a smaller number of external components and routing. Also keeping one end of components connected to ground simplifies routing and minimises the effects of parasitic capacitance. Hence the complexity in designing an interconnect architecture featuring more routability between the programmable analogue blocks is related to the required number of internal/external passive components for the programmable analogue block.

3.6 Interconnect Architecture

The complexity in designing the interconnects for the programmable analogue circuits will depend on the following:-

- Design of the individual CAB
  - required number of input/output voltage/current ports.

  - required number of feedback connections within the CAB to suit functionality.
  - required number of internal passive components.
  - requirement for switchable components.
- Required number of external passive components.
- Requirement for additional routing for power supplies(analogue and digital) or clock distribution trees (switched-capacitor circuits).
- Required flexibility in switching between configurations, achieved through maximum routability between CABs.
- Number of signal input/output pads available in the periphery of the chip.
- The restrictions of the technology used
  - available number of metal layers for the efficient layout of the signal buses, active and passive devices.
  - careful layout of the routing architecture considering the possible variation of the process parameters.
  - efficient routing solutions to avoid cross talk, bottle necks and minimise interconnect routing capacitances.

A comparison between switched-capacitor and current-conveyor circuits for the complexity involved in routing may highlight the key points listed above. Non-overlapping clocks is one of the main requirements in switched-capacitor circuits. The distribution of clocks in [76, 31, 22] through appropriate clock tree structures to minimise skew (overall delay from the clock generator to all the destinations) and additional clock buffering stages to facilitate a good fan-out increases the complexity of the overall FPAA routing architecture. Current-conveyor circuits avoid such routing complexities, because they are quite flexible in implementing a range of circuits with minimal dependency on internal/external passive components. They do not require any clock distribution networks apart from the proper power supply routing. A comprehensive discussion on the design of the target FPAA interconnection architecture is included in Chapter 4.

3.7 Decisions on design strategies

Decisions on the choice of FPAA design strategies discussed in the previous sections were made to make progress in the design of the target FPAA. These decisions were made considering the advantages and disadvantages of the possible design strategies and also the targeted area of applications. The *first* decision was to adopt a continuous-time approach for designing the target FPAA. The main reason behind this decision is to satisfy the following:-

- to achieve a higher bandwidth of operation
- to keep the required passive components external and minimise the number
- to avoid dependence of the circuit behaviour on clocks and switching noise
- to avoid the requirement of any form of signal reconstruction circuits (smoothing or anti-aliasing filter circuits)

The *second* decision was to incorporate both voltage-mode and current-mode of operation for the CAB. Thus it is possible to achieve the following:-

- to achieve good dynamic range in current-mode circuit
- to achieve high fan-out in voltage-mode to compensate for the low fan-out in current-mode circuit
- to achieve higher bandwidth provided by the current-mode circuit
- to minimise the number of external components and passive feedback connections, as in current-mode circuits

The *third* decision was to incorporate the facility for supporting fully-differential input signals in to the CAB. The CAB would avoid fully-differential signals at the output, and would have only a single-ended output signal.

The three main decisions mentioned above can be summarised again, as the requirement of a continuous-time, hybrid mode (voltage and current) circuit with full-differential inputs and single-ended output, for the design of the CAB in the target FPAA.

The available sub-circuits used in analogue circuit design were considered again with respect to the decisions made on the design strategies and the target FPAA applications. Current-conveyor circuits seem quite attractive in

- following a continuous time approach
- providing maximum flexibility in switching between applications/configurations
- requiring fewer internal/external components
- providing a good dynamic range
- providing a high precision observed in the analogue behaviour
- providing a higher bandwidth of operation
- reducing parasitic effects by having grounded capacitances rather than capacitive feedback connections

Considering the above advantages, current-conveyors can been used in implementing the FPAA as in [36]. These sub-circuits have certain advantages and disadvantages discussed earlier in Section 3.3. One other disadvantage is that they do not support fully-differential input voltage signals and can support only one input voltage, while processing of differential voltage signals is very common in analogue signal processing systems. Design proposals of using low-voltage Operational Transconductance Amplifier (OTA) for novel CMOS realisations of second generation conveyors with single and differential inputs have been developed [106]. There have also been other novel methods of implementing full-differential current-conveyors for maximum linearity, good dynamic range and common-mode rejection as in [28, 40, 27]. But considering the related complexities involved in implementing a routing architecture for the target FPAA, it is not feasible to use fully-differential circuit techniques. So it would be an advantage to have a hybrid sub-circuit that can handle fully-differential input signals and also have the ability to operate in both voltage-mode and current-mode depending upon the varying configuration of a CAB or an array of CABs in the FPAA. Hence the feature of switching between voltage-mode and current-mode of operation, is an effective method to satisfy the range of target applications, depending upon the specifications of the target design. The designed CAB should have the minimum number of switches and signal paths, without affecting the range of applications covered. It should also reduce the complexity of the routing architecture without affecting the maximum routability between CABs for the target FPAA. *So the real challenge in this project is the design of a target FPAA chip in a suitable technology that can address all the design strategies and design constraints discussed.*

An investigation was done on other possible sub-circuits used in instrumentation based applications. A literature review [47] by *J. H. Huijsing* was found, that has a comparative study on various monolithic circuit solutions for Instrumentation amplifiers. One of the proposed circuits introduces the extension of the opamp circuit called the differential difference amplifier (DDA). The DDA compares two differential input voltage signals, when only single-ended voltages are compared in an opamp. In a closed-loop environment this circuit forces two floating voltages to the same value, and thus has many interesting applications in the analogue circuit domain. More developments of the DDA have been investigated and another paper by E. Säkinger and W. Guggenbühl, [80] was found. It describes the DDA as a versatile building block with infinite input impedance, low output impedance and a requirement of fewer matched components/devices (external) when compared to the opamp. This is illustrated with a few examples on DDA based applications in [80], namely:(a)Comparator with floating inputs, (b)Level shifter, (c)Voltage inverter without external resistors and (d)Instrumentation amplifier

The most attractive feature in the above demonstrated applications is the requirement of very few external components and the flexibility achieved in switching between these applications. For example, a common opamp based implementation of an instrumentation amplifier requires three opamps (two opamp designs also exists) and several matched resistors. An instrumentation amplifier can be realised using just one DDA block and only two resistors that determine the gain [80]. Although a less well known circuit than the opamp, the DDA can perform many functions namely, rectification circuits [98], instrumentation amplifier [46, 34], comparators [78, 56], data conversion circuits [45], filters [42, 102] and other miscellaneous circuits [105, 9, 48, 44, 25, 51, 23, 65, 43, 45, 24, 85, 34, 21, 91, 77, 92, 58, 7, 8]. This illustrates the versatility of the DDA as a flexible analogue circuit for implementing a range of functions with fewer external connections and matched components. Hence it was decided to explore and experiment more with the DDA and analyse the possibilities of the DDA concept reconciling them with the FPAA Design strategies discussed earlier. The next section includes more about the DDA as a versatile building block for analogue circuits.

3.8 Introduction to DDA

The DDA can be considered as an extension of the opamp. An opamp with a negative feedback loop, adjusts its output in order to reduce the differential input voltage to a negligible value. An opamp compares two ground referred voltages. The DDA extends this by comparing two differential voltages. The symbol for a DDA is shown in Figure 3.5, where the input terminal voltages designated as V_{pp}, V_{pn} for the non-inverting input port and V_{np}, V_{nn} for the inverting port. The differential input ports are V_{pp} - V_{pn} and V_{np} - V_{nn}. Thus the output of the DDA is given by the equation 3.1:

$$V_o = A_o\left[(V_{pp} - V_{pn}) - (V_{np} - V_{nn})\right] \tag{3.1}$$

where A_o is the open-loop gain of the DDA. On introducing a negative feedback to V_{pp} and/or V_{np}, in the above equation, then the following equation 3.2 is obtained:

$$V_{pp} - V_{pn} = V_{np} - V_{nn} \text{ with } A_o \rightarrow \infty \tag{3.2}$$

The difference between the two differential voltage increases as A_o decreases, for a finite open-loop gain A_o. The function of the DDA can be represented in the form of a block diagram as shown in Figure 3.6.

From the Figure 3.6 we observe that the DDA is implemented using two V-I converters, that convert the two differential voltages into currents, which are then subtracted, converted into voltage and amplified. Again from equation 3.2 we see that, even if the difference of the two differential input voltages is virtually zero, the single-ended voltages V_{pp} and V_{pn}(V_{np} and V_{nn}), do not exhibit the virtual short circuit property, as in opamp circuits. The DDA cannot be considered as a combination of three opamps. The first stage amplifiers have relatively low gain to permit a significant differential input voltage. The Gm's must also match to provide an accurate open-loop gain.

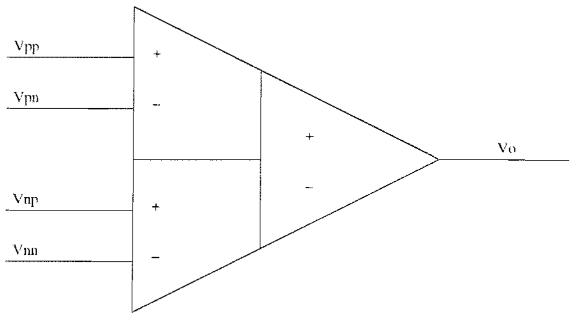


FIGURE 3.5: Symbol of the DDA

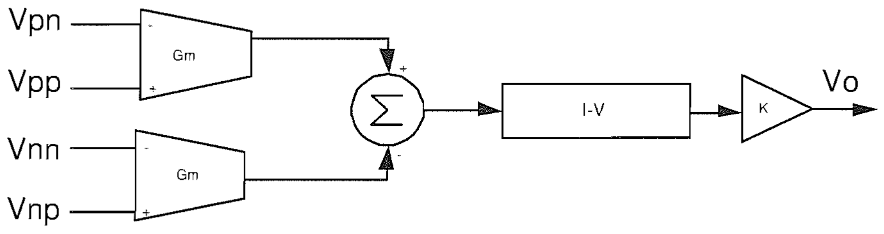


FIGURE 3.6: Block Diagram of the DDA

Again from Figure 3.6, the two V-I converters are obtained from the transconductance stage and the subtraction of the output currents is easily carried out by direct and cross connection of the differential stages to the summing buses (Σ +, Σ -). Finally the differential current is amplified from the bus by the output block by a large value (K). The CMOS realisation of the DDA is shown in Figure 3.7.

The two differential stages (M1,M2,M3,M4) form the transconductance elements. The current sources (M5,M6,M11,M12) are implemented using the cascode techniques widely used in analogue circuits. Thus achieving a good matching between the tail currents of the two differential stages and an increase in the output resistance of the current sources. The high gain stage is achieved through a current mirror (M7,M8), that converts the differential current of the bus to a single-ended current. The output stage (M9,M10,M15,M16) implemented here is a class AB stage. The biasing of M10 is incorporated into the design using M17 and M18. The above circuit was simulated using PSpice with MOS models having the process parameters for a $0.5\mu m$ process. The transistor dimensions are shown in Table 3.2.

As the two inputs of each of the two differential pairs in the DDA are not virtually shorted and the differential pairs do not operate at a fixed V_d (differential input voltage), as V_d increases, the transconductance of the differential pair decreases and becomes zero for $|V_d| \geq \sqrt{\frac{2Ibias}{K}}$. Thus if K or $(\frac{W}{L})$ of the differential pair transistors of the DDA is small, then the input range, noise and distortion is large. The input range increases with decreasing $\frac{W}{L}$. The input range could also be increased along with the decrease in noise

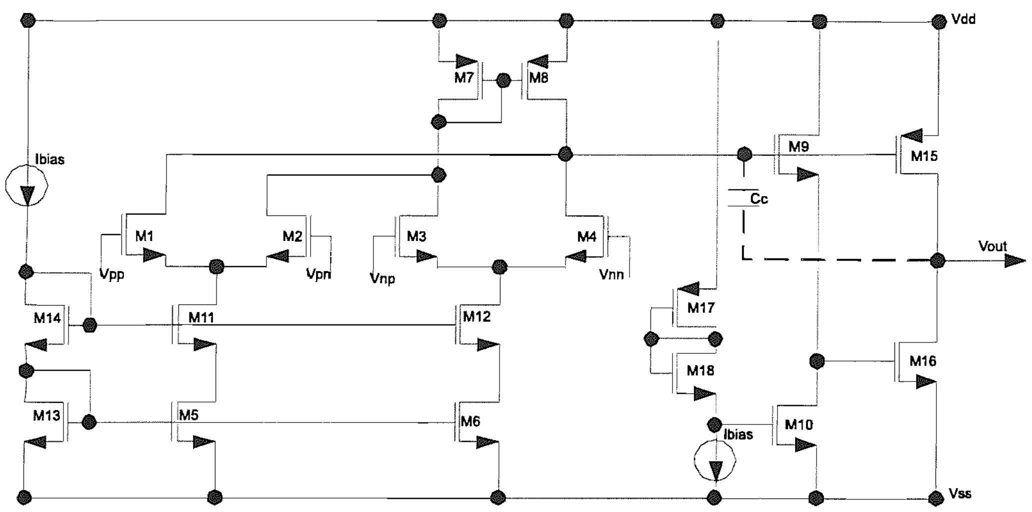


FIGURE 3.7: class AB realisation of the DDA

Transistor	Type	Width	Length
M1 M2 M3 M4	N	2μm	10μm
M5 M6 M11 M12 M13 M14	N	15μm	2μm
M7 M8	P	20μm	10μm
M9 M10 M16	N	30μm	2μm
M15	P	90μm	2μm
M17	P	2μm	2μm
M18	N	2μm	2μm

TABLE 3.2: Transistor Dimensions

Parameter	Experimental Results
Differential Gain	58dB
Phase	-90degrees
GB	1MHz
CMRR	75dB
Output Swing	-5V to 5V
Offset Voltage	-4mV
Slew Rate without Compensation Capacitance	400 $\frac{V}{\mu s}$
Power Consumption	5.96 mW
Output Resistance	600Ω
Harmonic Distortion as a function of output voltage	1.2%

TABLE 3.3: Summary of the DDA Parameters

and nonlinearity, by increasing the Ibias, which is not ideal for low-power applications. Thus in Table 3.2 *the smaller widths of the differential input transistors is a trade-off between achieving high input differential range, low noise and low distortion in the DDA design.* The class AB DDA was simulated for the parameters summarised in Table 3.3, using PSpice with the transistor dimensions in Table 3.2, power supplies of Vdd = 5 V and Vss = -5V and a bias current of Ibias=20μA.

3.9 DDA Based Applications

The DDA based applications in [80, 46, 9] were validated using Spice simulations and a few applications with the corresponding simulation results included in Appendix A (*appropriate page numbers of the figures in Appendix A are available in the List of Figures at the begining of this thesis*) are discussed here.

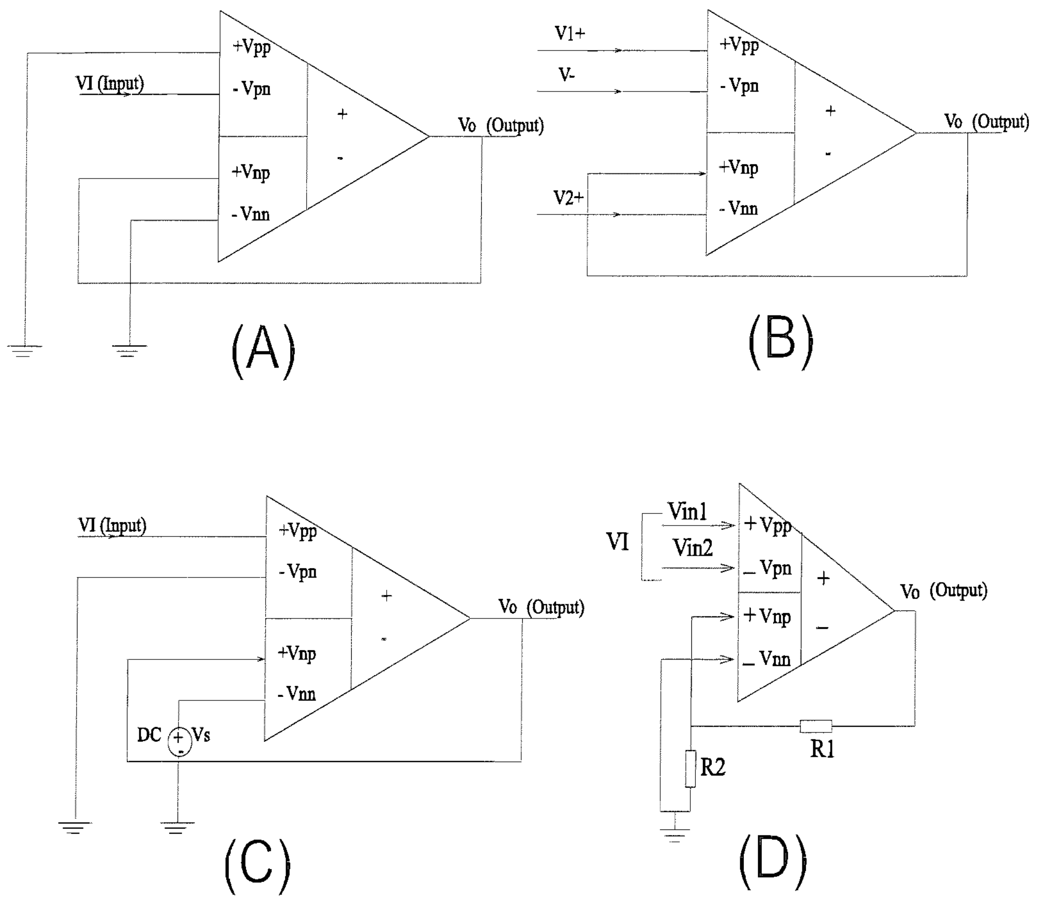


FIGURE 3.8: DDA Configurations as an (A) Inverter, (B) Adder/Subtractor, (C) Level Shifter and (D) Instrumentation Amplifier

3.9.1 Voltage Inverter

A voltage inverter without any external matching resistors, can be implemented as shown in Figure 3.8(A). Thus the realisation of the inverter with the DDA is achieved with a few additional connections and no external components, unlike using opamps that requires two external matched resistors. The PSpice simulation result is shown in Figure A.2. The configuration shown in Figure 3.8(B) can be changed to that of a *level shifter*, by adding a DC voltage V_S at the input port V_{nn}, thus shifting the input voltage V_I by the voltage V_S resulting in V_O.

3.9.2 Adder/Subtractor

The adder/subtractor configuration can be achieved using the DDA as shown in Figure 3.8(B), with $V_{pp} = V_{1+}$, $V_{pn} = V_{-}$, $V_{nn} = V_{2+}$ and $V_{np} = V_O$ (negative feedback). The output of the DDA is thus given by equation 3.3.

$$V_O = V_{1+} + V_{2+} - V_{-} \tag{3.3}$$

This circuit configuration can be used for the addition/subtraction or both, of the input signals, without the need of any additional components. The same circuit in Figure 3.8(B) can be used to realise other circuits. For example, an *inverter* can be achieved again by grounding V_{1+} and V_{2+}. The simulation of the DDA as an adder is shown in Figure A.4. A *voltage doubler* can be realised by connecting $V\text{-}_{1+}$ and V_{2+} as the input and grounding the V_{-} node. The simulation result with a 1.5V amplitude sinusoidal input at 1kHz is shown in Figure A.5. The realisation of the *level shifter* and a voltage subtractor can also be achieved using the same circuit configuration in Figure 3.8(B). The DDA as a level-shifter is shown in Figure 3.8(C) along with the simulation result in Figure A.7. The simulation result of the DDA subtractor is shown in Figure A.8.

3.9.3 Instrumentation Amplifier

An instrumentation amplifier can be realised using one DDA and two resistors as shown in Figure 3.8(D), compared to the normal implementation using opamps that requires at-least three opamps and several matched resistors. The gain of the DDA based instrumentation amplifier is determined by the two resistors, i.e. $\frac{(R1+R2)}{R1}$. The simulation result is shown in Figure A.10. In all the simulation and the verification of the DDA as an adder/subtractor, voltage doubler, voltage inverter, level shifter and instrumentation amplifier, the following can be observed:-

- requires very few external components
- rarely requires feedback connections through passive components
- requires single feedback connection for all the applications
- requires very few changes to switch between any of these applications
- avoids parasitic effects by keeping most of the signal path to ground

The above features are in favour of the decisions made on design strategies in section 3.7. The most attractive feature is the flexibility of the block and the ease in switching between configurations with very few changes. For example the inverter circuit and the adder/subtractor circuit can be switched between each other by removing the grounded connections to Vpp and Vnn, and applying appropriate inputs based on the equation 3.3 for an adder/subtractor/voltage doubler. This feature would also help in achieving less complex routing architecture if we choose the DDA as the CAB for the target FPAA. The DDA block also supports the input differential stage. The larger number of input ports might lead to the need for optimising the total number of input pins for the target FPAA, by sharing of resources through input multiplexers. The same solution might be applicable for other switching resources and output blocks. But this issue of optimising the resources to reduce the complexity of the chip architecture requires an analysis of the interconnect architecture, which will be discussed in Chapter 7. The DDA does not support current-mode of operation. But probably the current-mode of operation could be incorporated into the DDA block to achieve the flexibility of voltage-mode and current-mode of operation. Considering the attractive features of the DDA block, it was decided to proceed further into the design of the DDA aiming CMOS 0.6 μm technology. A suitable analogue circuit design tool, Cadence 4.4.5, which provides a front to back design environment was chosen. The design was targeted for AMS CMOS 0.6μm CUP process to analyse the feasibility of the DDA design as a CAB in detail and layout the target FPAA chip for fabrication.

Chapter 4

Design of the FPAA

Observing the attractive features of the DDA and foreseeing the possibility of using the DDA as the CAB for the target FPAA, it was decided to research further on the circuit design of the DDA. Thus in an effort to make the DDA feasible and reconciling to the decisions made on design strategies in Chapter 3, various circuit techniques were explored for the DDA. This chapter includes the design of the DDA and the various design decisions taken after suitable simulations. The whole circuit design was done using Cadence 4.4.5 (a very popular custom IC design tool). The Analog Artist tool in Cadence 4.4.5 was mainly used to design the circuits for the DDA and test circuits for simulating appropriate performance characteristics of the DDA. The suitability of the DDA for the target applications was also tested. This chapter includes the simulation results of those DDA applications, which have been tested. The rest of the exhaustive set of schematics and graphs are included in Appendix B (*appropriate page numbers of the figures in Appendix B are available in the List of Figures at the begining of this thesis*). The DDA block was then used to build an array of CABs using suitable resources (switch blocks) and interconnect architecture to form a FPAA. The design of these additional resources and the important decisions on a suitable interconnect architecture are also discussed in this chapter. The chapter concludes with the design of the target FPAA suitable for the next phase, which is the physical layout of the target FPAA chip for fabrication.

To start with the design of the DDA, few design goals were set. It should work with a 5V power supply as per the anticipated AMS 0.6$mu$m 5V CMOS process. The output voltage swing should be -1V to +1V with a bias current of about 10$mu$A. A bandwidth of atleast 1MHz should be achieved. The DDA should also be able to handle common-mode input voltage from -1V to +1V. The open-loop gain should be reasonably high to approximately 80dB. Low power consumption is not considered as a key goal here and hence not focussing on circuit techniques for low power design. To keep it simple

Transistor	Type	Width	Length
M1 M2 M3 M4	N	5μm	10μm
M5 M6 M11 M12 M13 M14	N	16μm	2μm
M7 M8	P	40μm	2μm
M9	P	88μm	2μm
M10	N	28μm	2μm
M15 M17	N	28μm	2μm
M16 M18	P	28μm	2μm

TABLE 4.1: Transistor Dimensions for Figure 4.1

quiescent currents of the order of 10uA would be fine. Voltage mode analogue applications are targeted, but would definitely consider extending the range of applications to current mode too.

4.1 Design of the DDA

The basic DDA block that follows the conceptual block diagram in Figure 3.6 (section 3.8, Chapter 3) was designed using the AMS 0.6μm, 5V, CMOS library. The transistors are labelled from M1-M18. The circuit is shown in Figure 4.1 and the transistor dimensions in Table 4.1. The transconductance elements of the DDA are realised with two differential stages M1-M2 and M3-M4 respectively. The unusual geometry (W<L) of these transistors is a result of the trade-off between achieving high input voltage range, low noise and low distortion (discussed earlier in section 3.8, Chapter 3). The current sources (M5, M6, M11, M12, M13, M14) are implemented use cascode technique described in [11]. This provides good matching of the tail currents. The current mirror (M7-M8), converts the differential current from the input transistors to a voltage to drive the common source high gain stage M9. The output buffer stage is a traditional class AB amplifier (M15-M16) in [12] aiming at low output impedance, low power dissipation and large dynamic range. The gate of M9 is biased by the active load of the input differential pairs and the gate of M10 (current source) is biased by the same bias voltage for the differential amplifier current source. All the transistors are operating in the saturation region with sources and substrates connected to VSS for NMOS and to VDD for PMOS transistors respectively.

4.1.1 DC Response of the DDA

The circuit in Figure 4.1 was simulated for the DC response of the output voltage. The power supplies for the test circuit, VDD and VSS was set to 2.5V and -2.5V respectively. The bias current Ibias1 was set to 20μA. The output swing was observed linear between 0.79V and -1.4V as shown in Figure 4.2.

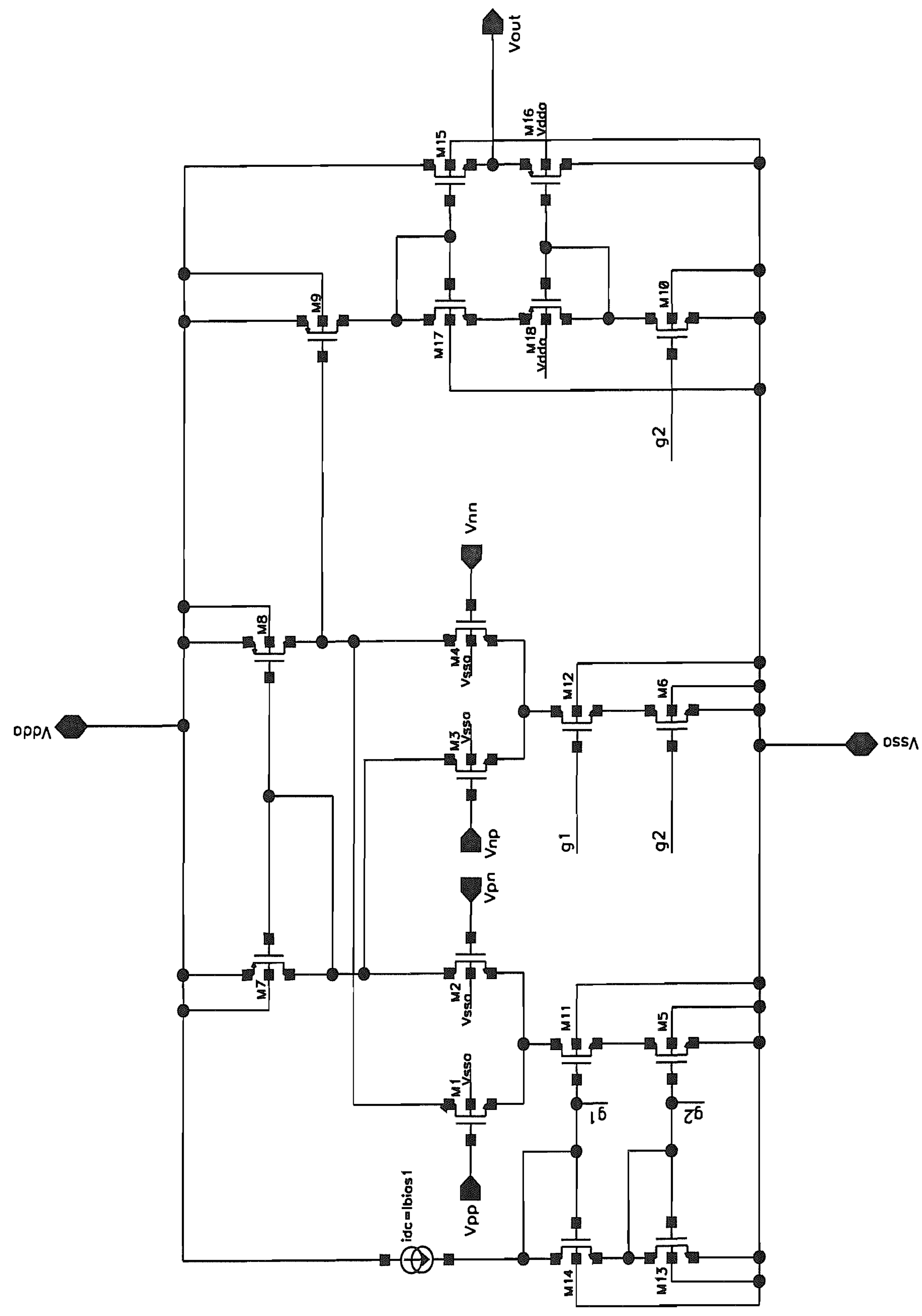


FIGURE 4.1: Class AB CMOS Realisation of the DDA

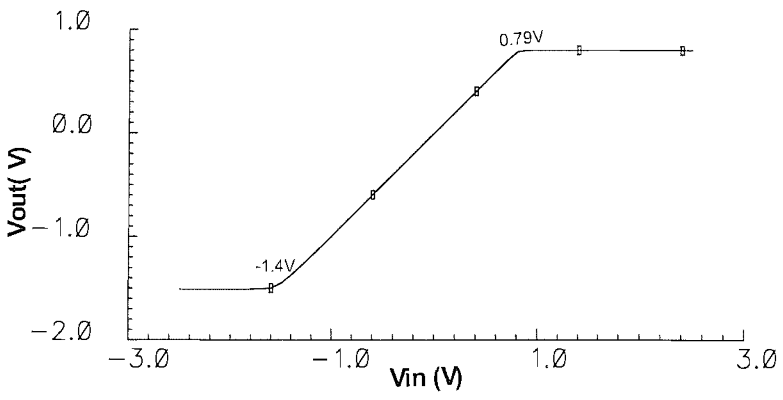


FIGURE 4.2: DC Response of the DDA

4.1.2 The Class AB Output Stage

The use of a class AB output stage enables the quiescent current to be kept low while still enabling large currents to be supplied to the load. This avoids problems of low slew rates when driving a capacitance load. However, we observe that the class AB stage in Figure 4.1 provides a limited voltage swing because of the common drain configuration and the necessary bias voltages. A common source configuration for the output stage will permit a wider voltage swing. A common source circuit will have higher output impedance, but this is reduced by global feedback. A more significant issue is that the drive for the output transistors needs level shifting, which must be compensated for changes in supply voltage, temperature or process parameters to ensure a stable quiescent current. A compensated, low power class AB output stage was given in a paper by *H. Elwan and M. Ismail* [29]. The output stage circuit is shown in Figure B.1. The analysis of how this stage compensates for voltage, temperature or process variations is not included in the paper, so it was considered and is included in Appendix B. Copying the equations B.8 and B.9 here again,

$$\sqrt{I_q}\left(\sqrt{\frac{2L_1}{K_pW_1}}+\sqrt{\frac{2L_2}{K_nW_2}}\right)=VDD-2V_{Tn}+V_{Tp}-\sqrt{\frac{2L_3}{K_nW_3}}\sqrt{\frac{K_nW_4}{2L_4}}(V_3-V_{Tn}) \quad (4.1)$$

Now, considering the transistors M5 and M6, the voltage V$_3$ is given by:

$$V_3 = VDD - V_{Tn} + V_{Tp} - \sqrt{I_b}\left(\sqrt{\frac{2L_5}{K_pW_5}}+\sqrt{\frac{2L_6}{K_nW_6}}\right) \quad (4.2)$$

Hence, using the equation 4.2 in equation 4.1 and considering the transistors M3 and M4 to be identical, I$_q$ may be related to I$_b$ as:

$$\frac{I_q}{I_b} = \frac{\left(\sqrt{\frac{2L_5}{K_pW_5}} + \sqrt{\frac{2L_6}{K_nW_6}}\right)^2}{\left(\sqrt{\frac{2L_1}{K_pW_1}} + \sqrt{\frac{2L_2}{K_nW_2}}\right)^2} = \frac{\left(\frac{1}{\sqrt{\beta_5}} + \frac{1}{\sqrt{\beta_6}}\right)^2}{\left(\frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}\right)^2} \quad (4.3)$$

The output class AB stage in the Figure 4.1 was replaced with the class AB scheme in Figure B.1 as shown in Figure B.2, to verify the class AB operation.

The circuit in Figure B.2 was simulated, for the DC response of the output terminal. The bias currents Ibias1 and Ibias2 was set to 30μA and 10μA respectively. The power supplies VDD and VSS was set as 2.5 V and -2.5V respectively. The simulated results are shown in Figure B.3. From the simulated results we observe the output voltage swing to be linear between -1.7V and 2.4V. The quiescent currents through the output (M15 and M16) stage was observed to be very high (more than 300 μA), which corresponds to class A operation. The requirement is for the quiescent currents to be low, roughly in the range of 5μA to 10μA. Analysing the cause for high currents through M15 and M16, a high current through the current source M10 and the level shifting transistor M9 were also observed (Figure B.3). The problem could be solved by decreasing the W/L for M9, increasing its Vgs. However to significantly reduce the current, this would require very long thin transistors. Hence an alternative scheme of adding two diode connected transistors M8 and M7 to the earlier discussed class AB scheme in Figure B.1, as shown in Figure B.5.

Analysing the modified circuit in exactly the same manner as the earlier class AB stage analysis, we can derive an expression relating I$_q$ and I$_b$ equivalent to equations 4.1 and 4.2 (analysis included in Appendix B). Copying equation B17 here:

$$\sqrt{2I_q}\left(\frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}\right) = VDD - V_{Tn0} - V_{T3} - V_{T8} + V_{Tp0} - \sqrt{2I_1}\left(\frac{1}{\sqrt{\beta_3}} + \frac{1}{\sqrt{\beta_8}}\right) \quad (4.4)$$

Considering M4 now,

$$\sqrt{2I_1} = \sqrt{\beta_4}(V_3 - V_{Tn0}) \quad (4.5)$$

The term V$_3$ can be found from:

$$V_3 = VDD - V_{T8} - V_{T7} + V_{Tp0} - \sqrt{2I_b}\left(\frac{1}{\sqrt{\beta_5}} + \frac{1}{\sqrt{\beta_6}} + \frac{1}{\sqrt{\beta_7}}\right) \quad (4.6)$$

Therefore eliminating I_1 in equation 4.4,

$$\begin{aligned}\sqrt{2I_q}\left(\frac{1}{\sqrt{\beta_1}}+\frac{1}{\sqrt{\beta_2}}\right) &= (VDD - V_{Tn0} - V_{T3} - V_{T8} + V_{Tp0}) \\ &-\sqrt{\beta_4}\left(\frac{1}{\sqrt{\beta_3}}+\frac{1}{\sqrt{\beta_8}}\right)(VDD - V_{T6} - V_{T7} + V_{Tp0} - V_{Tn0}) \\ &+\sqrt{\beta_4}\left(\frac{1}{\sqrt{\beta_3}}+\frac{1}{\sqrt{\beta_8}}\right)\left(\frac{1}{\sqrt{\beta_5}}+\frac{1}{\sqrt{\beta_6}}+\frac{1}{\sqrt{\beta_7}}\right)\sqrt{2I_b} \end{aligned} \quad (4.7)$$

From the above equation, we observe that if the first two terms cancel out, then the relation between I_q and I_b depends primarily upon the geometry of the transistors. If for simplicity, the effect of substrate to source potential changing the threshold voltages is ignored here, this requires that $\sqrt{\beta_4}\left(\frac{1}{\sqrt{\beta_3}}+\frac{1}{\sqrt{\beta_8}}\right)=1$.

This alternate class AB stage was applied to the DDA circuit as shown in Figure B.4. A diode-connected NMOS transistor M19 is added in between the source and drain terminals of M9 and M10 respectively, which introduces a voltage drop at the gate-source voltage of M9. The length of the current source transistor M10 was made sufficiently larger than M9, for the M10 drain current not more than 10μA. The currents again should not be too small (in the range of nA) to minimise cross-over distortion. Also note the addition of an extra transistor M20 along with the existing M17 and M18 in the compensation scheme.

The addition of diode connected transistors can be analysed with respect to the balancing of the total number of gate-to-source voltages in the class AB stage. This can be explained with the following analysis. For the original circuit, assuming that all the transistors are operating in the saturation region, from Figure B.2:

$$V_{SG_{15}} + V_{GS_{16}} + V_{GS_9} = Vdd - Vss \quad (4.8)$$

$$V_{SG_{17}} + V_{GS_{18}} + V_{GS_{10}} = Vdd - Vss \quad (4.9)$$

After the addition of M19 diode-connected transistor as in Figure B.4, equation 4.8 becomes:

$$V_{SG_{15}} + V_{GS_{16}} + V_{GS_9} + V_{GS_{19}} = Vdd - Vss \quad (4.10)$$

The above change introduces an imbalance in the total number of gate-to-source voltages in the class AB stage (formed by the transistors M9, M10, M15, M16, M17, M18 and M19), while comparing equation 4.9 and equation 4.10. Hence in order to balance the overall number of gate-to-source voltages of the class AB output stage, an additional transistor M20 (NMOS) is added to the compensation chain of transistors M17 and M18. So the equation 4.9 can be now written as:

$$V_{SG_{17}} + V_{GS_{18}} + V_{GS_{20}} + V_{GS_{10}} = Vdd - Vss \quad (4.11)$$

Hence we observe that equations 4.10 and 4.11 are balanced with respect to the total number of gate-to-source voltages. Thus making the quiescent current less sensitive to changes in Vdd - Vss or threshold voltage respectively.

The modified DDA circuit in Figure B.4 was simulated, for the DC response with Ibias1=10μA, Ibias2=30μA, VDD=2.5V and VSS=-2.5V. The simulated results are shown in Figure B.6. From the simulated results we can observe the controlled and low-quiescent currents (10 μA) and a maintained linear output voltage swing (-1.7V to 2.4V).

4.1.3 Stability and Frequency Compensation of the DDA

Following the verification of the DC Response and the class AB operation of the DDA circuit, the DDA was tested for stability configured as a unity gain buffer. Transient analysis showed oscillations. The reason for the oscillations is that the phase shift exceeds 180^o at unity loop gain frequency. Compensation schemes to ensure stability of opamp based circuits are discussed in most textbooks for analogue circuit design [6, 49, 39]. These methods are important as they make the opamp suitable for particular configurations and ensure stability for opamp based applications. Among these methods Miller compensation [6, 49] is attractive as the low-frequency pole can be established with a single moderate value capacitor. Another technique, Nested-Miller compensation [104], employs a number of Miller compensation capacitors between amplifier stages and the output of the opamp, which tend to occupy more chip area. Miller compensation moves the dominant pole to a lower frequency, however a feed-forward path through the compensation capacitor results in an additional zero located in the right half plane. This zero introduces extra negative phase shift/phase lag in the transfer function of the opamp. A method to avoid the effects caused by the right hand plane zero, is the placement of a resistor in series with the Miller capacitor. This is often called the lead-compensation method [49] and, typically, a MOS transistor (operating in the triode region) is used instead of a resistor. The circuit of the DDA was modified to include compensation and is shown in Figure B.7. The compensated DDA was again simulated for the transient response. The DDA was observed to be stable for both the simulated transient response, without any external load resistance (Figure B.8) and with a load resistance of 1MΩ. A stable transient response was observed for the compensated DDA with a 10pF capacitive load too as shown in Figure B.9.

4.1.4 AC Response of the DDA

The compensated DDA was simulated for the open-loop gain and the phase margin. An AC gain of 84dB and a phase margin of $+82^o$ was observed for the open-loop gain and phase response of the DDA respectively, is shown in Figure B.10. An open-loop gain of

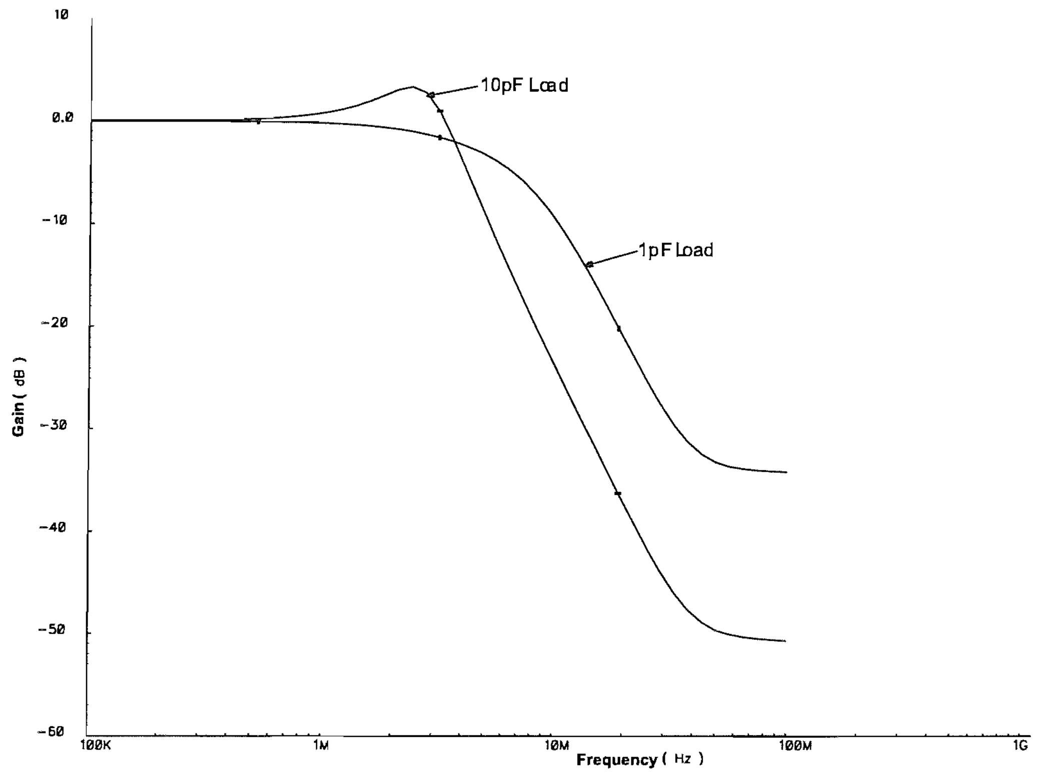


FIGURE 4.3: Closed-loop frequency response of the DDA as unity gain amplifier driving 1pF and 10pF capacitive load

82dB and phase margin of $+42^o$ was observed for the DDA driving a 10pF (typical probe capacitance) load capacitance, is shown in Figure B.11. The reduced phase margin here is due to the high capacitive load of 10pF. The closed loop AC response of the DDA configured as a non-inverting unity gain amplifier was also simulated for driving a 1pF and a 10pF capacitive load respectively. The observed closed-loop frequency response of the unity gain amplifier is shown in Figure 4.3. We see that a high bandwidth of around 10MHz is achieved with a 1pF load capacitance. For a 10pF load the unity gain starts to roll off after peaking at approximately 2.5MHz.

4.1.5 Input performance characteristics

The definition of the input characteristics is complicated for the DDA circuit. The typical characterisation of the input as in a opamp using differential and common-mode voltages cannot be used. The reason is the extra input ports in the DDA when compared to an opamp. The basic property of the ideal DDA with negative feedback can be expressed as:

$$Vpp - Vpn = Vnp - Vnn \tag{4.12}$$

From the above equation, we observe that the main characteristic of the DDA is its ability to balance the two differential voltages (Vpp - Vpn) and (Vnp - Vnn). But this equality is dependent upon the common-mode voltages at the two pairs of ports. Let us represent the four input ports of the DDA as p1 (Vpp), p2 (Vpn), n1 (Vnp) and n2 (Vnn). Then parameters to be considered while measuring the input performance of the DDA can be summarised as:

- differential voltages at the ports p1-p2 and n1-n2 respectively.
- common-mode voltages of ports p1-p2 and n1-n2 respectively.
- differential common-mode voltages between the ports p1-p2 and n1-n2 (when the common-mode voltages of p1-p2 and n1-n2 are not equal).
- differential difference voltage (DDV) between the ports p1-p2 and n1-n2 respectively (difference between differential voltages at the port p1-p2 and port n1-n2).

Hence the characterisation of the DDA input performance involves a complex analysis on the effect of the above parameters and the interaction between them. It is indeed very difficult to summarise the concerned parameters as specific voltage ranges. In fact, such an analysis could be considered as future work in this project. A similar analysis and measured input characteristics of the DDA is presented in [80], by E. Säkinger. Hence, due to the involved complexities, only the following input characteristics of the DDA were simulated, namely:

- the CMRR in dB, considering only one half of the DDA
- the DDVR (Differential Difference Voltage Range)

4.1.5.1 CMRR

The compensated DDA was simulated for both the differential voltage gain and the common-mode voltage gain. The measurement was done considering only the upper-half of the compensated DDA. The CMRR in dB was calculated by subtracting the common-mode voltage gain from the differential voltage gain, and was observed to be 97dB, as shown in Figure B.12.

The compensated DDA circuit in Figure B.7 was modified for a better input common-mode voltage swing (which will affect the output voltage swing in unity gain configuration). The standard cascode current mirrors below the differential pairs, were replaced by a wide complaint current mirror [50], as shown in Figure B.13. Observe the addition of the transistor M22, with a W/L ratio equal to one-quarter of the identical W/L ratios of M5, M6, M11, M12, M13 and M14. This modification improved the output voltage swing (approximately 2.4V and -2) at the cost of decreasing the CMRR gain to 78dB, which is a tolerable trade-off.

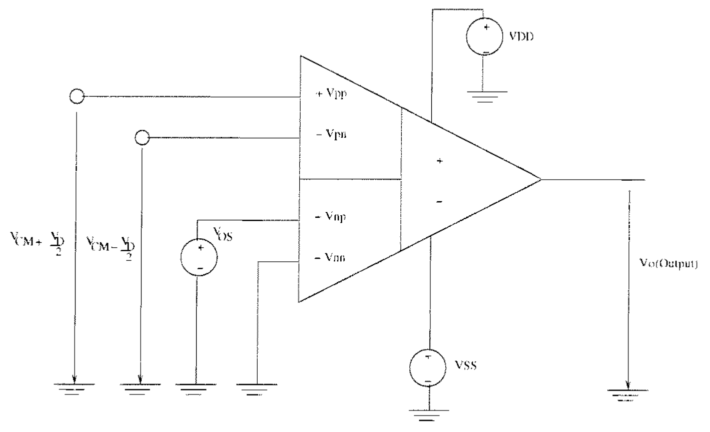


FIGURE 4.4: Offset cancellation of the DDA configured as an opamp

4.1.5.2 DDVR

The DDVR is the range over which the DDA is able to balance the two differential input voltages (Vpp-Vpn) and (Vnp-Vnn). This determines the maximum input signal voltage range for some configurations (eg:- inverting unity gain buffer). The simulated DDVR range of the DDA configured as an inverting unity gain buffer (Figure B.14) shows a linearity between -1V and +1V, as shown in Figure B.15.

4.1.6 Input Offset Voltage

The input offset voltage of the DDA is another characteristic of great importance, as it does affect the output response of the target configuration. This is a serious problem here, because the input offset voltage can be large with the high gain of the DDA. The offset voltage can be cancelled when using the DDA as an opamp, with Vpp and Vpn inputs used in the same way as the single input differential stage of the opamp. To compensate for the DC offset a compensation voltage is applied to the Vnp or Vnn inputs of the DDA. The idea illustrated is shown in Figure 4.4. From the figure we observe that the output voltage is:

$$V_{out} \approx A_d\{[V_D - (V_{OS} + V_{off})] + \frac{1}{CMRR}V_{CM}\} \tag{4.13}$$

, where V$_{out}$ is the output voltage, V$_D$ is the differential input voltage, V$_{CM}$ is the common-mode input voltage, V$_{OS}$ is the input offset compensation voltage and V$_{off}$ is the offset voltage. From the equation we observe that V$_{OS}$ can compensate V$_{off}$. This method of compensating the offset voltage is not possible always when the DDA is targeting other applications. Considering the other possibilities for offset compensation, it was found that the addition of a third differential pair to the DDA can solve the problem. As long as this third differential stage is a low-sensitive pair that induces very small currents, it will not affect the circuit behaviour of the DDA. The modified DDA with this offset compensation scheme is shown in Figure B.16.

From the Figure B.16, we see that the offset compensation scheme is introduced into the DDA. The transistors M23 and M24 form the third differential pair, with a tail current from M26 controlled by the switch transistor M25 (S1 input). The gate of M23 is connected to ground, which the gate of M24 may be connected to ground or the input offset using the switch inputs S2 and S3. By keeping the tail current for this pair small (-400nA) and using small W/L ($5\mu/10\mu$), the required offset control voltage is 10mV per 0.02mV of offset.

Another offset compensation technique for the DDA that uses a digital successive approximation algorithm is proposed in [51]. This technique of an automated offset compensation is an attractive feature, but not feasible here due to the requirement of other blocks for the implementation of the successive approximation algorithm (M2M current DAC (Digital-to-Analogue Converter), comparator and SAR (Successive Approximation Register)), which will consume considerable area of the chip.

4.1.7 Slew Rate and Settling Time

The slew rate of the DDA determines the maximum rate at which the output changes when input signals are large. The DDA was simulated in unity-gain configuration with an output load capacitance of 10pF. The unity-gain configuration can be considered as the ideal configuration for worst-case measurements on stability and slew rate. This is because of the largest feedback in the unity-gain configuration leading to larger values of loop gain.

The simulation results (Figure B.17) show that the DDA is not slew rate limited and it has a settling time response of the order 0.52μs. The large overshoot during the rise of the signal is because of the large load capacitance in the simulation setup. The step response for the unity-gain DDA amplifier driving a 1pF capacitive load shows no overshoot (Figure B.18) and a settling time of 0.15μs. In general the DDA, unlike an opamp, does not have slew rate limitation, because the low transconductance input stage of the DDA does not overload for a wide range of input voltage. The internal Miller compensation capacitance provides good stability with no overshoot and ringing, provided the load capacitance is small.

4.1.8 The final design of continuous-time DDA

The DDA design shown in Figure B.16 can be considered as a good design with regard to the performance characteristics discussed in the earlier sub-sections (4.1.1-4.1.6) and *the achieved simulated DDA performance parameters for a power supply of +2.5V and -2.5V is summarised below:*

- AC open loop gain 84dB and a phase margin of $+82^o$
- Gain bandwidth product of 1.2MHz
- CMRR 78dB
- Output voltage swing of -2V to 2.4V
- DDVR -1V to +1V
- Offset voltage of -4mV
- Not slew rate limited
- settling time response of 0.52μs with 10pF capacitive load.

The DDA also meets two of the decisions on design strategies, discussed earlier (section 3.7, Chapter 3), which were :

- *to adopt a continuous-time approach for the design of the CAB as the first decision.*
- *to incorporate the facility for supporting fully-differential input signal ports and only a single-ended output signal port as the third decision.*

The second decision to incorporate both voltage-mode and current-mode of operation for the CAB, could only be partially achieved, because the DDA supports only the voltage-mode of operation. Hence the possibility of achieving current-mode of operation along with the existing voltage-mode from the DDA, was analysed. The following section includes the discussion on this effort and also the changes made in the design of the DDA to support this additional feature.

4.2 Voltage/Current mode DDA

While analysing and investigating the possibility of achieving a current-mode of operation from the DDA, a paper [7] proposes a fully differential second-generation current conveyor(CCII) achieved by incorporating a current sensing technique into the DDA. This idea of a fully differential second-generation current conveyor presented in [7] is shown in Figure 4.5. The DDA described has two differential input ports (V_{pp}, V_{pn}, V_{np}, V_{nn}), as in the DDA designed earlier (Figure B.16), but has differential output port (V_{op} and V_{on}) rather than the single-ended output V_{out} in Figure B.16. A current

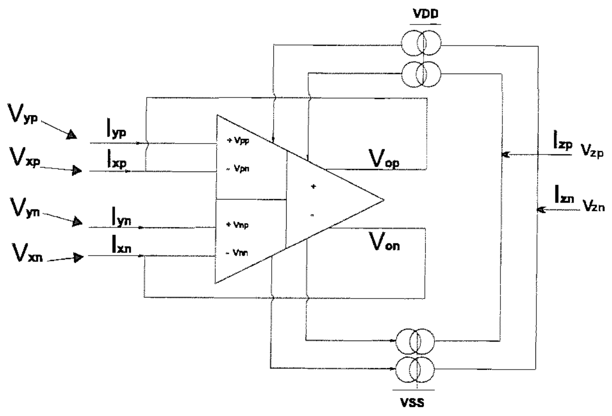


FIGURE 4.5: Fully Differential Second-generation current conveyor

sensed DDA is obtained by copying the currents through the two output terminals (V_{op} and V_{on}) to two additional current terminals (I_{zp} and I_{zn}) using current mirrors. The current sensed DDA can be configured as a CCII by feeding back the two output voltage terminals (V_{op} and V_{on}) to two of the inputs (V_{pn} and V_{nn}), respectively. The circuit behaves as a current-conveyor with differential input voltages, input currents and output currents respectively, i.e. V_{dx} = V_{xp} - V_{xn}, V_{dy} = V_{yp} - V_{yn}, I_{dx} = I_{xp} - I_{xn} and I_{dz} = I_{zp} - I_{zn}. Thus the feedback results in V_{dy}=V_{dx} and I_{dz} becomes equal to I_{dx}. Also, the feedback action develops the required low input impedance at the X terminals. The disadvantages of using fully differential structures were discussed in section 3.3, Chapter 3.

A similar current sensing technique may be applied to the single-ended DDA. The output stage may be modified as in Figure 4.6. M1 and M2 are the output transistors of the original DDA, but the currents in these transistors are replicated using transistors M3 and M4 to provide a current output, I_{out}. If feedback is applied from V_{out} directly to the appropriate input terminal of the DDA (eg: V_{pn}) then the circuit will behave as a current conveyor with V_{pp} acting as the Y terminal, V_{out} as the X terminal and the extra current output as the Z terminal.

This circuit suffers from the limitation of finite output resistance of transistors M1-M4, which means the current at the Z terminal will not accurately match that at the Y terminal. This may be overcome by the use of cascode transistors as in Figure 4.7. The final design incorporated two cascoded output stages to provide two duplicate current outputs with a high output impedance, which reflect the current at the voltage output terminal (Figure B.19). The feature of two current ports Iout1 and Iout2 was provided to facilitate a better current fan-out capability. The transistor dimensions for the designed DDA/Current conveyor with bias currents Ibias1=20μA, Ibias2=10μA and Ibias3=20μA are shown in Figure B.19.

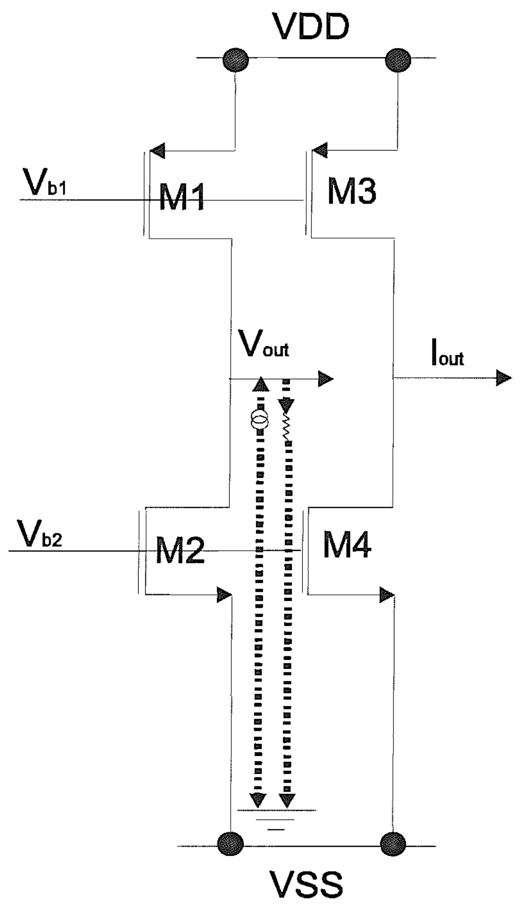


FIGURE 4.6: Current sensing in the output stage of the DDA

4.2.1 DC Analysis

The DDA/Current conveyor (DDACCII) in Figure B.19 was simulated for the output current characteristics of the Z terminals (Iout1 and Iout2) by sweeping the current fed into the X terminal (-100μA to +100μA) of the current terminal. The simulation setup is shown in Figure 4.8(a). The Y input (V_{pn}) was maintained at a DC voltage of zero volts. Observe the linearity of the output currents in Iout1 and Iout2 in the range of -60μA to 50μA, as shown in Figure 4.8(b). The simulation setup shown in Figure 4.8(a) was used to perform a DC sweep on Vin (-2.5V to 2.5V)fed into the Y terminal along with the DC sweep of the current Idc (-100μA to +100μA) fed into the X terminal. The simulation results are shown in Figure 4.9. From the DC Response of Vout (X terminal) we can observe an imaginary box (shown in dotted lines) that defines the appropriate ranges of operating input voltages and the corresponding linear currents available from the DDA/Current conveyor.

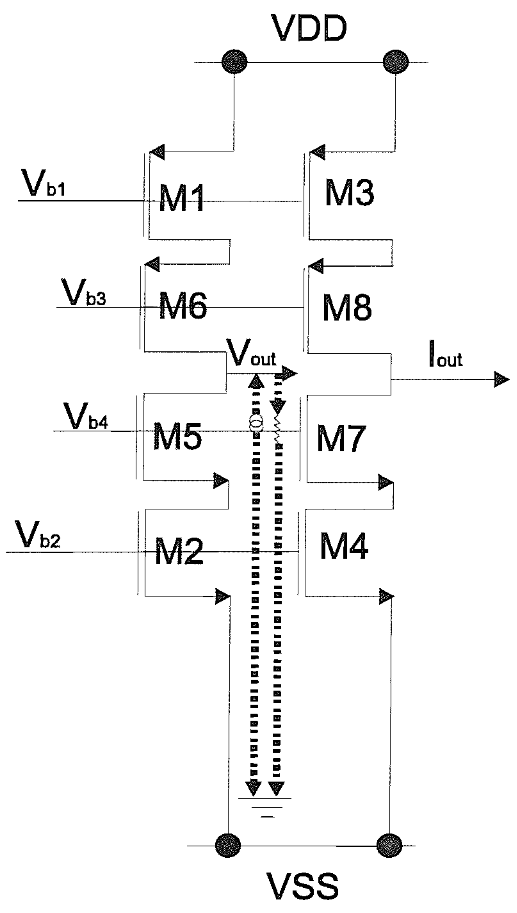
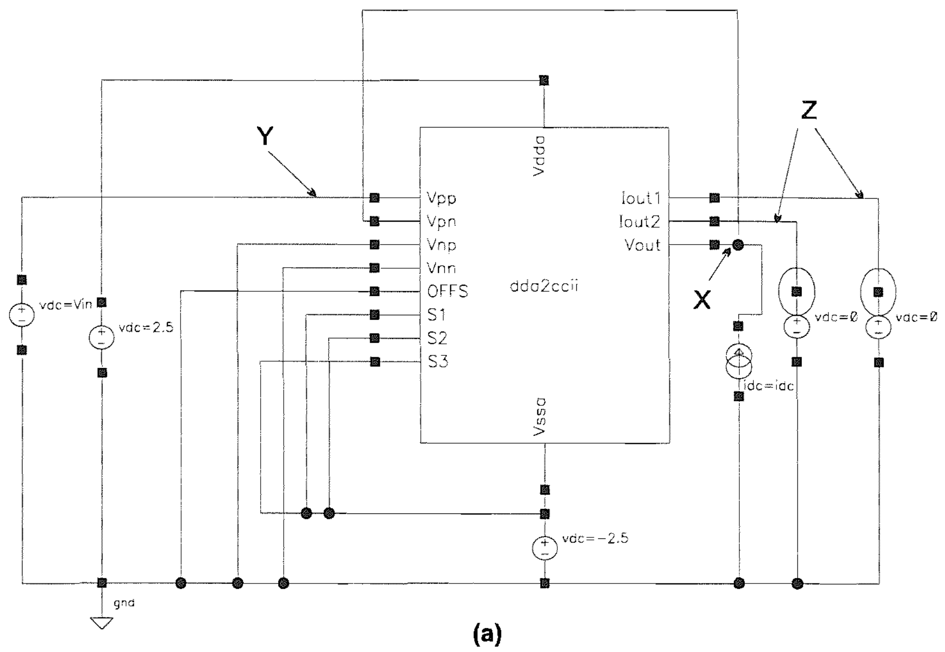


FIGURE 4.7: Cascoded current sensing in the output stage of the DDA

4.2.2 AC Analysis

The small signal transconductance of the DDACCII was simulated using an AC analysis. The circuit was configured as a CCII with V_{np} and V_{nn} connected to ground and V_{pn} connected to V_{out} (Figure 4.10(a)). A variable bias current was injected at the X terminal (V_{out}) and an AC voltage applied to the Y input (V_{pp}). The X input was decoupled to ground using a 10mF capacitor. The 10mF capacitor is a common simulation trick provided for close-loop DC feedback (to bias the amplifier) and to ensure AC open-loop gain could be measured to lower frequencies. The transconductance was greatest with a bias current of O, decreasing as the bias was increased to +/-25μA and +/-50μA (Figure 4.10(b)). The maximum transconductance corresponds to 17.8mA/V. The transconductance shows a -3dB frequency of about 40kHz.

Removing the 10mF capacitor and replacing it with a 10kΩ resistor, the transconductance should equal the conductance of the resistor (0.1mA/V) and this is found to be the case. The bandwidth is increased to 1MHz. Increasing the resistance to 100kΩ, both the voltage at V_{out} and the transconductance show peaks in the region of 2MHz (Figure 4.11). This is partly due to the under-compensation of the amplifier with unity



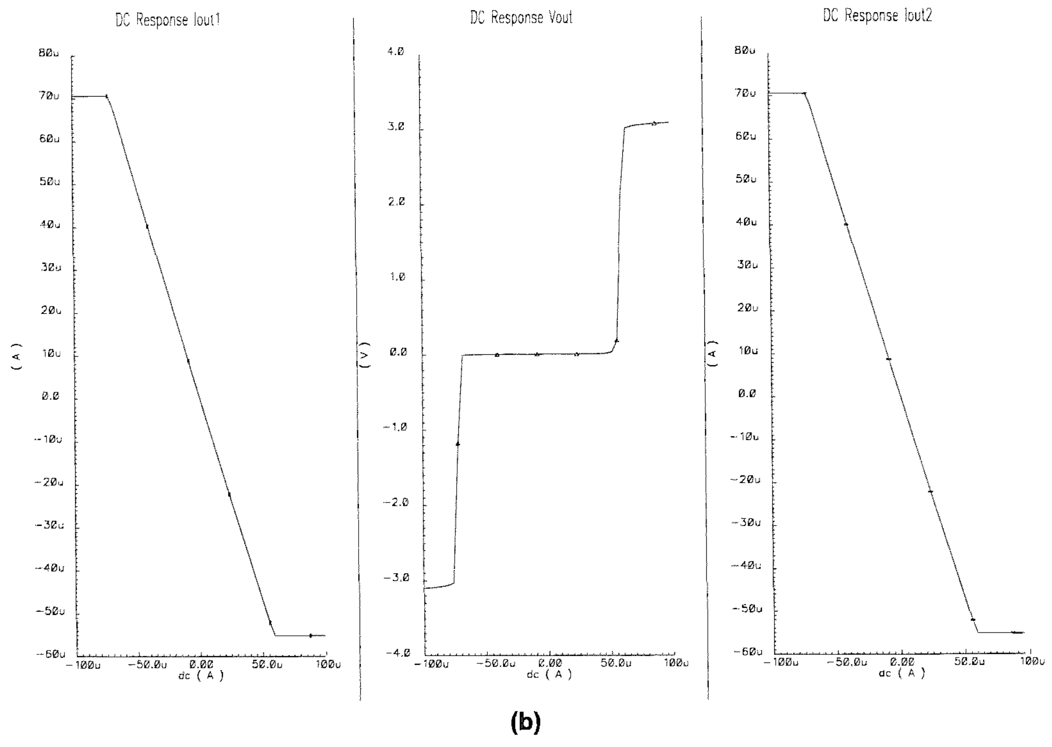


FIGURE 4.8: (a)Simulation setup for the output current characteristics of the Z terminals(b)Output current characteristics of the Z terminals(Iout1 and Iout2) and output voltage characteristics of X terminal (Vout)

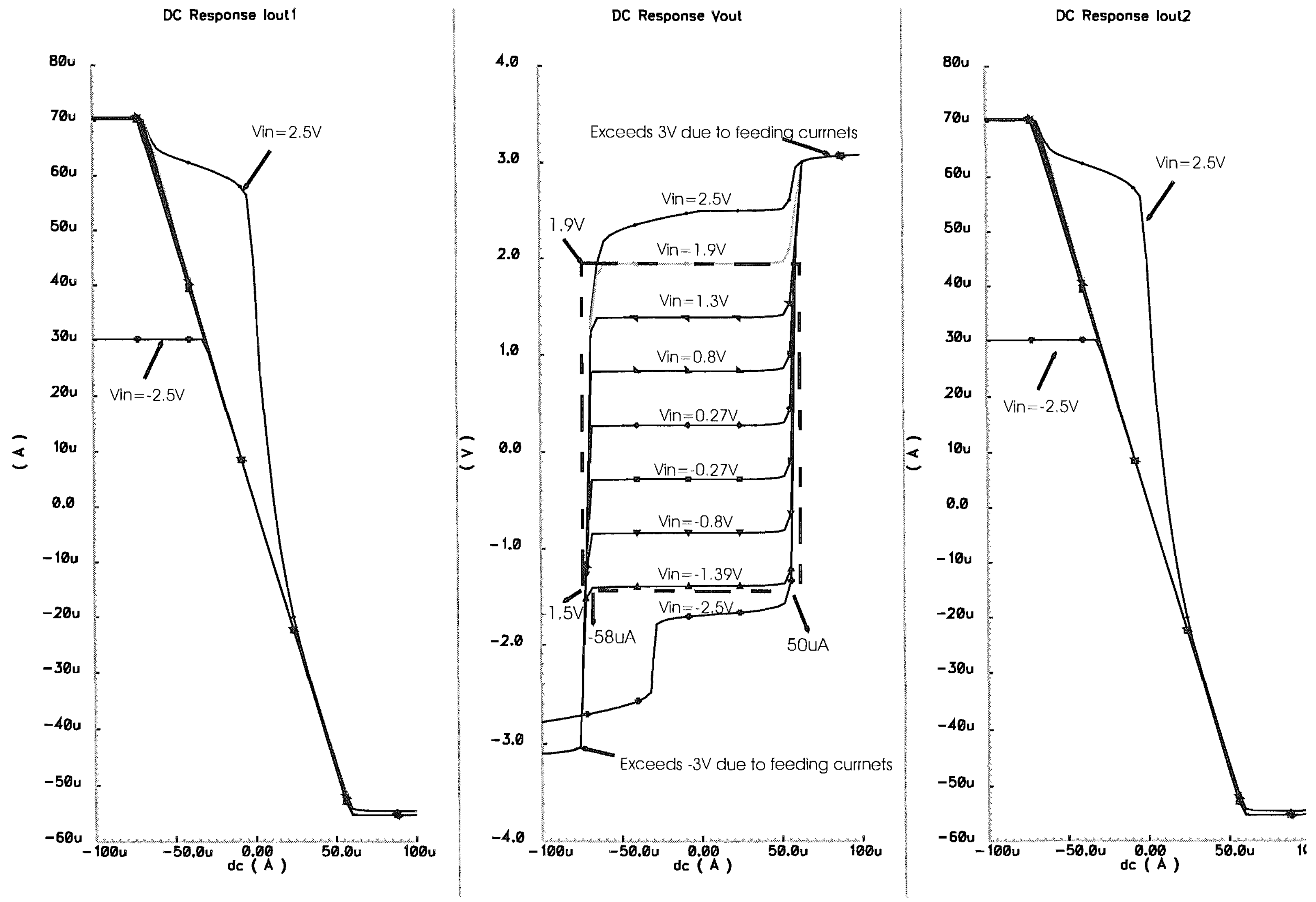
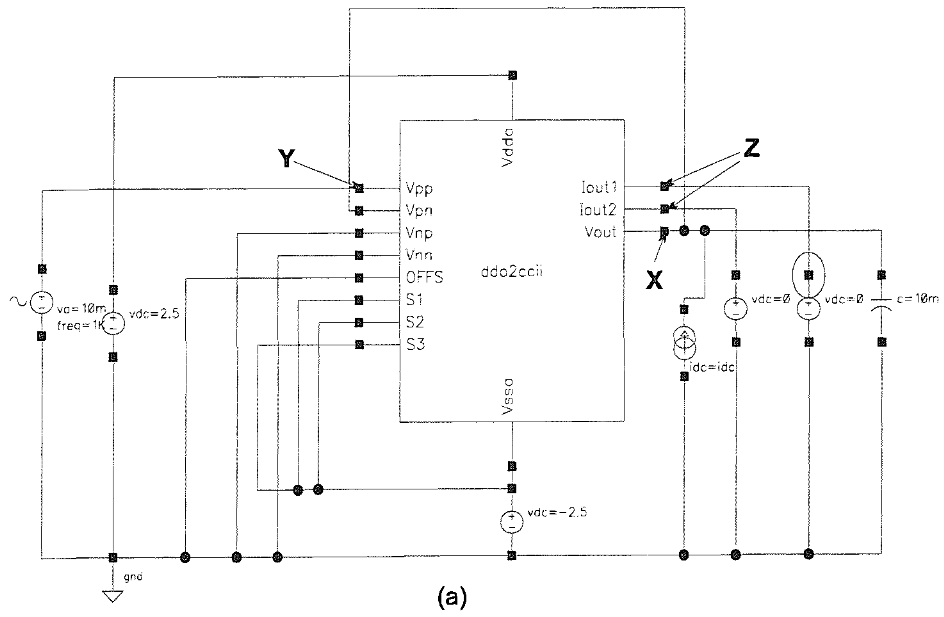


FIGURE 4.9: Output characteristics of the Z terminals (currents) and X terminal (voltage), after a DC sweep of Vin (-2.5V to 2.5V)



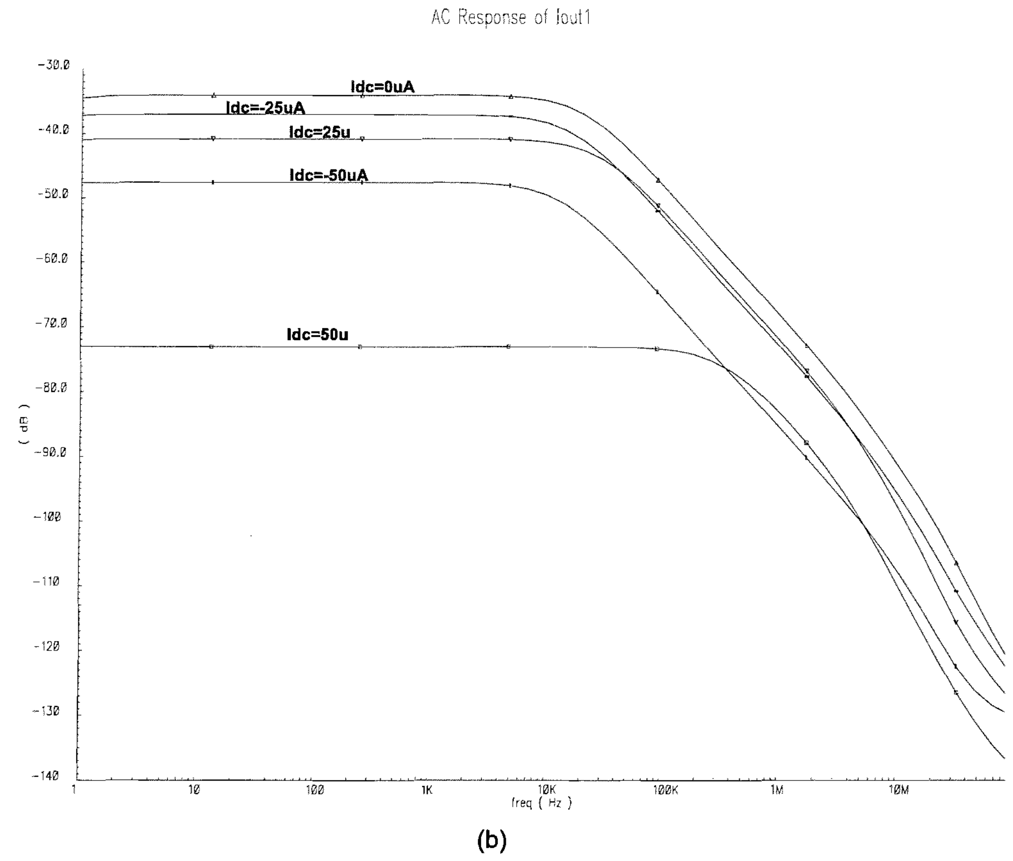
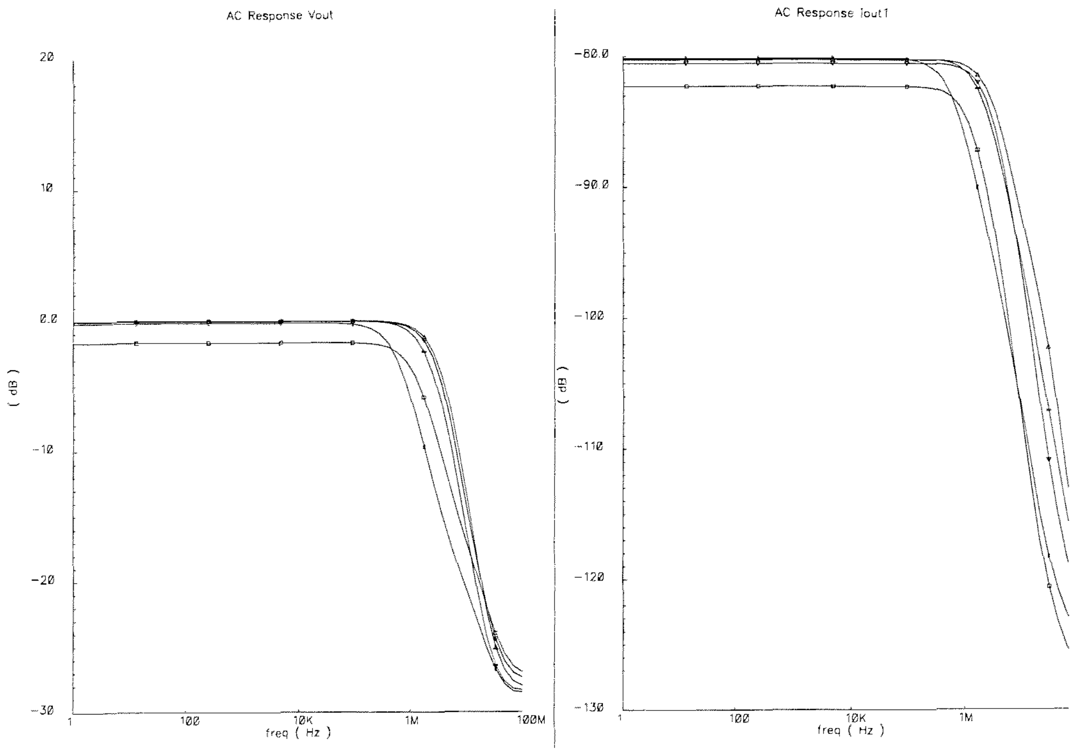
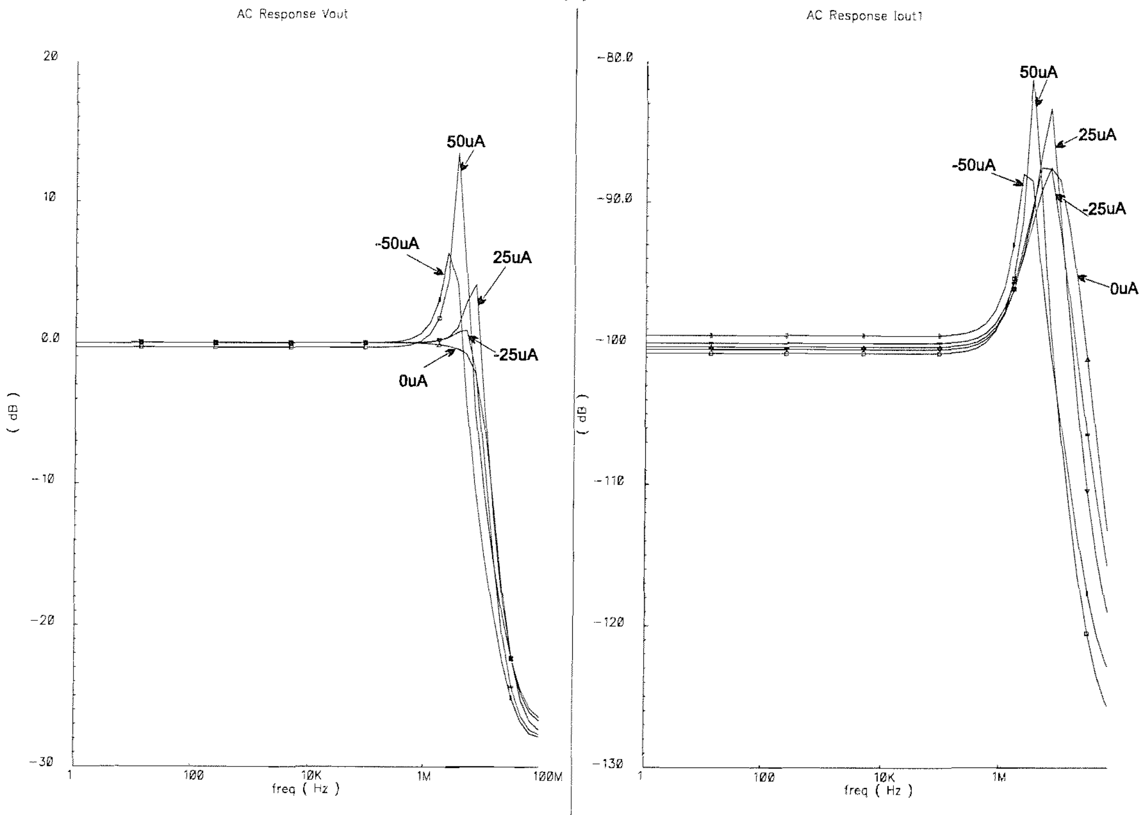


FIGURE 4.10: (a)Simulation setup for AC Analysis (b) Varying transconductance of the DDACCII with varying Idc



(a)



(b)

FIGURE 4.11: Frequency response with external resistance of (a)10KΩ (b)100KΩ

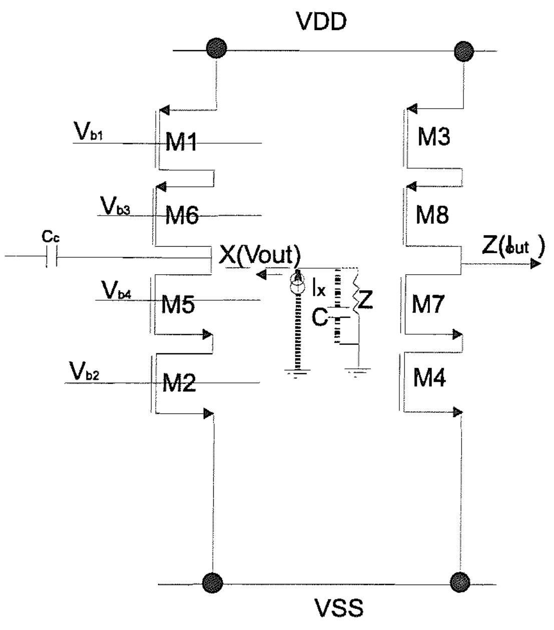


FIGURE 4.12: Currents through the cascode transistors of DDACCII block

gain feedback (giving the peak in the voltage transfer response) and partly that the current mirrored to the current output also includes current flowing in the compensation capacitor(C_c), shown in Figure 4.12. This latter problem could be overcome using a buffer transistor in the compensation circuit, but if the X terminal is brought out through a pad to an external resistance, a similar effect will be produced by the pad capacitance, which at 5pF is substantially larger than C_c. Effectively the pad capacitance places an upper limit on the value of the external resistance at the X terminal, if peaks of this sort are to be avoided.

4.2.3 Transient Analysis

The DDACCII was simulated for the transient response. The applied input to the Y input was a pulse of 1V, 10μs pulse width, 20μs in period, t_d=t_r=t_f=500ps. An external resistance "res"Ω was connected to the X terminal and was swept during the transient analysis from 1kΩ to 1MΩ. The input current Idc fed into the X (Vout) terminal was set to zero. The obtained transient response for the varying external resistances and Idc=0A is shown in Figure B.21.

We observe from the simulated results that the output voltage response follows the input voltage better gradually for higher values of resistances. The earlier simulation setup was simulated again with varying resistances (10kΩ-1MΩ) and by feeding a constant Idc current of 30μA. The transient responses are shown in Figure B.22. The "ringing" behaviour was observed for external resistances greater than 100K and this instability again increases with increasing external resistance. But it was observed that these oscillations do not sustain throughout the signal level, but reduce quickly even for a high resistance of up to 1MΩ. Thus the transient analysis proves the stability of the DDACCII configured as a current conveyor.

4.3 Switching in DDACCII

Having designed the DDACCII which could operate in voltage-mode and current-mode, the flexibility of switching between possible configurations was analysed. The possible configurations for the single DDACCII block in voltage mode and current mode were considered again as shown in Figure 4.14 and Figure 4.15. Comparing the required connections for each configuration in both the figures, we observe that some of the connections are quite common in almost every configuration. On the basis of this comparison, an appropriate number of switches was assigned to the DDACCII block as shown in Figure 4.13.

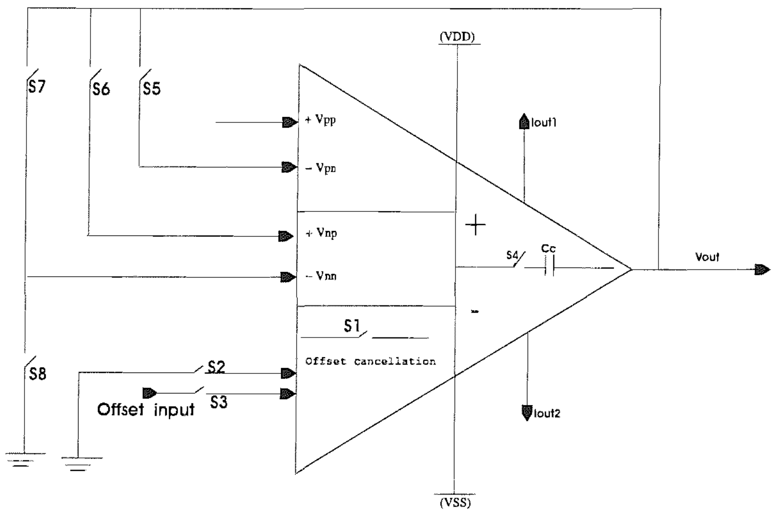


FIGURE 4.13: DDACCII with the possible switches

The ON resistances of all the MOSFET switches (S1-S8) sized with equal width of 10μm and minimum length 0.6μm was observed to be in the range of 800Ω to 900Ω with varying input common mode voltages. This ON resistance was found to be negligible compared to the high output impedance contributed by the cascoded output stages of the DDACCII. The total capacitance (10fF-50fF) and the time constant (RC value) contributed by these switches again is negligible. Since the switches are always switched statically and not dynamically, the non-ideal switch effects as in switched-capacitor circuits (nonlinear capacitance on each side of the switch, channel charge injection and capacitive coupling from the logic signal on each side of the switch) can be ignored. The possibility of using dynamic switching external to the DDACCII block at the input and the output ports was considered and is discussed in section 4.3.1.

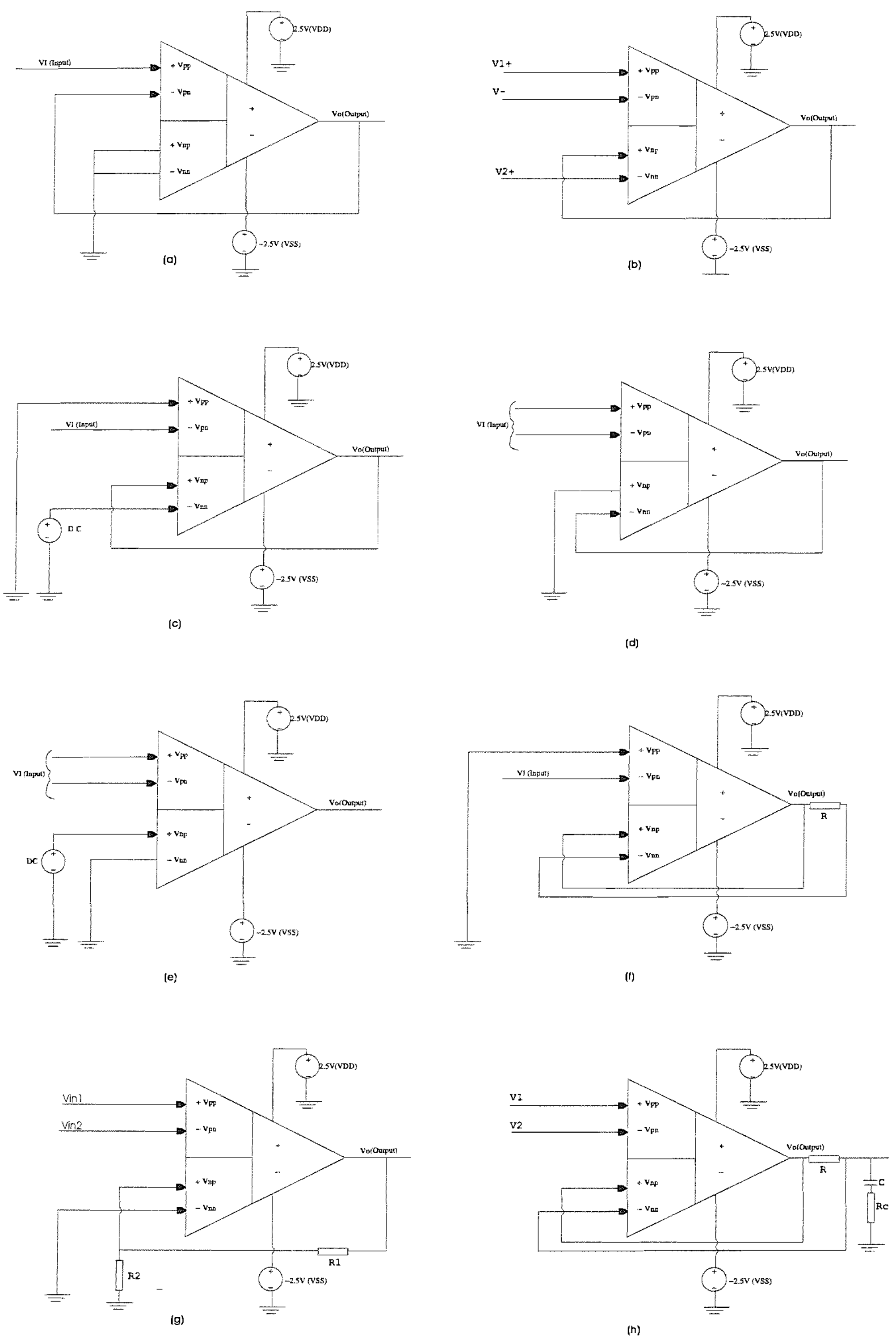


FIGURE 4.14: (a)Unity-gain buffer, (b)Adder/Subtractor, (c)Level shifter, (d)Comparator, (e)Opamp with Offset cancellation, (f) Voltage Controlled current source, (g) Instrumentation amplifier, (h)Integrator with phase lag compensation

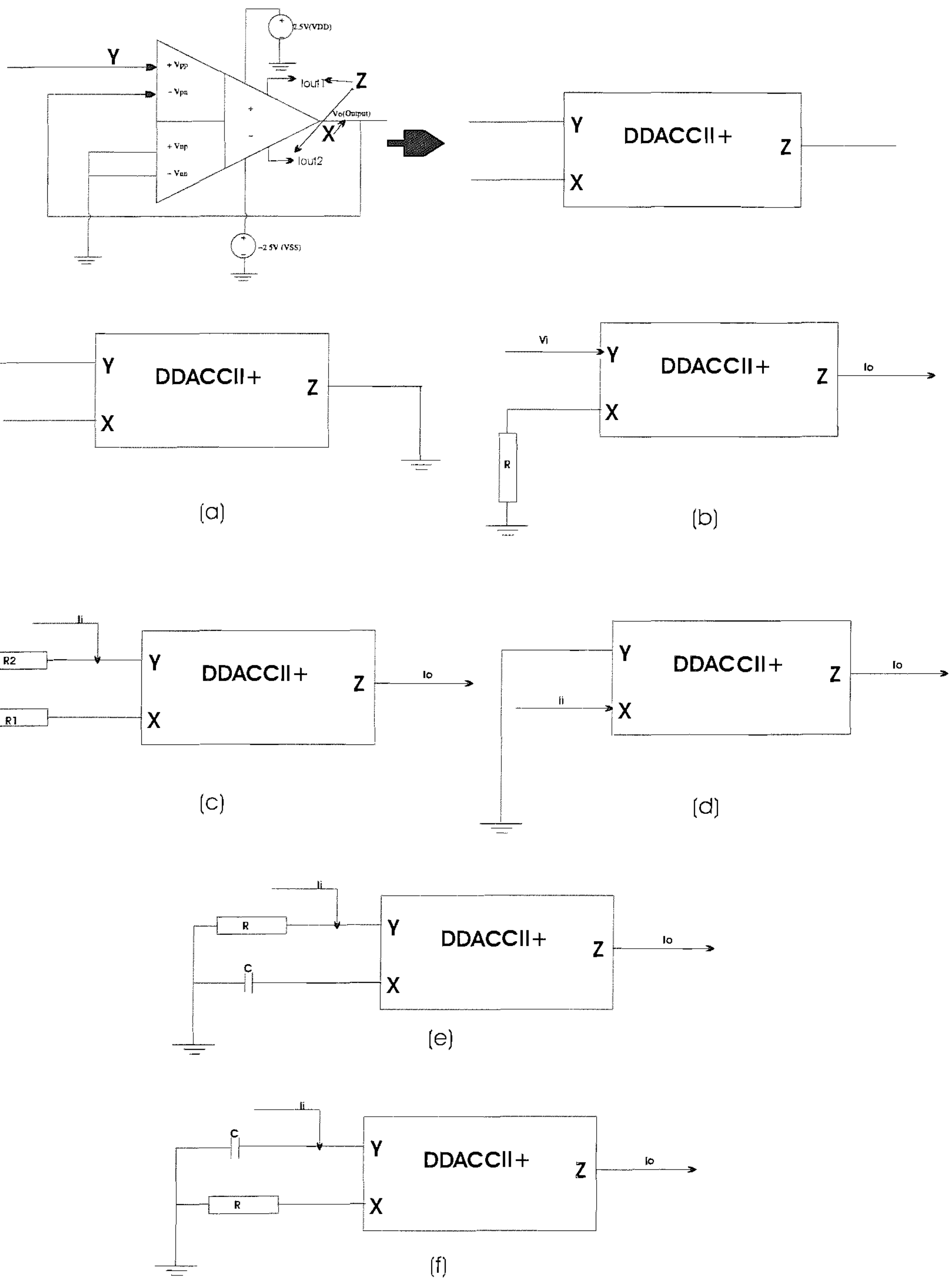


FIGURE 4.15: (a)Voltage controlled voltage source, (b)Voltage controlled current source, (c)Current Amplifier, (d)Current controlled current source, (e)Current differentiator, (f)Current integrator

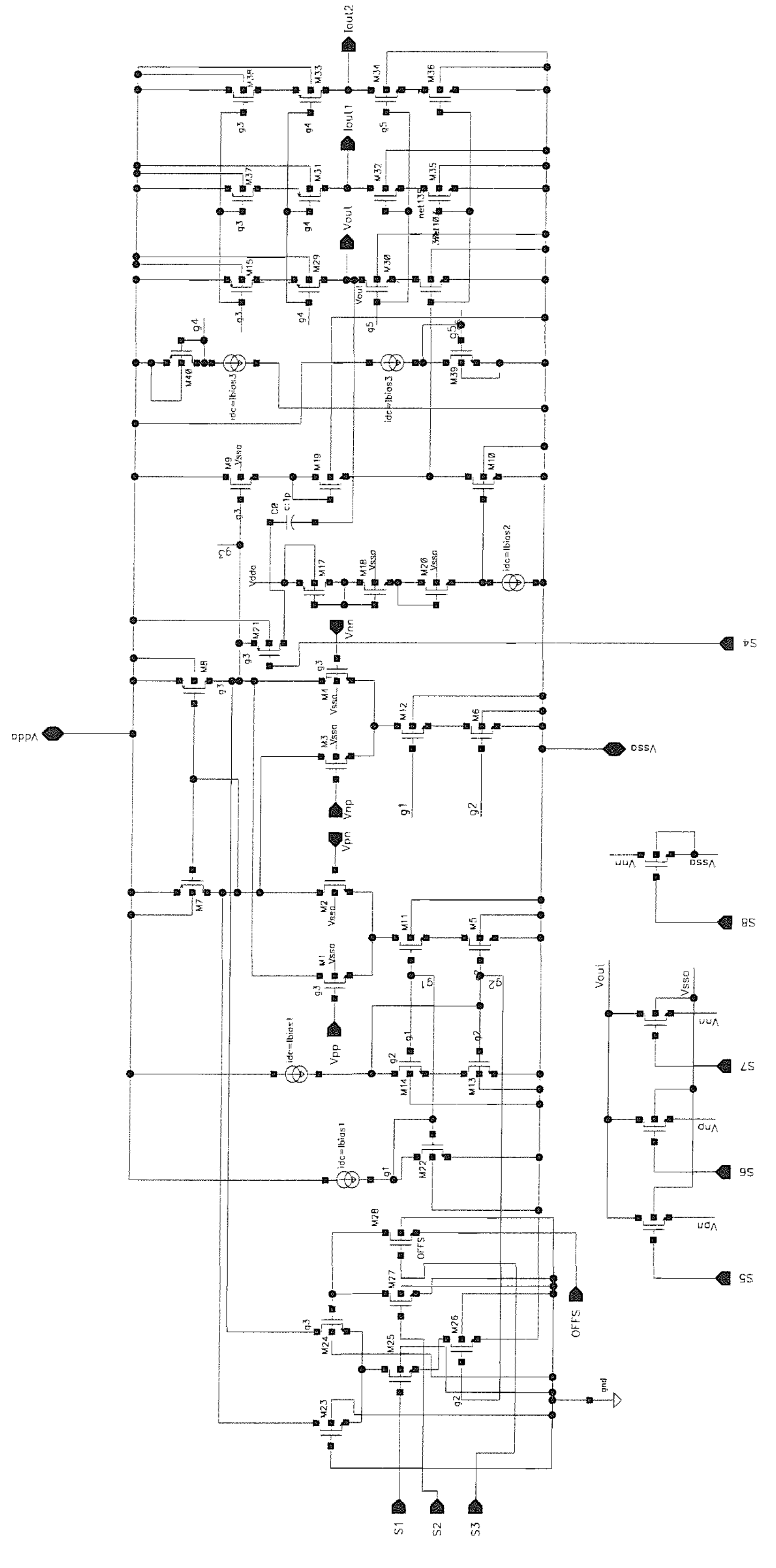


FIGURE 4.16: DDACCII with the incorporated switches

4.3.1 DDACCII and Dynamic Switching

The DDACCII CAB was simulated for applications that uses dynamic switching. The dynamic switching was implemented by a double pole double throw switch (dpdtsw) as shown in Figure 4.17. The dpdtsw switches between polarities formed by the connected terminals p1-p4 and p2-p3, based on the activation of the enable signal "enb". According to the signal level of the enable signal, the switched inputs are available as two outputs op1 and op2. The switches are implemented using transmission gates formed by PMOS and NMOS transistors of equal transistor dimensions (W=10μm, L=0.6μm) as shown in Figure 4.18. This switch was used to set the inputs (Vpp, Vpn, Vnp, Vnn) of the DDACCII from input sources in dynamic fashion. The switch was also used at the outputs (Vout, I1, I2) of the DDACCII, to dynamically switch between any of the outputs, and hence provide a controlled set of inputs to the next stage in the circuitry. The use of this switch in implementing a current-mode Schmitt trigger and a VCO, with the DDACCII CAB is described in the Chapter 6.

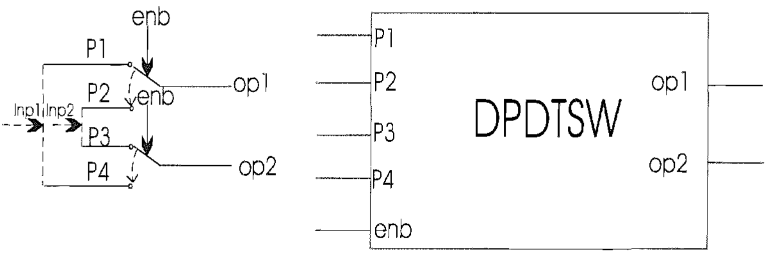


FIGURE 4.17: Double pole Double throw switch

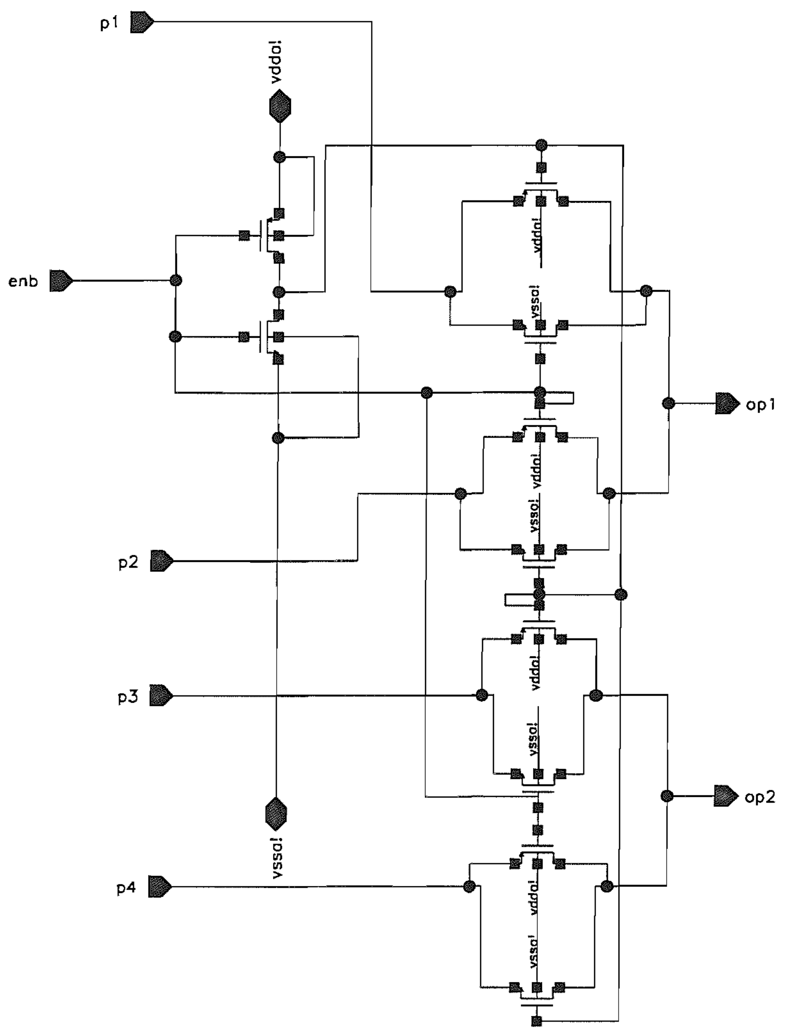


FIGURE 4.18: Double pole Double throw switch schematic

4.4 Noise Analysis

The noise simulation of the designed DDACCII circuit is important as it gives an estimate of the inherent noise. In the case of the DDACCII, the noise response will change according to the circuit configuration. Apart from inherent noise, the interference noise may also be quite significant with respect to the overall noise response. As the DDACCII CAB will be a part of the FPAA array, the interference noise will be occurring primarily due to the interconnects, which depend on the layout of the FPAA chip. There could also be other unwanted interaction of the circuit with the outside world, for example, the power supply noise on ground wires. As an initial estimate, the noise simulation of the DDACCII block configured in voltage mode and current mode was performed. This is compared with the measured noise from the chip in Chapter 6. The equivalent input voltage noise at the input of the DDACCII CAB is expected to be quite large because of the low transconductance of the input transistors. This is evident from the equivalent input voltage noise of a MOS transistor, approximated for low and moderate frequencies as, $Vi_n^2 = 8kT\frac{1}{3G_m} + \frac{KF}{WLC_{ox}f}$, where f is frequency, G_m is the small-signal transconductance from gate-to-channel, T is the temperature, k is the Boltzmann's constant, KF is the flicker noise coefficient, W is the width of transistor, L is the gate length of the transistor.

The equivalent input and output noise of the DDACCII was simulated at the Vpp input of the DDACCII, with a feedback connection from Vout to Vnp (hardwired) and the other two inputs (Vpn and Vnn) grounded. The noise response in Figure B.23(a) shows an equivalent input noise of $74\frac{nV}{\sqrt{Hz}}$ at a frequency of around 380kHz . The equivalent output noise measured at Vout was same as the equivalent input noise as expected for the unity gain buffer configuration and is shown in Figure B.23(a). The noise analysis was also done on the DDACCII configured as a current conveyor, with the hardwired feedback connection from Vout to Vpn and all other inputs grounded. An AC current was fed into the Vout terminal (analogous to the X terminal in current conveyor). The simulated equivalent input noise at the X terminal was around $3.4\frac{pA}{\sqrt{Hz}}$ at a frequency of around 380kHz. The simulated equivalent output noise at the Z terminal (Iout1 and Iout2) was also similar to the measured equivalent input noise. The noise response of the DDACCII as a current conveyor is shown in Figure B.23(b). The harmonic distortion of the circuit is also a significant performance measurement. It is quite difficult to come with a final value for the total harmonic distortion of the DDACCII, as it will depend upon the circuit configuration. As a worst case scenario, the DDACCII was configured in a closed-loop gain of 100 as shown in Figure B.24. The estimated harmonic distortion as a function of the amplitude of the input signal level was found be less than 0.1 percentage in spite of the very high gain. The observed fundamental, third and fifth harmonics and their corresponding amplitudes are also shown in Figure B.24.

4.5 FPAA Architecture

Taking account of the various tested and verified applications of the DDACCII CAB, we consider the possibility of building larger analogue systems using the DDACCII CAB. This requires the DDACCII CABs to be arranged in the form of an array, which can be configured to provide sufficient flexibility to achieve the target applications. This flexibility in programming or configuring an array of CABs depends on the arrangement pattern, also known as the interconnect architecture. The interconnect architecture defines

- the amount of interconnectivity between the CABs
- the resources involved in achieving the desired interconnectivity between the CABs
- the different type of connections between CABs (direct, general purpose, long line) with minimum number of switches in the signal path

The complexities involved in choosing an appropriate interconnect architecture may be analysed better by considering the various interconnect architectures proposed for the FPGA. This might help in relating and exploring the interconnect architecture issues and other related strategies for the target FPAA architecture. FPGA can be programmed to specify the function of the logic blocks and their interconnections. The switch blocks in the FPGA can be configured to connect wire segments to form larger networks. The logical blocks are then connected to these networks using programmable connection blocks. Additional IO blocks are also provided in the FPGA to provide the interface between the input/output pins and internal signal lines. The additional facility of programmability in the FPGA makes the architecture more complex than that of a conventional gate array. Switches are generally used to configure the interconnection paths. There could also be other possible configurable connection devices namely SRAM-driven pass transistors, EPROM-driven pass transistors, or anti-fuses. Considering a connection path in the FPGA, a signal may pass through many connection devices, which posses higher resistances and capacitances than a regular metal wire. Hence a significant time delay is associated with the signal path. It is quite often that this delay exceeds the delay of the logic blocks. This results in the degradation in the performance of the FPGA. The high area consumption of these pass devices for providing interconnections in the FPGA is also responsible for its lower density. To improve these shortcomings of low density and speed, papers [99, 55] propose a hierarchical interconnection structure based FPGA, called the HFPGA. The HFPGA has an architecture as shown in Figure 4.19(a) and (b), where the logic blocks are connected to form "clusters". A graphical representation for the same architecture is also shown. It has been proved in [55], that the total number of switches in an HFPGA is less than the conventional FPGAs under the requirement of equivalent routability.

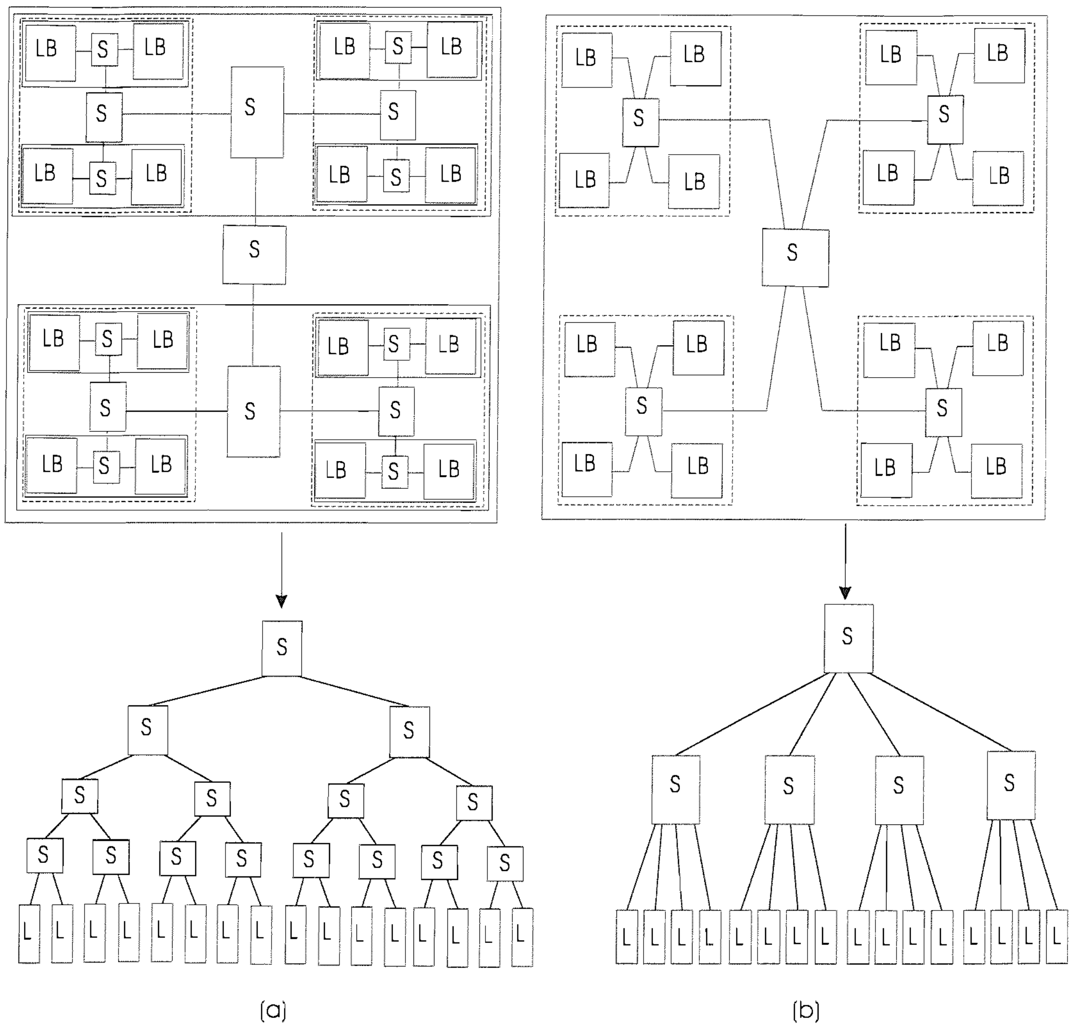


FIGURE 4.19: (a) and (b) showing two possible hierarchical interconnection architectures for an HFPGA

Different FPAA architectures have been proposed in [10, 75, 60, 36, 61, 64]. These FPAA architectures and the technology behind each architecture have been discussed earlier in Chapter 2. We observe that a majority of them follow an array-based architecture, which consists of rows and columns of CABs. Additional resources are provided for both horizontal and vertical routing between these rows and columns in the array. The other proposed architecture in [75] following a current-mode based signal processing, aims to minimise the number of programmable devices in the signal path. But all these architectures are dependent on the nature of the CAB and the target applications the FPAA has to meet. There are still other issues to be clarified as to how certain constraints can affect the FPAA architecture and the choice of the CAB. The following sub-sections include a brief discussion on the possible strategies which are crucial in choosing the target FPAA architecture.

4.5.1 Limiting/Maximising Routing Resources

The strategy for defining the available routing resources in the FPAA affects the maximum routability available between CABs. This is also related to the amount of flexibility that can be exploited from a CAB. Maximising the allowable routing between CABs will improve the circuit coverage or routability of the FPAA. But the optimum solution would be to achieve maximum routability with limited number of resources. Hence the dependency of the interconnect architecture on the following constraints would be of particular interest: (a)the availability of a limited number of wiring resources, (b)the availability of a limited number of switches and its effect on the allowable connections, (c)the availability of limited routing alternatives due to the above two constraints, (d)the availability of input and output ports in the CAB to exploit maximum flexibility and (e)the availability of limited number of I/O blocks in the periphery of the chip and limited sharing by the array of CAB I/O signals.

4.5.2 Layout of the FPAA

The final layout of the target FPAA depends on the interconnect architecture. It is crucial to analyse the complexities involved in implementing the layout, due to the strategy of choosing either very limited or comprehensive routing resources (I/O blocks, local/global switches between CABs) of the FPAA interconnect architecture. Hence the following can be considered: (a)analysing the need for special layout techniques for interconnect architectures (minimise cross talk), internal capacitors/resistors, (b)analysing the effect of restrictions in the particular technology (number of metal layers available, process parameters), (c)analysing the possible methods to reduce the total layout area of the FPAA and hence the cost for fabrication.

4.5.3 FPAA Routing

Efficient routing plays a very important role in reducing the parasitics introduced in the interconnects. The most significant interconnect parasitics are: (a)resistance of interconnection track, (b)stray capacitances of tracks, (c)cross coupling capacitances between parallel and crossing tracks respectively and (d)series resistance and parasitic capacitance introduced by switches in the programmable architecture.

In an effort to reduce the above parasitics, certain constraints are introduced in the routing to avoid the overall degradation in the analogue behaviour, namely: (a)minimising wire lengths, (b)reducing the number of parallel and crossing wires between different wires and (c)limiting the number of switches in the signal path.

Another consideration under FPAA routing is the compatibility of the architecture to existing routing algorithms for an efficient FPAA router keeping the performance degradation minimal. The fact that existing routers for FPGAs cannot be used in their current form directly to solve the FPAA routing problem is discussed in [32]. Hence, the paper presents a new routing algorithm for array-based FPAAs (Motorola MPAA020 architecture [10]). The router is also demonstrated to be highly efficient, fast and keeps the degradation within acceptable limits.

The time constraints imposed by the desire to fabricate a prototype chip and test the CAB design meant that, a full consideration of these issues was not possible before the test chip was designed and fabricated. Hence, having completed the design of the DDACCII CAB, it was decided to implement an architecture based on the analysis of the available FPGA and FPAA architectures. Considering the advantage of the smaller number of switches for a HFPGA than an array based FPGA, and also the previous concentration on array based FPAA architectures, it was decided to investigate the feasibility of implementing a hierarchical interconnect architecture or HFPAA (Hierarchical Field Programmable Analogue Array).

The HFPAA would have architecture as shown in Figure 4.20, where the DDACCII CABs are connected to form "clusters". Each cluster contains four DDACCII CABs, which are connected again as two sub-clusters . Within each sub-clusters are two CABs connected by switch S1. The two sub-clusters are connected by the switch S2 in the next level hierarchy. These clusters can be used to build larger analogue systems, as we go up the hierarchy to S4. But there is an important difference incorporated in this hierarchical structure compared to that in HFPGAs [99, 55], with respect to the accessibility of the signals of interest anywhere in the hierarchy. From the tree based representation of HFPGAs, it is obvious that the signal at the lowest level of the hierarchy has to travel to the apex of the tree to be accessed or interact with another part of the tree. A similar scenario cannot be tolerated in an analogue circuit, which may require some external components and the signals might have to be accessed/interact externally. Thus it is necessary to facilitate the accessibility of signals at any level of the hierarchy to the I/O blocks of the target FPAA. It is also necessary to facilitate allowable interaction of signals between CABs at different level of the hierarchy without strictly following the tree based structure. Due to this property, a graph based representation of the architecture is adopted to describe the type of connection between CABs of the same level or higher levels in the hierarchy, which will be discussed later in section4.5.4.2.

This hierarchical architecture based on the positive results presented in [55] should be applicable conceptually to analogue circuit blocks. Consider an analogue circuit/CAB or a digital circuit/logic block as a black box. The wires/connections to and from this black box, can be classified as: (a)input connections, (b)output connections, (c)feedforward connections and (d)feedback connections. The key difference as regards routing between this analogue circuit and a digital circuit is the need for connections to external

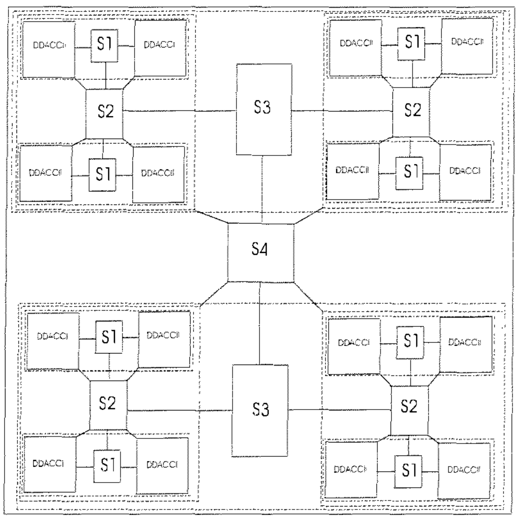


FIGURE 4.20: Hierarchical interconnection Architecture based FPAA

components to achieve the required behaviour. The other difference might be with respect to the number of signal wires. In digital circuits an application is not generally achieved from a single logic block. There is generally parallel processing distributed between a collection of logic blocks. In the case of an analogue circuit, a good range of applications can be achieved from a small number of flexible blocks like the DDACCII CAB. Hence the complexity with respect to the number of switch connections is less for an array of analogue blocks than for an array of logic blocks. Also the analogue blocks require a less dense array compared to the larger cluster of logic blocks required to achieve an extensive range of applications.

4.5.4 Design of the FPAA Architecture

Deciding upon implementing a hierarchical based architecture for the FPAA, the possible design of a cluster was analysed based on the idea shown in Figure 4.20. Considering the four CABs in each cluster and the switches(S1 and S2) connecting the four CABs, it is important to define the required and allowed routability in the cluster. This analysis will define the design of the switches S1 and S2 in the cluster. Before this analysis, the DDACCII CAB was modified to support configurability of the internal switches and also the additional circuitry to provide the required bias currents within the DDACCII CAB. The final design of the DDACCII CAB is discussed in the following section.

4.5.4.1 Final design of the DDACCII CAB

The DDACCII CAB was modified to enable the configuration of the static states of the eight internal switches(S1-S8) by an eight bit register as shown in Figure B.26 The standard DFC flip-flops (Appendix D.1)available in the AMS(Austria Micro Systems)

0.6μm CUP process library were used for the 8-bit register. This configuration register has an input(BIN) to stream in the configuration bits and also the clock input (CLK) for clocking the inputs serially through the shift register towards the right. The BOUT pin is to facilitate the shifting out of the bit stream, which is serially fed into the configuration register of the adjacent switching resource/CAB.

The other additional circuitry incorporated in to the DDACCII design is the addition of current mirrors that distribute the input bias current(10μA) available at "INP". The input bias current is mirrored accordingly to provide the required bias currents of 10μA and 20μA.

The other modification is the change in the terminology used for the most negative potential in the circuit. The 0.6μm CUP process does not support separate analogue and digital grounds. The CUP process is built on a P-type substrate with no way to isolate the bulk. Hence the analogue power rails in the design are between Vdda (5V) and gnd!(common for analogue and digital). The MIDV input is the mid-point (2.5V) input between the analogue power rail and is the reference voltage for analogue signals.

The DDACCII CAB with all the inputs and the outputs is shown in Figure B.25 and the schematic in Figure B.26. Note the accessibility of the ports provide on either side of the block. This is helpful to avoid routing wire connections going around the block and reduces the complexities in interconnect architecture to a certain extent. This is more significant and will be evident while implementing the layout of the target chip, which will be discussed later.

4.5.4.2 Design of the cluster

Considering Figure 4.20, we find that the cluster can be divided into two sub-clusters. Each sub-cluster is formed by two CABs and the switch box S1. Having the DDACCII CAB, we need to define the amount of required and allowable routability. This will define the required structure for the switch box. Considering Figure 4.21 and the required connections from CAB1 to CAB2 to achieve possible voltage/current mode applications, the required routability is listed in Table 4.2. These choices were made after considering the actual circuit configurations. The other connections were ignored so as to simplify the layout of the design.

CAB1	CAB2
VPP1 TO	IOUT21, VOUT2, VPP2
VOUT1 TO	VPP2, IOUT21
IOUT11 TO	VPP2, VPN2, IOUT21

TABLE 4.2: Considered connections from CAB1 to CAB2 through S1

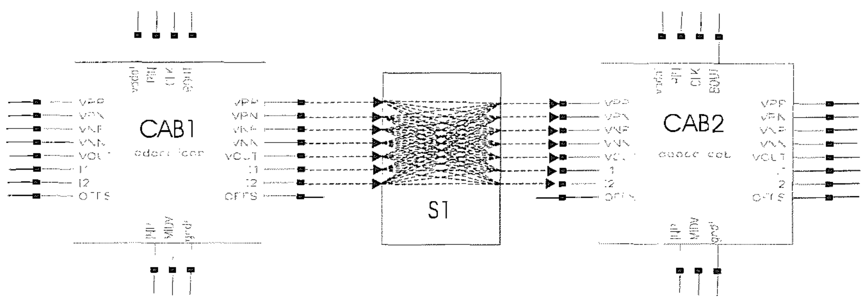


FIGURE 4.21: Sub-cluster and maximum routability from CAB1 to CAB2

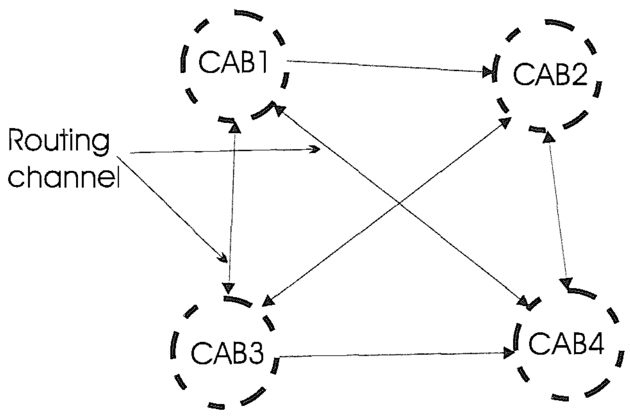


FIGURE 4.22: Graph representation of the cluster

Considering the next level within the cluster, the required allowed routability between the two sub-clusters through the switch box S2 was analysed. This analysis was done by following a graph based representation as shown in Figure 4.22 that defines the required allowable connections and the direction of signal flow. Observe the representation of the four CABs as nodes in the graph and the edges which symbolise the routing channel between the nodes. The directions and connections between the four nodes were decided on the basis of the possible target voltage/current mode applications achieved. This defines the required structure of the switch box S2 as listed in Table 4.3

The cluster was designed taking account of the required connections between the four CABS and the CAB positions as in Figure 4.22. This design containing the four DDACCII CABs and the switch boxes S1 and S2 as shown Figure B.27. Considering the routability between the DDACCII CABs at a sub-cluster level and at a cluster level, observe the structure of the switch boxes S1 and S2. The switch box S1 contains 8 switches formed by NMOS transistors with dimensions of W=10μm and L=0.6μm, this provides the connections listed in Table 4.2. The 8 switches can be configured by a 8-bit shift register in the switch box S1. Hence S1 has the input ports for input bit-stream, clock and the port for the bits to be serially sifted out to the right to the adjacent DDACCII CAB. The switch box S2 contains 24 switches formed by NMOS transistors dimensions of W=10μm and L=0.6μm, which satisfy the connections listed before in

Table 4.3. The 24 switches can be configured by a 24-bit shift register in the switch box S2. Hence S2 has the input ports for input bit-stream, clock and the port for the bits to be serially shifted out to the left. The bits are shifted left in S2 during the configuration of the cluster, serially fed into the DDACCII CAB (CAB2)towards the right of the sub-cluster at the bottom. The bottom sub-cluster composed of the DDACCII CABs (CAB2 and CAB4) and the switch box S1 is similar in structure to the sub-cluster formed between CAB1, CAB3 through switch box S1. The configuration bits for the bottom sub-cluster are fed to CAB2 from the bits serially shifted out left from the switch box S2. The configuration bit stream is shifted right through the configuration registers in CAB2 (8-bit), S1 (8-bit) and then CAB4 (8-bit).

All the MOSFET switches in the switch boxes(S1 and S2) were sized with equal width of 10μm and minimum length 0.6μm, so that they have an ON resistances in the range of 800Ω to 900Ω, with varying input common mode voltages. Also the allowable common mode range was such that a single MOSFET was sufficient and a transmission gate was not required. The ON resistance of these MOSFET switches was found to be negligible compared to the high output impedance contributed by the cascoded output stages of the DDACCII. The total capacitance (10fF-50fF) and the time constant (RC value) contributed by these switches again is negligible. Since the switches are always switched

Source CAB	Source CAB port	Destination CAB	Destination CAB port
CAB1	VOUT1	CAB4	VPP4
CAB1	VOUT1	CAB2	VPP2
CAB2	VOUT2	CAB3	VPP3
CAB2	VOUT2	CAB1	VPP1
CAB3	VOUT3	CAB4	VPP4
CAB3	VOUT3	CAB2	VPP2
CAB4	VOUT4	CAB1	VPP1
CAB4	VOUT4	CAB3	VPP3
CAB1	VPP1	CAB2	I21
CAB1	VPP1	CAB4	I41
CAB2	VPP2	CAB1	I11
CAB2	VPP2	CAB3	I31
CAB3	VPP3	CAB4	I41
CAB3	VPP3	CAB2	I21
CAB4	VPP4	CAB1	I11
CAB4	VPP4	CAB3	I31
CAB1	VOUT1	CAB2	I21
CAB1	VOUT1	CAB4	I41
CAB2	VOUT2	CAB1	I11
CAB2	VOUT2	CAB3	I31
CAB3	VOUT3	CAB4	I41
CAB3	VOUT3	CAB2	I21
CAB4	VOUT4	CAB3	I31
CAB4	VOUT4	CAB1	I11

TABLE 4.3: Considered connections between CAB1 to CAB4 through S2

statically and not dynamically, the non-ideal switch effects as in switched-capacitor circuits (nonlinear capacitance on each side of the switch, channel charge injection and capacitive coupling from the logic signal on each side of the switch) can be ignored.

A plan of fabricating a target FPAA chip as a part of this research, containing a single cluster was also considered for the cluster structure shown in Figure B.27. The next section includes the plan of the chip, which forms the basis for the layout of the chip discussed in the next Chapter 5.

4.6 Plan of the target FPAA chip

As a part of the research on programmable analogue circuits, a demonstration on the validity of the developed and discussed concepts above, by fabricating a FPAA chip was important. Hence this FPAA chip is required for the following:-

- to measure the performance of the developed DDACCII CAB.
- to measure the performance of the CABs at a sub-cluster, cluster or any level of the hierarchy.
- to explore the complexities in implementing the layout of the chip.
- to estimate the area consumption for implementing a certain array of CABs in the chip and the corresponding dependence on the total cost for the chip.
- to explore any trade-offs which could have been refined in the design of the CAB, architecture of the chip etc.
- to help in future research (routing and placement algorithms, study of interconnect strategies etc, based on the developed FPAA architecture).

Considering the available resources and funds in the research group, it was decided to implement only one cluster as in Figure B.27, for the target FPAA chip in 5V, 0.6μm CUP (3 Metal Layer) process. The implementation of the cluster should be enough to demonstrate the flexibility of the CAB and the target applications which can be achieved from the cluster of CABs. The chip would also include two double pole double throw switches (dpdtsw) to demonstrate the possibilities of achieving dynamic switching, discussed earlier in section 4.3.2. It was also decided to include an extra DDACCII CAB, separate from the cluster and routing structure, which could be used for measuring performance characteristics (frequency, AC, DC, Noise analysis).

The total number of pins for the target FPAA chip is also an important decision to be considered for the plan of the chip. The total number of required pins for the target chip was analysed with respect to the following:-

- the pins for analogue and digital power supplies
- the input and output pins for each DDACCII CAB in a cluster
- the input and output pins for a DPDTSW
- the input and output pins for the test DDACCII CAB alone
- the miscellaneous global inputs (CLK, configuration bit stream input BIN, input bias current) for the chip

After carefully considering the above factors it is obvious that the total number of required pins can be reduced to a large extent. *One of the possibility is to keep the common inputs for all the blocks inside the chip as a global input pin/pad.* For example the analogue/digital power supplies could be made global. Hence the pins for Vdda, Vdd, analogue/digital ground, MIDV could be shared. Similarly the common practise for providing the required bias currents in the chip is through a single input bias current input, which is again mirrored to multiple bias currents of required range using current mirror techniques. The same could be implemented for providing the appropriate bias currents through a single input pin/pad INP. The miscellaneous inputs like BIN, CLK, BOUT for the registers used in the chip, as the registers can be serially cascaded for the bits to serially shifted during the configuration period. The completion of the configuration can be found by probing the BOUT connected at the end of the shift register, inspecting for the first test bit fed into the BIN.

The other possibility for reducing the total number of pins in the chip, is to consider the possibilities of providing the inputs/output pins/pads on a resource sharing basis. For example let us consider the total number of inputs/outputs required in a cluster of four CABs. We observe that the individual inputs/outputs required for each CAB can be allocated from a set of common input/output pads using I/O blocks, that can multiplex signals to the CABs requiring inputs or multiplex signals out of the CABs to the output pads. *But considering the additional area required for these additional I/O blocks, multiplexers and the affordable total cost for fabricating the test chip, this possibility of resource sharing was not pursued. Hence it was decided to provide dedicated input/output pads required for all the CABs in the cluster and also for the test CAB in the chip.*

The design of the target FPAA chip at a block level and the transistor level is shown in Figure B.28 and Figure D.10 respectively. We observe that the total number of pins for the target chip is 67. But in the actual layout the analogue and digital ground pads are separate and hence the total number of pads at the periphery of the chip will be 68. The pins are listed in Table D.1 along with the pin types.

The designed FPAA chip was simulated for various configurations. The observed simulated transient responses at the outputs of all the CABs was as expected. For example, the designed schematic of the FPAA chip was validated using the test setup shown in Figure B.29. The FPAA chip was configured using an appropriate bit stream for the following:-

- DDACCII CAB1 as an inverter, fed with a 0.05V pulse input.
- DDACCII CAB3 as voltage doubler that was fed with the inverted output from CAB1, through the switch box S1.

- DDACCII CAB2 as voltage doubler that was fed with the doubled voltage output from CAB3, through the switch box S2.
- DDACCII CAB4 as voltage doubler that was fed with the doubled voltage output from CAB3, through the switch box S1.
- the test DDACCII CAB at the bottom was hard wired as a voltage doubler again that was fed with the doubled voltage output from CAB4.

The observed transient responses of the chip at outputs of all the CABs are shown in Figure B.30(a), along with the input waveform fed into CAB1. Figure B.30(b) shows the waveforms confirming the input bit-stream, the clock signals and the activation of the power supplies after the configuration of the registers. Observe that the clock period had a period of $10\mu s$ and a total of $740\mu s$ to configure the 72-bit register (24-bit registers in a row of the cluster, so 24 x 3 rows = 72). The observed simulated transient responses at the outputs of all the CABs was as expected and are shown in Figure B.30(a).

Chapter 5

Layout of the FPAA Chip

The layout of the FPAA chip was the next major task to be implemented. This chapter deals with the step-by-step layout procedure and highlights important layout techniques used for implementing the layout of the FPAA chip submitted for fabrication in 0.6μm CMOS CUP process. The plan and design of the target FPAA chip discussed in Chapter 4 was used as a basis for planning the layout of the chip and also for verification through simulations, which will also be discussed in this chapter. Hence, this chapter covers the extensive work that was required to implement the layout of the chip and for simulation based verification.

5.1 Layout Plan

Having the design and the plan of the target FPAA chip, the layout was implemented by following a top-down planning approach. The planning of the layout was done with respect to the architecture planned for the chip discussed in section 4.5, Chapter 4 and also the design rules for the 0.6μm CMOS CUP process. This (AMS0.6μm CMOS CUP) process is a 4 metal layer process that facilitates an operating voltage of 5V. The layout was planned at a block level, so that the whole chip could be considered as a collection of blocks or "leaf cells". A rough estimate of the available area to implement the core of the FPAA, after placing the input/output pads at the periphery of the chip was made as shown in Figure C.1. This was done by roughly stacking the total number of pads (68), choosing appropriate pad types (refer Appendix D), as per the planned design of the FPAA chip. This also helped to plan the layout of the other items in the chip that consume larger space and hence are more difficult to change because of their larger size. The power supply routing, clock routing or any other group of signals that consume a lot of the routing area, which might be critical path signals were also considered. Hence, it is possible to visualise the positions of the individual blocks within the available space for the layout of the core of the FPAA chip, shown again in Figure C.1. The planning was followed by a bottom-up implementation of the layout of the leaf cells, building up the individual blocks and their interconnections, to complete the layout of the whole chip. But some important layout issues and layout techniques for planning and implementing the layout of the chip were considered and are discussed in the following subsections.

5.1.1 Power Supply

The power supply lines of the chip are important signals that needs to be considered to meet the high connectivity and current requirements connected to every block or transistor in the design. The objective was to match the required power supply connections with electro-migration requirements and resistance characteristics for all the circuitry on chip. However these large structures inevitably consume a considerable amount of the chip area. Hence, some power estimations can be done based on the earlier design of the chip, which can then be extrapolated for the target chip after reviewing the following:-

- the available metal layers in the process and their associated resistive and capacitance nature.
- the process parameters for all the vias and contacts.
- the need for any multiple power branches to provide separate power supply lines for isolating any blocks that introduce noise (high-speed blocks).
- the total number of power pads and their corresponding positions in the chip, to estimate the overall routability and resistance paths from the pads to the different blocks.

Considering the above factors and the process parameters for 0.6μm CMOS CUP process, the main power supply routing for the chip was done in METAL3 (M3) layer following a tree/root structure approach. In this approach, the power line starts as wide as possible and gradually when it is connected to the various blocks, the power supply lines become thinner, analogous to the roots of a tree. The planned wide analogue and digital power supply lines in the available area of the chip is shown in Figure C.1. Metal slots will be provided in the wide M3 structures, which is a layout technique discussed in section 5.1.6.1, of this chapter.

5.1.2 Clock tree

The clock tree is a very common clock implementation scheme, where a network of buffers are inserted into the clock signal path in such a way that the overall delay from the clock generator/pad to all destinations is minimised. Instead of optimising a single signal path, the path is broken into several paths which are strategically buffered to minimise the delay. The clock network resembles a tree, where the main clock signal branches throughout the chip through buffers and hence the clock signal made available for all the required blocks/leaf cells. Layout techniques similar to the "root" routing approach used in power routing can be used for implementing clock routing. The capacitance effects are more important in the case of a dynamic signal like the clock. Hence, the choice of the appropriate routing layer is made that reduces both the associated resistances and capacitances. The clock signal was planned to be split as three buffered branches one for each row of the cluster having 24-bit register (Two 8-bit registers of the two CABS + 8-bit register of S1 = 24-bit register of S2 = 24-bit register). The standard cell "CLKBUF" available in the AMS 0.6μm library was used as the clock buffer (Appendix D.2). The exact positions of the clock buffers could not be planned, until the position of the logic cells requiring the clock was fixed.

5.1.3 Routing Plan

Apart from planning the routing of the prime signals discussed above, it is also essential to determine the overall complexity of the routing to be implemented. It is possible to predict these complexities by assigning specific areas of the chip for routing only, also known as routing channels. It is also possible to predict the possibilities of bottle necks or any complex routing areas in the chip. These can be predicted by having a routing plan based on the following:-

- signal based estimates
  - estimate of the total number of signals required for every block
  - estimate of the signal source and destination
  - pad list and pad positions around the die
  - rough estimates of the block positions so that the signal routing can be sketched
  - estimating the need for any routing over the blocks, by defining the tolerable boundaries of the blocks and location of the routing channels.
- establishing routing direction for each available metallic layer on a channel-by-channel basis (discussed in the next section)
- estimating features and overheads to handle any future changes in the design was also built into the plan, by leaving spare lines and extra space in the available area for the layout of the core of the chip.

5.1.4 Routing Direction

Assigning a particular direction for every metallic layer available in the process is a good practice for implementing the layout of the chip, keeping in mind the variation of process parameters across the chip. This variation is important with respect to the routing layers and the associated resistances and coupling capacitances. With respect to these factors, it was decided to implement all major vertical routing outside the cells/blocks in METAL3. This is because from the AMS 0.6 μm process parameters METAL3 offers much lower resistance per square than METAL1. In addition, METAL3 is physically the highest layer, it suffers less from cross coupling effects. It was decided to use METAL1 for most of the vertical routing inside the cells, except for any longer vertical routing channels in METAL3. All the major horizontal routing inside and outside was to be implemented in METAL2 as strictly as possible.

5.1.5 Stack generation for matched transistors

Matching of devices is a significant issue in analogue and mixed signal design, which has to be considered very carefully to achieve high performance. It is quite typical of a CMOS process that the absolute parametric accuracies of transistors vary, while the parametric ratios may have a better matching (true for resistors and capacitors too). Hence, analogue circuit design relies a lot on matching of components rather than absolute accuracy. Hence, the layout techniques must be considered carefully to achieve a good matching between transistors.

One of the common layout techniques for matched transistors involves splitting their gates into a number of parallel gate strips on the basis of unit width. The unit width is chosen in such a manner that all the other matched transistor widths can be implemented as integer multiples of this unit width. This is followed by arranging them in an appropriate pattern like the common centroid technique [13], which is effective in cancelling out the effects of linear process parameter gradients. Another efficient method of laying out the split gate transistors called the stacking of transistors is proposed in [68]. In this, a compact stack of transistors is formed by allowing the transistors to share the same diffusion region, if either of the drain or source terminals are connected together. This arrangement is based on the Euler path discussed in [68], which provides a continuous path from the ground for a NMOS transistor (or vdda/vdd for a PMOS transistor), through all the gate strips once and back to ground (or vdda/vdd). Thus the derived path decides how the gate strips are laid next to each other so that the adjacent drain/sources can share common diffusion regions.

The method of stacking matched transistors is clearer by considering the output current mirror stages as shown in Figure 5.1(a). From the figure we observe that the PMOS and NMOS transistors ought to be well matched for ensuring a good current mirror

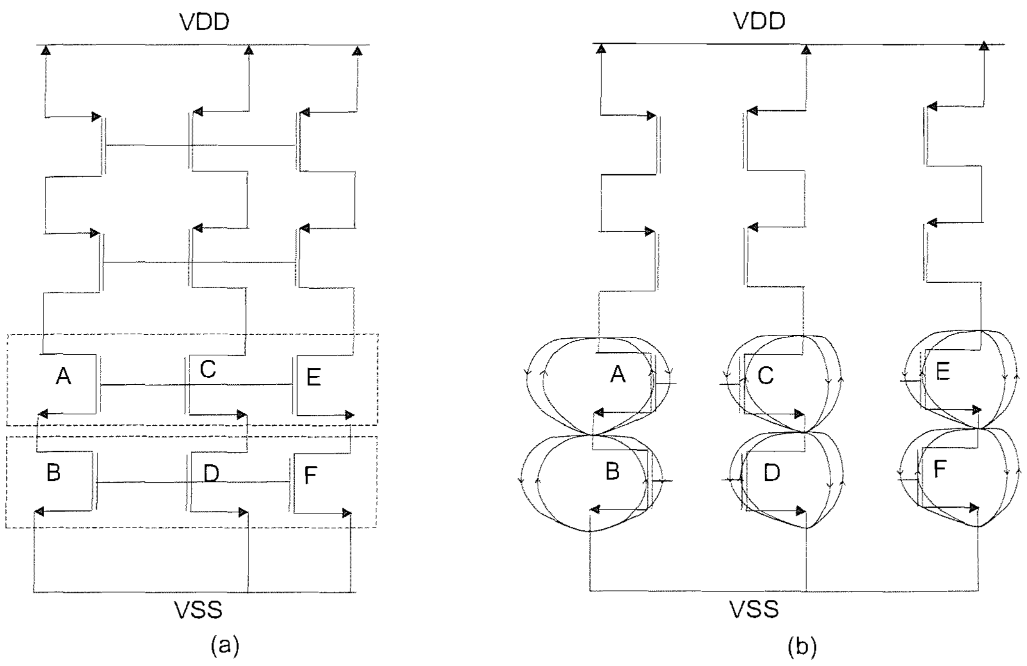


FIGURE 5.1: Stacking of transistors and Euler paths

performance. For simplicity of discussion, let us apply the Euler path to the NMOS transistors of the current mirror alone as shown in Figure 5.1(b). Applying the Euler path for the branch having A and B, the Euler path stack can be implemented by splitting each transistor into two as:

$$B\ A\ A\ B$$

This is similar to the common centroid method as the arrangement is symmetrical about the central axis. But considering the addition of the next NMOS branch having C and D, the Euler path stack now becomes:

$$B\ A\ A\ B\ D\ C\ C\ D$$

, which is not symmetrical about the centre. The arrangement could be made symmetrical, by splitting the transistors into four gate strips, so that the Euler stack path now becomes:

$$B\ A\ A\ B\ D\ C\ C\ D\ D\ C\ C\ D\ B\ A\ A\ B$$

Similarly the Euler path stack for the three NMOS branches in Figure 5.1(b) can be written as:

$$F\ E\ E\ F\ D\ C\ C\ D\ B\ A\ A\ B\ B\ A\ A\ B\ D\ C\ C\ D\ F\ E\ E\ F$$

The above arrangement is common centroid in nature. The placement of dummy transistors (NMOS or PMOS) of minimum length on either ends of the stack is a common practice. The Euler path stack with the dummy transistors can be rewritten as:

$$D1\ F\ E\ E\ F\ D\ C\ C\ D\ B\ A\ A\ B\ B\ A\ A\ B\ D\ C\ C\ D\ F\ E\ E\ F\ D1$$

The dummy transistors (D1) prevents the unmatched undercut effects on the edge of the outermost gate strips (F gate strips). The gate, source and drain terminals of these dummy transistors are connected to gnd (for NMOS) or vdd (for PMOS). Hence, the stack generation method was extensively used for implementing the layout of matched transistors and also sometimes to create compact leaf cells with the following features:-

- the transistors involved are split as multiples of a unit transistors, which are interdigitised.
- the stack of the unit transistors are made common centroid to cancel the process parameter gradients.
- all the gate, source and drain regions are made uniform in dimensions.
- dummy transistors are included in the stack accordingly to prevent the uneven undercut.
- guard rings are provided around the stacked transistors to ensure a good bulk potential.

The above features will be clearer when this method will be mentioned during the discussion about the layouts of the individual building blocks/leaf cells.

5.1.6 Layout Consideration Due to Process Constraints

The fabrication process does impose some limitations, which affect the circuit behaviour and these can be minimised to a certain extent through effective layout techniques. Some of the layout techniques that are used to address these limitations are discussed below.

5.1.6.1 Wide metal slits/holes

As per our earlier discussion, it is necessary that the layout of some specific signals (power supply lines) are made wide enough to minimise the electro-migration and resistance effects. There are no limitations as such to the maximum width that can be allowed for these lines. However, the problem arises during an increase in temperature of the chip. With a considerable increase in temperature, an expansion of the wide metal regions occurs. This repeated expansion of the wide metal, eventually introduces cracks in the isolation and passivation layer that protect the silicon wafer, thus causing the reaction of impurities into the chip with different materials, that can finally affect the circuit behaviour. Hence, it is a common practice to introduce properly spaced slits/holes of specific dimensions as per the process and design rules, to reduce the effect. Similar holes were introduced in the METAL3 (M3) wide power supply rails with a minimum M3 hole width of 1.5μm, minimum M3 hole length of 10μm, minimum M3 hole spacing of 10μm and a minimum 9μm M3 enclosure of M3 hole.

5.1.6.2 Interlayer connections

Vias are important structures used in the layout to directly establish connection between any two metal layers. These interlayer connections are very important, as they define the positions in the current path between the two layers. They are particularly significant in the case of large metal lines carrying larger currents. Considering vias as holes in the isolation layer between two metal layers, a connection is established only when the upper-metal layer fills these holes to connect the lower-metal layer. But in practice it is difficult to size these holes and their subsequent metallic layer fillings exactly to the layout dimensions. The design rules attempt to ensure the reliability of the vias. Via array configurations are also used to address these problems. Similar layout techniques will be highlighted in the sections involving the layout of the blocks/leaf cells.

5.1.6.3 Coverage Rules

Coverage rules are introduced to address a problem called the step coverage effect. The rising or falling slope of a layer as it passes from one area of the silicon wafer to another along with various other layers underneath, is called a "step". A process that uses the "planarization" technique alleviates this problem, as the surface of the wafer is levelled with isolation material between layers. The density rule or the metal coverage rule is related to the planarization technique, which defines specific density/coverage requirements of a given layer over a specific area. This rule for a particular layer would state that, for areas of 100 x 100 μm^2 regions, the metal polygons should cover at least 75 percent of the area, so that enough polygons can ensure the proper planarization for the layers above. It is a common technique to introduce dummy polygons to ensure the layer consistency over the chip region and these rules are normally integrated into the modern CAD tools that support the design rules for the process. Similar dummy techniques were used for the layout whenever the density rule had to satisfied during the layout of the chip. These dummy polygons can be observed more often for the M3 layer and will be highlighted during the discussion for the layout of the chip.

5.1.6.4 Antenna Rules

The "Antenna Rules" deal with process induced gate oxide damage caused when exposed polysilicon and metal structures, connected to a thin oxide transistor, collect charge during the etching process. This may build up a potential sufficiently large to cause breakdown of the thin oxide. This antenna effect is more significant for modern processes supporting smaller feature sizes. Some of layout techniques used to eliminate the antenna effect are:-

- to eliminate the antenna by breaking or cutting the long poly/metal layer and by introducing a bridge in a higher metal
- to place a diode to the substrate close to the gate of the concerned transistor, to provide a leakage path to the substrate.

5.1.6.5 State of Latch-up

The condition of "latch-up" refers to parasitic npnp structure, which may behave like a thyristor and if turned on may short VDD to the substrate VSS. The following layout techniques are adopted to reduce the susceptibility to latch-up:-

- introducing maximum number of substrate and tub contacts.
- introducing minimum space between the substrate and tub contacts.
- ensuring an even coverage of substrate and tub contacts for a given area.
- adopting the method of providing a ring of substrate and tub contacts around the transistor area, also called "guard rings".
- routing the power supply lines in metal only and not in any resistive layer like diffusion or polysilicon.
- grouping transistors of the same type for guard ring protection rather than having to protect against latch-up in different areas.

Guard rings were also incorporated into the layout method of stack generation for transistors and will be mentioned during the discussion of the block level layout.

5.2 Layout of the FPAA

Having performed top-down planning, the layout was implemented following a bottom-up block level layout approach. Instead of doing the layout of the whole core of the chip, it was decided to breakdown the task using a hierarchical approach. Hence it was decided to start with the layout of the DDACCII CAB, then the switch boxes S1 and S2, gradually moving towards higher level in the hierarchy to form the cluster. Finally the whole cluster along with the rest of signal routing and interconnections forms the core of the chip.

5.2.1 Layout of the cluster

Starting with the layout of the cluster, we know that it can be split into instances of the layout for DDACCII CAB and the switch boxes S1 and S2 respectively. The layout of the DDACCII CAB was implemented first, so that verification at a block level could be done using the extracted parameters. The simulation of the functionality of each block using the extracted parameters from the layout, confirms the layout of the respective block. The other form of verification to check the layout is the LVS (Layout Vs Schematic) check. It is after the DRC (Design Rule Checks) for the process, the LVS checks are done to verify the equality between the implemented layout and the schematic of the block. The CAD tools nowadays support all these features for DRC checks, LVS checks and for performing an extraction of all the parasitics in the layout.

5.2.1.1 Layout of the DDACCII CAB

The layout of the DDACCII CAB was implemented by dividing the design again into smaller instances, which could, for example, form a compact block. The method of stacking transistors with proper guard rings, discussed in section 5.1.5 was extensively used for the layout. Tapping the appropriate connections for all the transistor terminals of the stack was easy, because of the equal extensions of the source, gate and the drain terminals in the stack. Hence the method of creating a stack using Euler path was used even for those transistors that did not require matching, as long as a compact stack could be implemented. The layout of the DDACCII CAB was implemented as shown in Figure C.2. After DRC and LVS checks, the layout was extracted for parasitics and verified for functionality and performance characteristics using the same simulation setup used during the design phase of the DDACCII. In these simulations, the simulator is set up to replace the schematic of the design by the extracted design with parasitics.

From the Figure C.2, observe the placement of the stacked transistors and labels indicating the transistors and input/output signal routing. The stacks are placed on both sides of the signal routing channel(top to bottom vdda, gnd! etc) to tap connections using the terminals outside the stack. Also observe some input/output signals routed through the blocks or horizontally in between the stacks (INP, VOUT), which are also called "feed-throughs", in layout techniques to pass signals through a row of cells. The verified layout in Figure C.2 was with eight standard DFC cells which form the 8-bit configuration register. The final implemented layout of the DDACCII CAB is shown in Figure C.3, which is having all the input and output pad structures in METAL3. These dummy M3 structures where added to satisfy M3 coverage rules. The eight standard DFC cells can be observed to be lying parallel to the NMOS and PMOS structures. This final layout of the DDACCII CAB was also extracted for parasitics and verified for functionality, performance and configurability.

5.2.1.2 Layout of the switch boxes

The layout of switch box S1 is shown in Figure C.4. The switches are formed by the NMOS blocks arranged vertically, with the inputs from i1 to i8(in METAL2) . The 8-bit configuration register is formed by the standard DFC cells placed in parallel to the NMOS switches. The pins BIN, CLK, BOUT, vdd! and gnd! are provided. The outputs are brought out horizontally in M2 layer.

The layout of switch box S2 is shown in Figure C.5. The switches are formed again by the NMOS blocks arranged vertically, with the inputs and outputs arranged in the same order as in the schematic block for S2. The gate connections for the inputs and outputs of the switches were brought out in M2 layer. But in order to maintain the order of the inputs/outputs vertically, M3 layer was used for all the long vertical routing, due to the lower resistance and coupling effects it offers. The 24-bit configuration register is formed by the standard DFC cells placed in parallel to the NMOS switches.

The layout of the switch boxes S1 and S2 were also verified with DRC and LVS checks. The parasitic capacitances and resistances were extracted and the circuit layout out was verified for functionality and performance. These layout instances of S1 and S2 along with CABs, were used to build the layout of the cluster, as per the plan discussed earlier. The layout of the cluster along with interconnections, DPDTSW blocks and other signal routing, to form the core is discussed in the next section.

5.2.2 Layout of the FPAA Core

The layout of the core was implemented by using the following cells as per the plan:-

- four instances of the DDACCII CAB cell for CAB1 to CAB4
- two instances of the S1 cell for routing between CAB1-CAB3 and CAB2-CAB4
- one instance of the S2 cell for routing between CAB1, CAB2, CAB3 and CAB4.

The other two blocks required in the core are, two DPDTSW cells and a DDACCII CAB as a test block. The implemented layout of the DPDTSW block is shown in

Figure C.6. The PMOS switch block used earlier in the DDACCII cell (S2 switch) and the NMOS switch blocks in DDACCII, S1 and S2 switch blocks, were used to form the transmission gates in the DPDTSW block. Note the arrangement and layout of the inputs (p1,p2,p3,p4,enb), outputs (o1,o2) and the power supply (vdda!,gnd!). The layout was verified using DRC, LVS checks and also for functionality and performance using the extracted layout of the DPDTSW.

The layout of the FPAA core as implemented is shown in Figure C.7. From the layout observe the placed instance of the cluster, the two DPDTSW cells and the test CAB as per the earlier plan in Figure C.1. The M3 layer was used for bringing out all the input/output connections for all the cells in the cluster, the DPDTSW cells and the test CAB. The other major wide metal signal routing in the vertical direction was also implemented in the M3 layer. The vdd! and the digital gnd! wide strips of M3 metal, with metal slits, can be observed in the left-hand side the layout. The vdda!, MIDV and analogue gnd!, were again implemented as three vertical wide M3 metal strips with slots on the right-hand side of the core. The metal slits are provided to these wide metal M3 strips as per the discussion in section 5.1.6.1.

The routing of the clock tree, as three branches, each buffered with a CLKBUF standard cell, was also implemented in the M3 layer. The bias currents for all the four CABs in the cluster were routed in the M3 layer, from the P-channel current mirror block at the bottom right corner of the core (near "INPBIAS"). This bias current is fed through the "INPBIAS" input and the N-channel current mirror again at the bottom right corner, from which the P-channel current mirrors it into four bias currents for the four CABs. The input bias current for the test CAB at the bottom of the core, is routed from the direct input "INPB" (top right-hand corner), in the M3 layer vertically down the chip. The layout of core was then placed into the available cavity formed by the pad ring. The pad ring was placed exactly as per the plan, with the same pad order from 1 to 68. The input/output and bidirectional pads were connected accordingly to the core. The order and position of the pads were implemented foreseeing the fabricated die to be packaged for the JLCC68 pin package (bonding diagram shown in Appendix D.11). The completed layout of the FPAA chip is shown in Figure C.8.

The completed layout of the chip was also verified for DRC and LVS checks. An extraction of all the parasitic capacitances and resistances was done and simulations where done to inspect the functionality and the performance of the chip. The extracted view of the chip, replaced the schematic for the simulation that verifies the configurability and the target functionality it was configured for. The simulation setup as in Figure B.29 discussed earlier in section 4.5, Chapter 4, during the test of the chip schematic, was again simulated using the extracted view of the chip and was confirmed for functionality.

The other important test which was carried out was the corner analysis of the chip. The corner analysis is a supported feature in most of the CAD tools, that facilitates to inspect the performance of a circuit for extreme conditions or "corners" with respect to a range of temperature and also different types of transistors. The Monte Carlo analysis, in which all independent process parameters are allowed to vary randomly within a predetermined range, could be performed to obtain a comprehensive performance analysis, through an extensive family of curves. The corner analysis was used because its faster in giving a quick overview on the performance of the chip for the extreme corner conditions, and the time available was limited by the submission date. The available parameters are normally defined in the process documents. The parameters available are called typical mean (TM) parameters which have been extracted from typical wafers. Additionally, the worst case tolerances of the main parameters are provided, which are used to establish worst case parameter sets. Four predefined worst case parameter sets are available:-

- WP = worst case power = fast NMOS and fast PMOS.
- WS = worst case speed = slow NMOS and slow PMOS.
- WO = worst case one= fast NMOS and slow PMOS.
- WZ = worst case zero = slow NMOS and fast PMOS.

The corners were defined using these worse case sets, along with varying temperature (0^{o}C,25^{o}C and 85^{o}C) and varying voltage(4.5V, 5V and 5.5V). The corner analysis showed that the circuit should operate satisfactorily over the full range of corners provided. The designed chip was fabricated in AMS CMOS 0.6 CUP by CMP, France. A discussion about the measured results from the fabricated HFPAA and its realities on silicon is included in the next chapter.

Chapter 6

Testing the FPAA

The fabricated FPAA was tested for measuring the performance of the DDACCII CAB and also to test the cluster design implemented on the chip. This chapter includes a brief discussion on some of the tested applications with the corresponding measured results. During the testing of the chip, the test CAB was working, but a defect was found in configuring the cluster of 4CABs using the shift-register. The chip was observed to be drawing more current from the power supply while feeding in the configuration bit-stream serially into the chip. This problem was then traced and found to be due to the shorted inverted outputs of the configuration DFC flip-flops to ground. This was not spotted in the simulations earlier, because the currents drawn from the power supplies were not explicitly monitored.

Identifying the problem a solution was deduced, so that the chip could still be useful in using the CABs of the cluster for target applications. The solution was to reduce the digital power supply from 5V to 2.5V, so that less current is drawn by the flip-flops while clearing the shift-register. Once the shift-register is cleared using a bit-stream of all ones, the current drawn will automatically reduce. Hence, the 4CABs in the cluster could still be used as all the inputs/outputs of the CABs were made available as separate pins on the chip. The following sections in this Chapter discusses some of the tested static and dynamic switching applications on the chip. *All the passive components required along with the CAB for achieving the target analogue static and dynamic applications were connected externally to the chip.*

6.1 Unity Gain Buffer

A unity gain buffer was implemented as shown in Figure 6.1. The output signal was recorded and was observed to be following the input signal (100mV,1 kHz square wave) applied to the Vpp input of the CAB, as shown in Figure 6.2. Figure 6.3 shows the

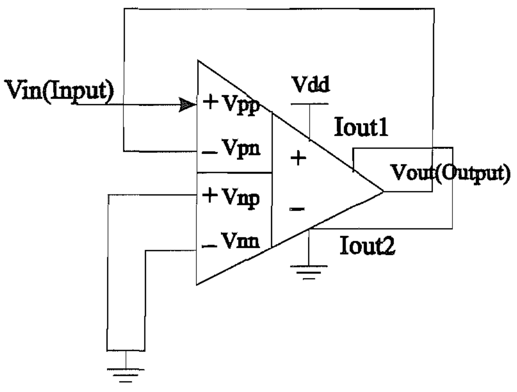


FIGURE 6.1: Unity Gain Buffer

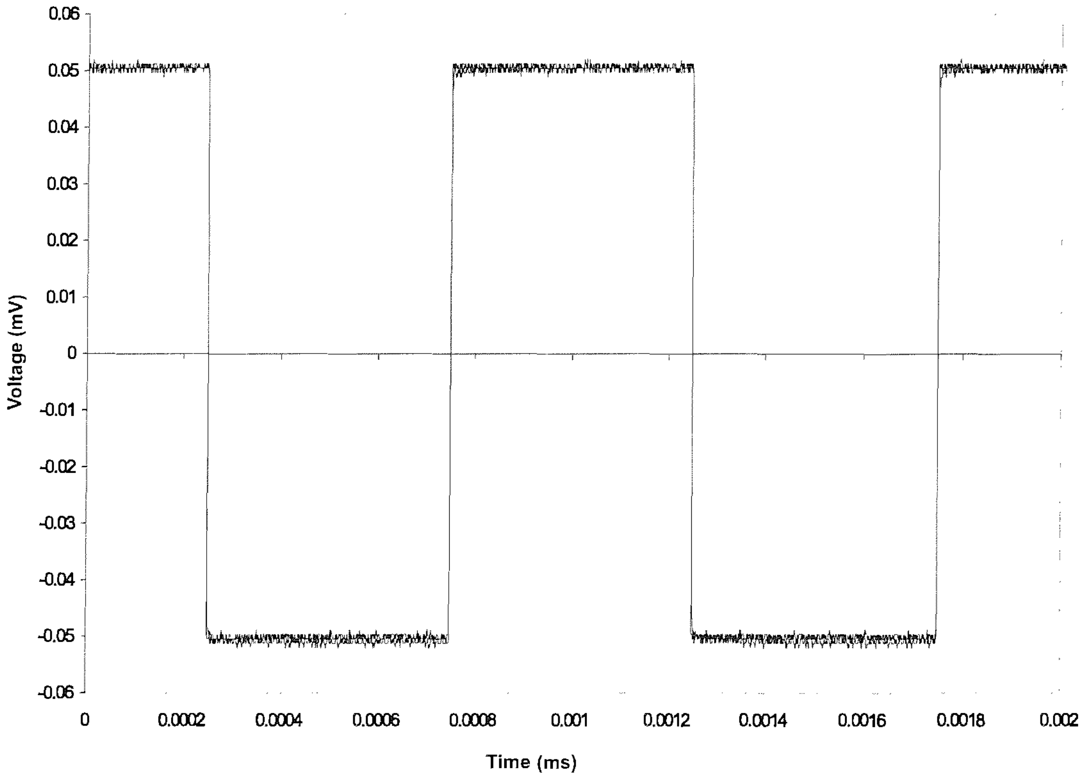


FIGURE 6.2: Measured Output response of the Unity Gain Buffer

measured output signal with an offset voltage approximately 6mV for an input signal (100mV, 500 Hz).

The circuit was also tested for slew-rate and settling time performance and the CAB was found not to be slew-rate limited. The results obtained were found to be similar to the earlier simulated results (Figure B.17). Figure 6.4(A) shows the measured slew rate response without any external capacitive load connected to Vout (other than the total pad capacitance of approximately 10pF). Figure 6.4(B) shows the measured slew rate response with an external capacitive load of 3nF (in addition to the total pad capacitance of 10pF) connected to Vout.

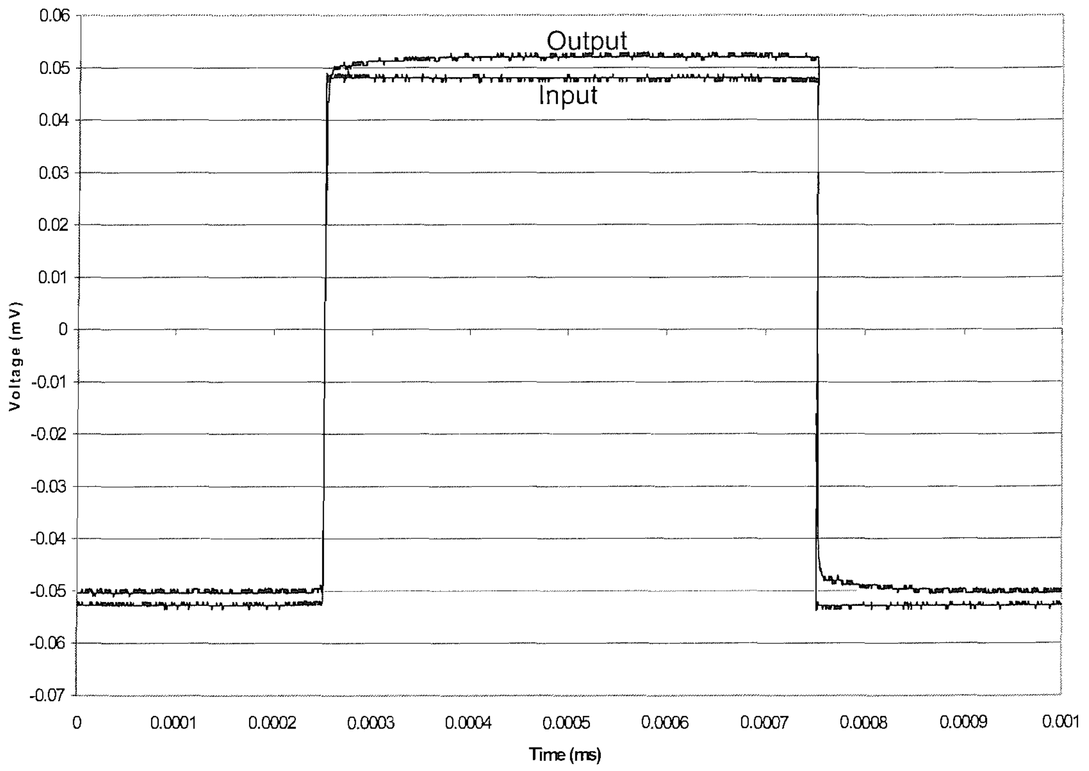


FIGURE 6.3: Plot of Input and Output waveforms showing the measured offset voltage of 6mV in the Output response of the Unity Gain Buffer

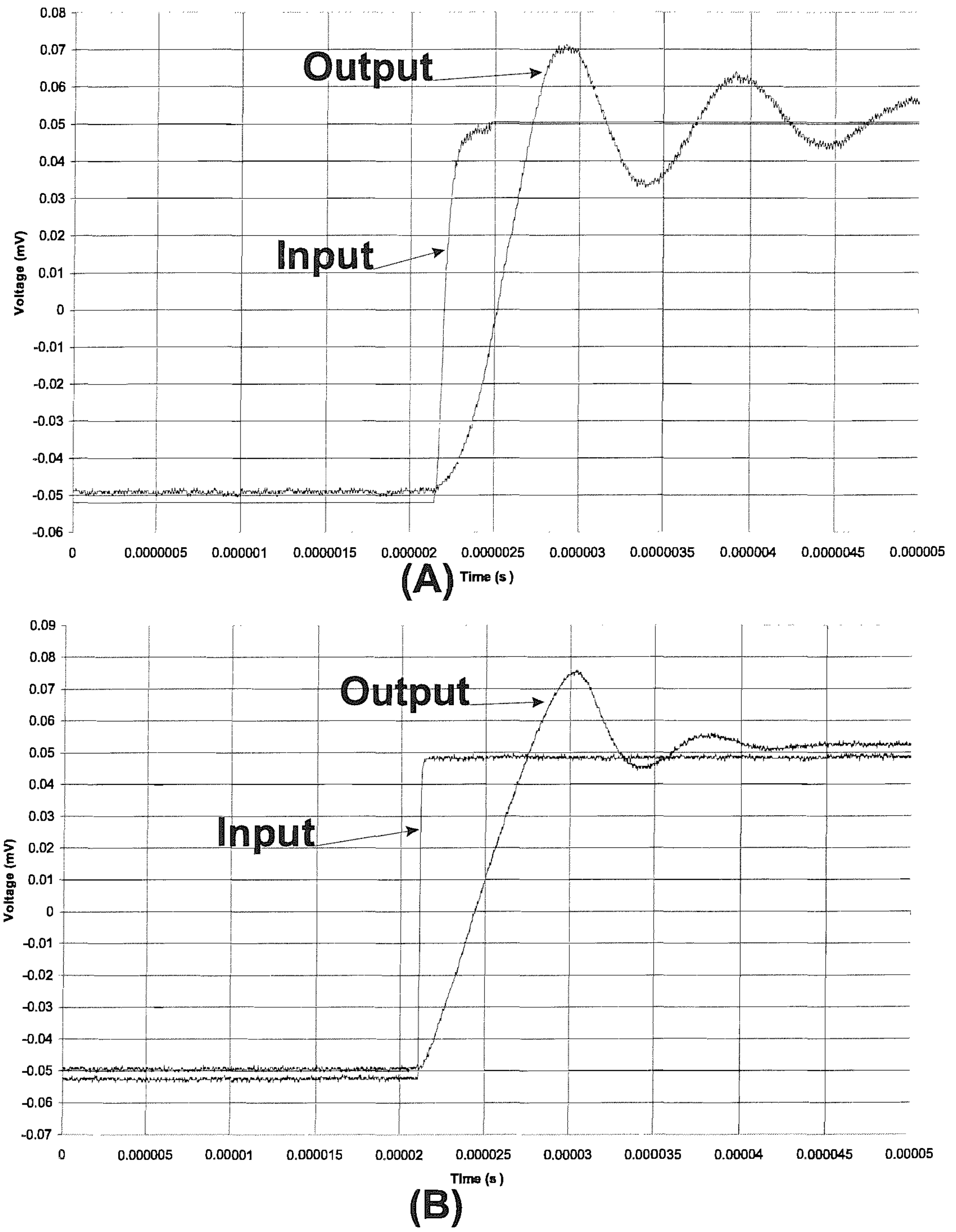


FIGURE 6.4: Measured slew rate response of the Unity Gain Buffer with (A)no external load capacitance (B)external load capacitance of 3nF

The circuit was tested for the DC response at the output terminal for a varying DC voltage at Vpp swept from -2.5V to +2.5V . The measured output DC response is shown in Figure 6.5. The offset voltage was found to be 6mV for 0V at the Vpp input. Figure 6.6 shows the scatter plot of the difference between the original Vout data set and a straight line fit given by the equation y = 0.9865x + 0.0588. The figures shows that the points are reasonably evenly scattered on either side of zero and shows good linearity certainly within the accuracy of the measurements.

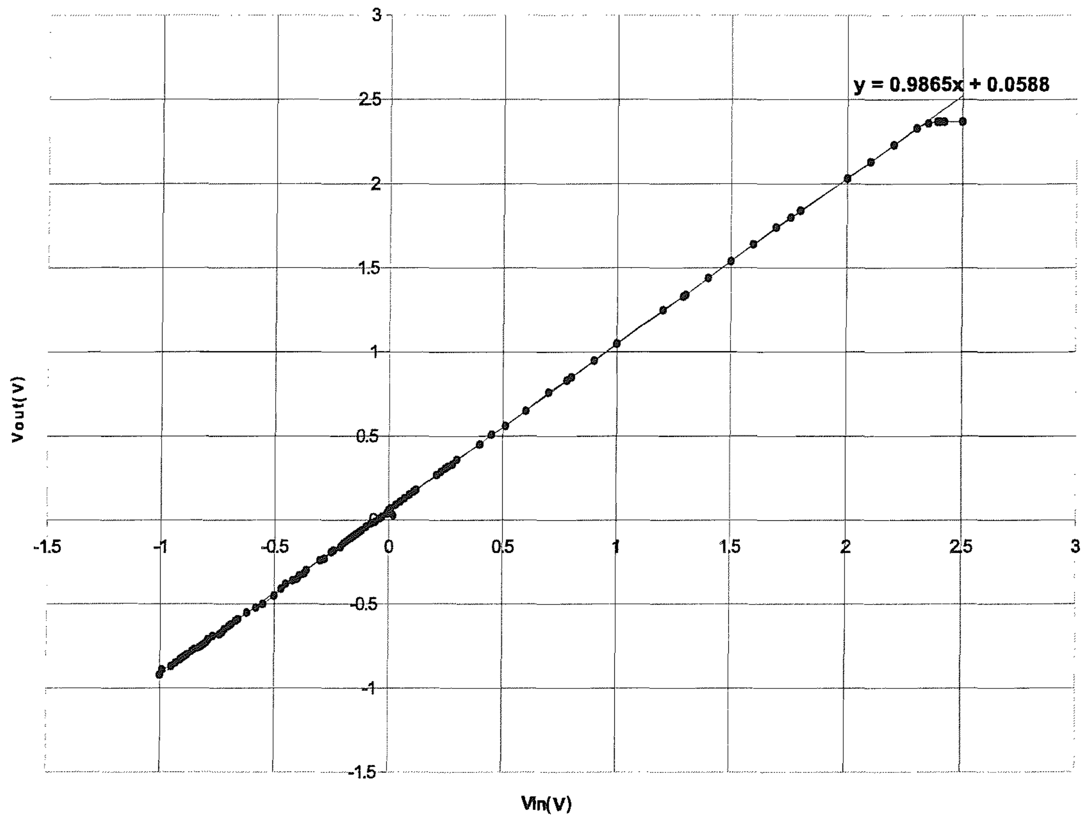


FIGURE 6.5: Measured DC output voltage swing of the Unity Gain Buffer along with a straight line fit for the data points given by the equation y = 0.9865x + 0.0588

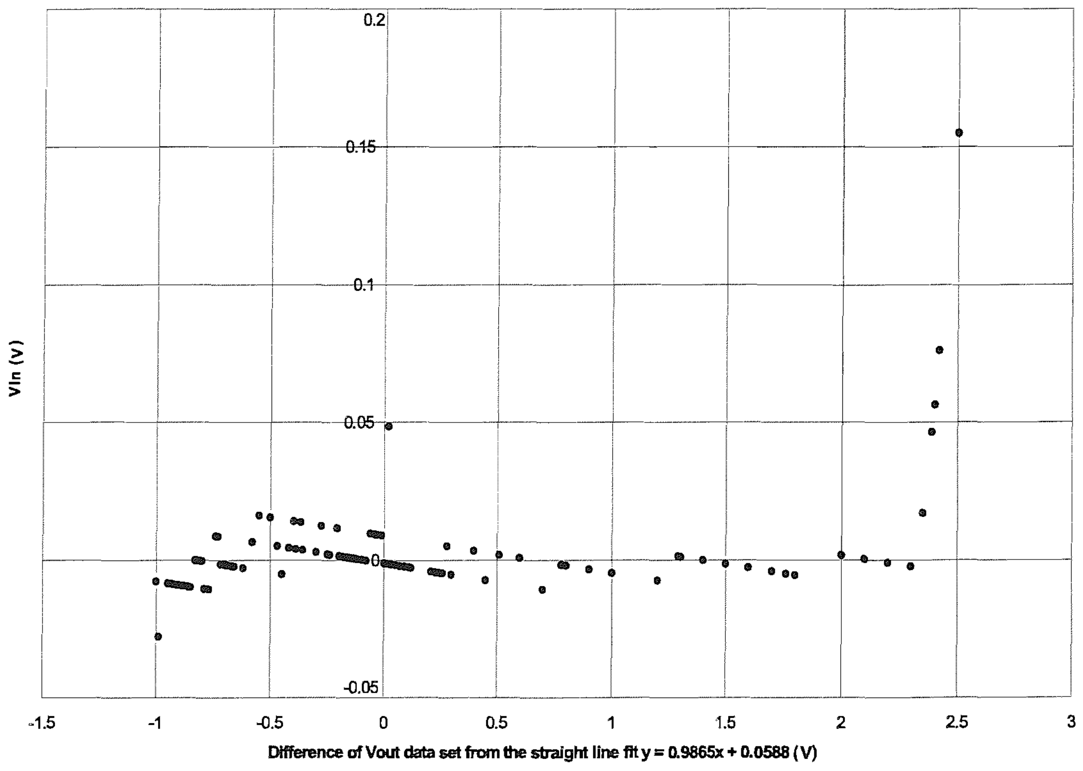


FIGURE 6.6: Scatter plot of the difference between Vout and the straight line fit given by the equation y = 0.9865x + 0.0588 for comparison of linearity

6.2 Instrumentation Amplifier

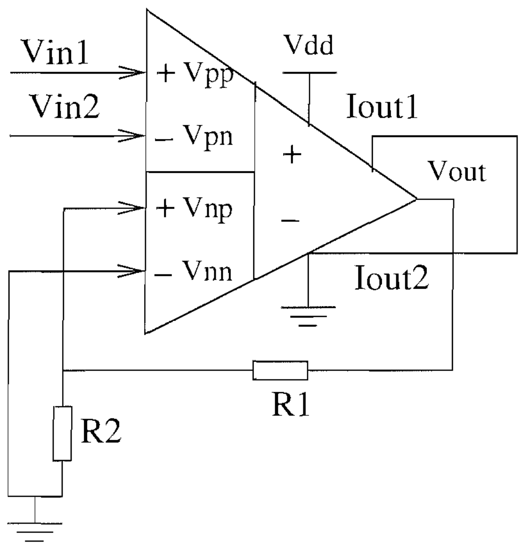


FIGURE 6.7: Instrumentation Amplifier

An instrumentation amplifier can be realised using one DDACCII CAB in voltage mode and two resistors as shown in Figure 6.7. This can be compared to the three opamp implementation using several matched resistors. The gain of the DDACCII CAB based instrumentation amplifier can be controlled by the two resistors R1 and R2. Without the use of the two resistors (R1 =0 and R2=∞), this circuit may be used to realise an effective converter for a differential to a single ended signal. The chip was configured as an instrumentation amplifier and was tested for the common-mode rejection ratio (CMRR), with an input signal of 400mV peak-to-peak amplitude, a frequency of 10KHz and with a gain set to 100 (R1=1MΩ, R2=10kΩ, also R1 and R2 connected externally to the chip). The measured CMRR for this configuration was about 55dB, with a differential gain of 40dB and an approximately measured common-mode gain of -15dB. The measurement of the common-mode gain was difficult because of the small output signal and high level of electrical noise in the laboratory.

The differential gain of the instrumentation amplifier measured for a gain set to 10 (R1=1MΩ, R2=100kΩ) and for an input signal of 100mV peak-to-peak amplitude varying in frequency from 1Hz-1MHz is shown in Figure 6.8. The measured differential gain was found to be 20.5dB . The peak at high frequency is an evidence of under-compensation of the CAB with the effective total load of about 15pF-20pF. Again the common-mode gain was difficult to measure because of external noise, but was about -50dB.

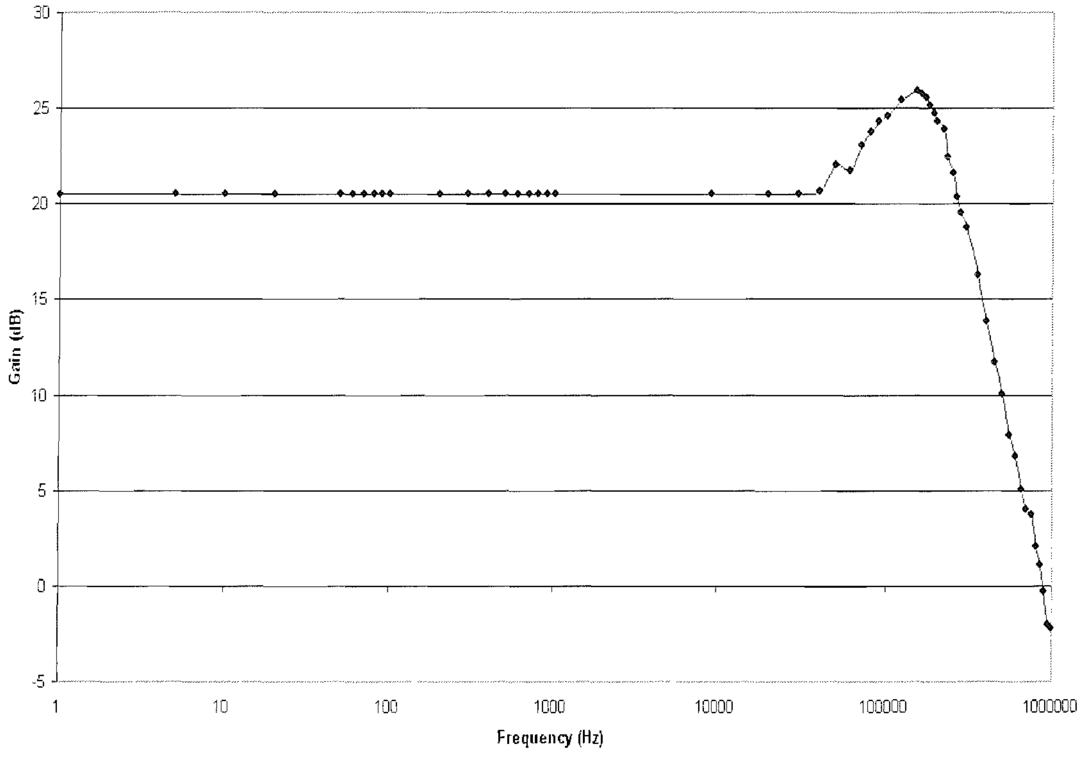


FIGURE 6.8: Differential-Gain of the Instrumentation Amplifier

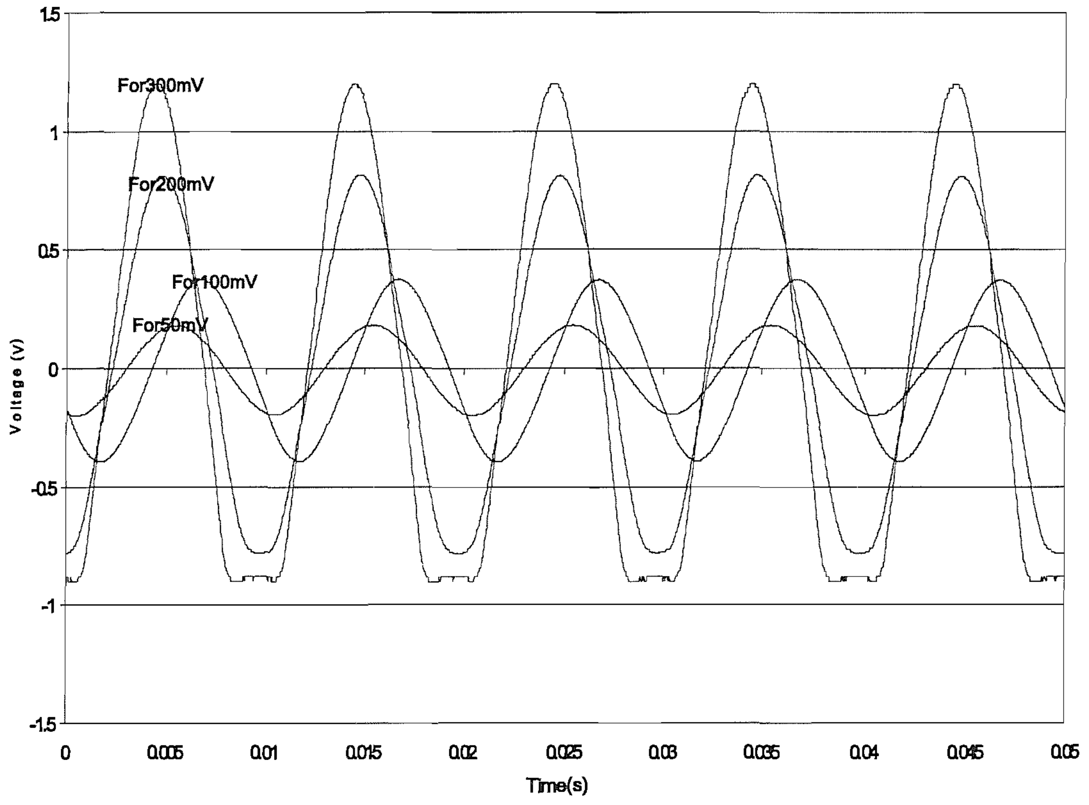


FIGURE 6.9: Output swing of the Instrumentation Amplifier

The maximum output swing for the earlier circuit of the instrumentation amplifier set to a gain of 10 was measured for an input varying voltage at the Vpp input (10mV-2V). A maximum output swing was found to be limited to -0.8V to 1.2V. A few of the recorded waveforms are shown in Figure 6.9. The the clipping observed at the negative peak (-0.9v) of the sinusoidal waveform for an input of 300mV can be seen here. The output waveforms does seem sinusoidal for the rest of the displayed waveforms.

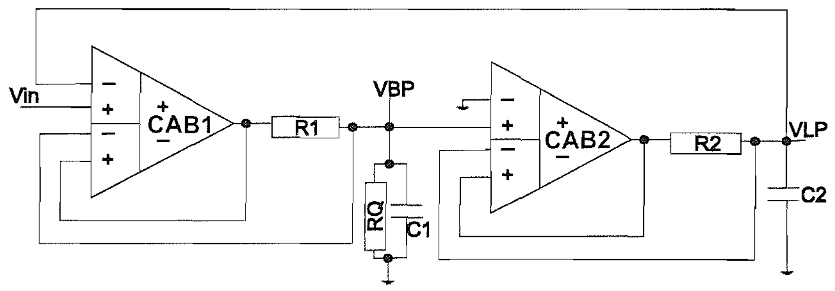


FIGURE 6.10: Resonator Filter

6.3 Resonator Filter

A second-order Tow-Thomas filter implementation, which gives a low-pass filter response and band-pass filter response, along with the desired Q factor and ω_n is shown Figure 6.10. This configuration is composed of a lossy integrator (CAB1, R1, RQ and C1) and a lossless integrator (CAB2, R2, C2). The inverter as in the three opamp implementation of the same filter is not required here because a choice of feedback to positive/negative polarity is available and need not be changed accordingly. Here a negative feedback is required from the output of CAB2 to CAB1 and hence can be achieved as shown in Figure 6.10. Hence, the function of the inverter is implemented by exploiting the differential feature of the DDA. If the DDA's are assumed to be ideal, the transfer function of the band-pass and low-pass outputs are given by:

$$\frac{V_{BP}}{V_{in}} = \frac{\frac{s}{R1C1}}{s^2 + \frac{1}{RQC1}s + \frac{1}{R1R2C1C2}} \tag{6.1}$$

$$\frac{V_{LP}}{V_{in}} = \frac{\frac{1}{R1R2C1C2}}{s^2 + \frac{1}{RQC1}s + \frac{1}{R1R2C1C2}} \tag{6.2}$$

where $\omega_n = \frac{1}{\sqrt{R1R2C1C2}}$ and $Q = RQ\sqrt{\frac{C1}{R1R2C2}}$. To simplify the design, if R1=R2=R and C1=C2=C, then

$$\omega_n = \frac{1}{RC} \tag{6.3}$$

$$Q = \frac{RQ}{R} \tag{6.4}$$

The measured filter response for a Q set to 1 and ω_n=1.8 KHz, with R1=R2=RQ=100K , C1=C2=680pF and V_{in}=200mV is shown in Figure 6.11. The components R1, R2, RQ, C1 and C2 were connected externally to the chip. This response can be compared with the simulated results shown in Figure 6.12.

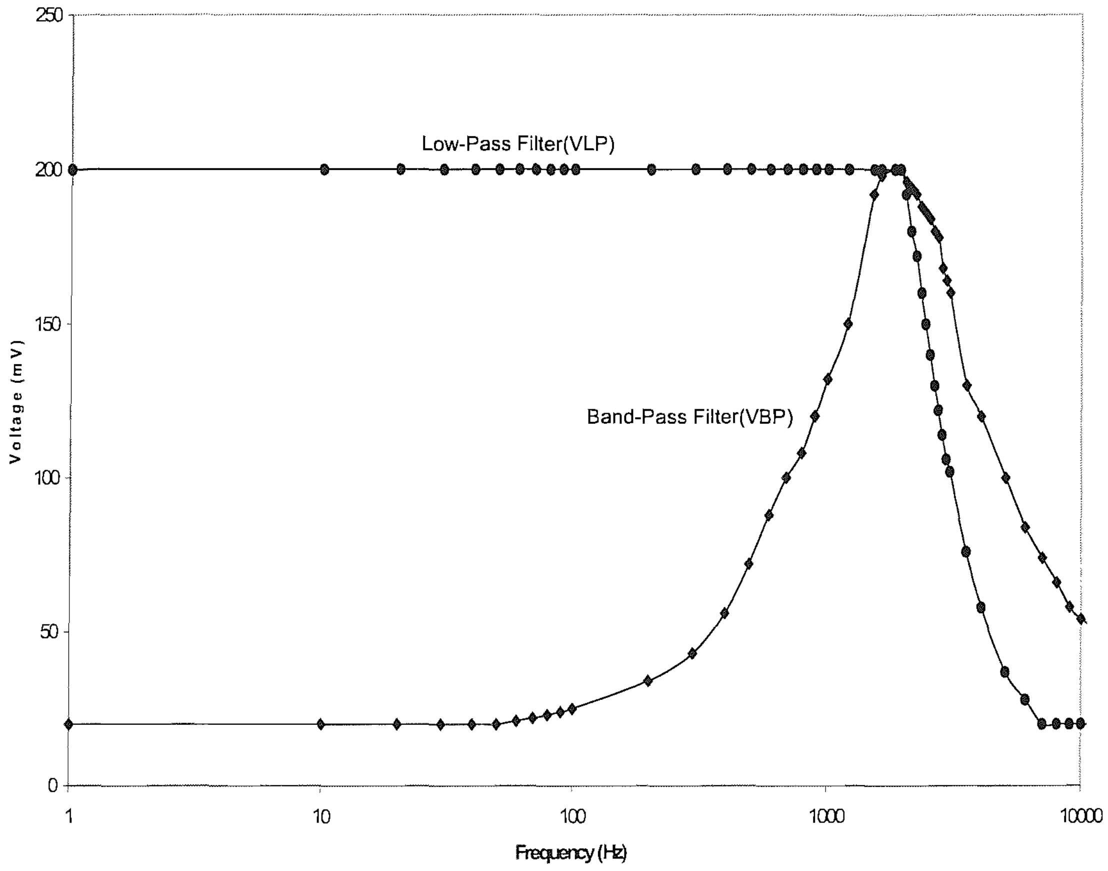


FIGURE 6.11: Measured Filter Response

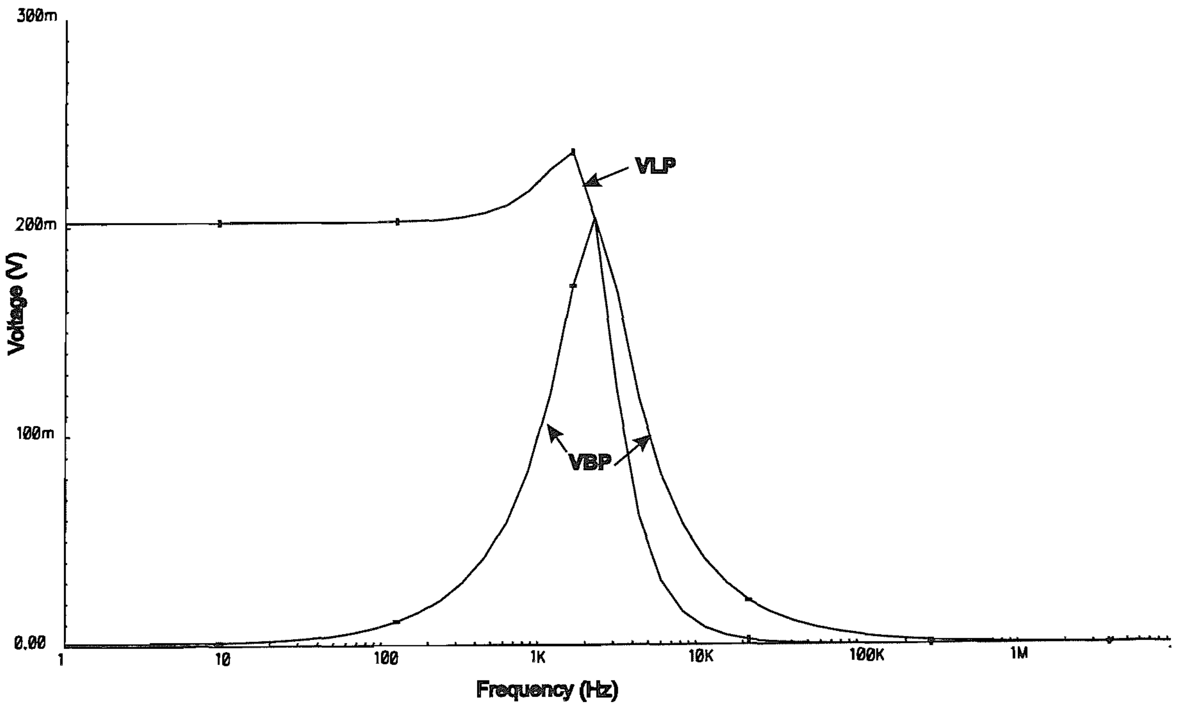
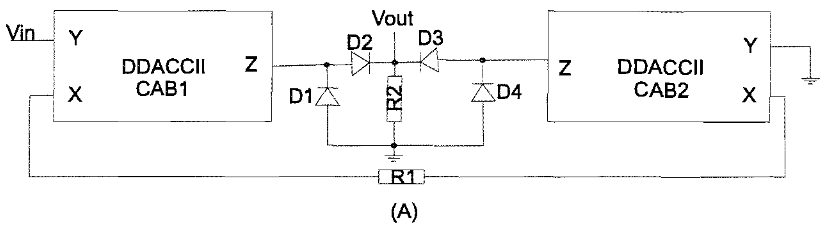


FIGURE 6.12: Simulated Filter Response



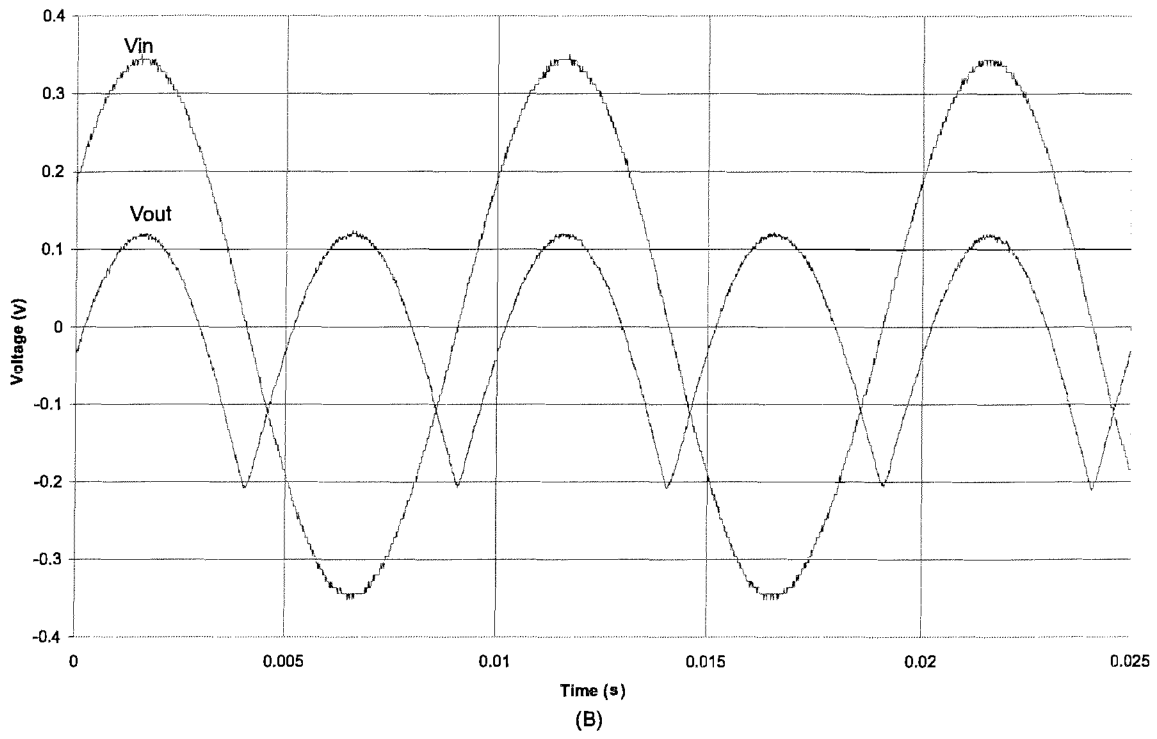


FIGURE 6.13: Full Wave Rectifier

6.4 Full Wave Precision Rectifier

A full wave rectifier can be implemented using two DDACCII CABs as shown in Figure 6.13(A). Here both the CABs form a differential V-I converter such that during the positive input cycle, the output currents of value $Vin/R1$ flow out of the Z terminal of CAB1 and into the Z terminal of CAB2, thus making only D2 and D4 active. Since, D2 is active, the current from the Z terminal of CAB1 flows into the output resistor R2, making Vout=Vin. During the negative cycle, only D3 and D1 are active. Thus, the output current of CAB2 is driven into R2 making Vout=Vin. The measured results for Vin of 0.6V peak-to-peak 100Hz sine wave input are shown in Figure 6.13(B). The components R1, R2, D1, D2, D3 and D4 were connected externally to the chip. The observed level shifting of Vout is due to the AC coupling of the oscilloscope. Hence, the cusps at the bottom are actually at zero volts.

6.5 Current-mode Schmitt Trigger and Voltage Controlled Oscillator (VCO)

A current-mode equivalent of the Schmitt trigger circuit can be implemented using the circuit idea as shown in Figure 6.14(i). CAB1 is configured as a current conveyor. The switch block S1, is a dpdt switch that can switch between two polarities p1-p4 and p2-p3 based upon the enable signal "enb". CAB1 functions as a current conveyor with a current input fed into the X (Vout) terminal (100μA, 10KHz) and the Y (Vpn) terminal grounded. The output current at Z (Iout1 or Iout2) terminal which follow the same current as in X, is used as the activation signal (enb) for S1. The threshold value of +10μA and -10μA is set at the two poles p1-p4 and p2-p3 respectively. The triggered output current available at op1 is subtracted from the input current fed at X. The recorded waveforms from the chip for the tested Schmitt Trigger configuration are shown in Figure 6.15. The observed slow switching speed is due to the associated pad capacitances (4pF/pad) involved in the required off-chip signal wiring for desired circuit connections. Hence, faster response will be achieved by keeping the required connections internal (based on simulated results). A voltage controlled oscillator can

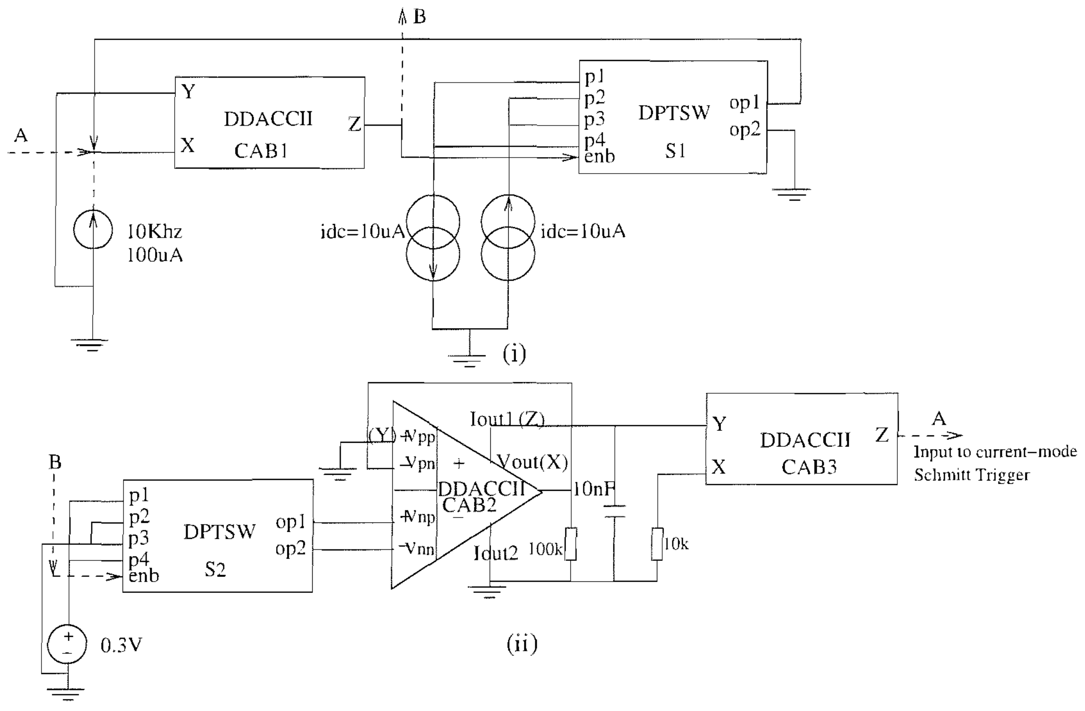


FIGURE 6.14: (i)Schmitt Trigger and (ii)VCO

also be implemented using the circuit concept shown in Figure 6.14(ii). The circuit concept for the Schmitt Trigger discussed earlier, forms a part of the VCO circuit. The CAB2 acts as a constant current source whose polarity may be switched either positive or negative by the action of switch S2. The CAB2 operates in a manner similar to the current conveyor, but with a differential Y input. This is achieved using the second differential pair (Vnp-Vnn) of CAB2. Hence, the flexibility of obtaining a differential Y input for the current conveyor is a clear advantage. The differential input is set by S2, which is fed by an input voltage (which controls the frequency) and the enable signal (B) from the Schmitt Trigger. The controlled differential input voltage to CAB2 and the voltage at the X terminal (CAB2) , generates the current at the Z terminal (CAB2). This current charges and discharges the 10nF capacitor. The voltage stored in the capacitor is fed to the Y input of the second current conveyor (CAB3). It is CAB3, which generates the X input for the Schmitt Trigger discussed earlier. The threshold value of the Schmitt Trigger here was set to +5μA and -5μA at the two poles p1-p4 and p2-p3 of S1 respectively. The additional current input fed earlier into the X (Vout) terminal of CAB1 (100μA, 10KHz), is not required. The recorded waveforms from the chip for the tested VCO are shown in Figure 6.16. The simulation results obtained for the same configuration as Schmitt Trigger and a VCO is included in the Appendix B (Figure B.31, Figure B.32).

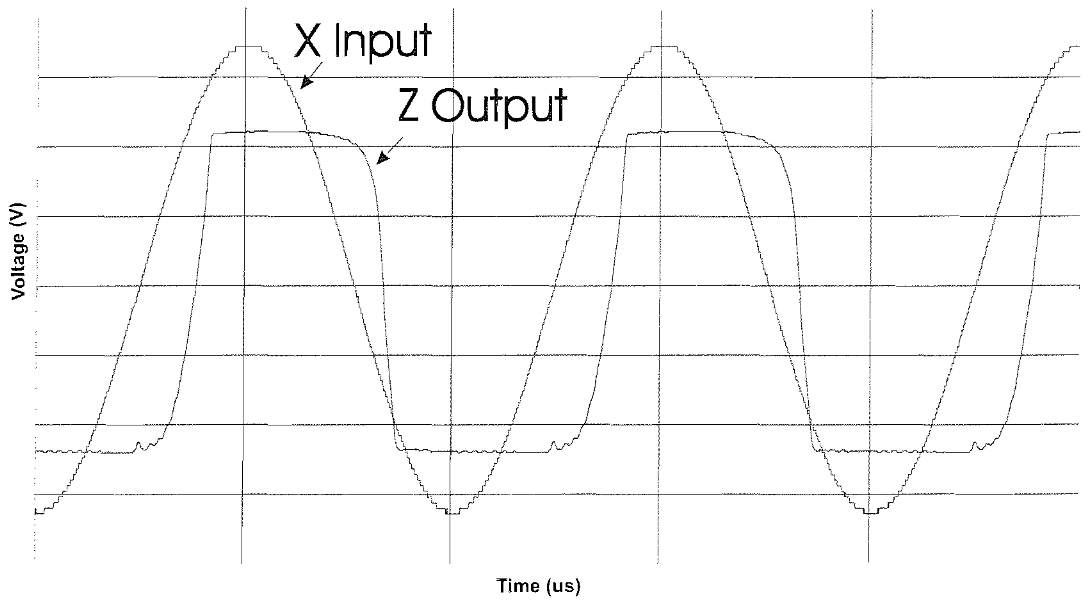


FIGURE 6.15: Measured Schmitt Trigger Waveforms (Scale: x-axis= 50μs/div, y-axis= 1V/div)

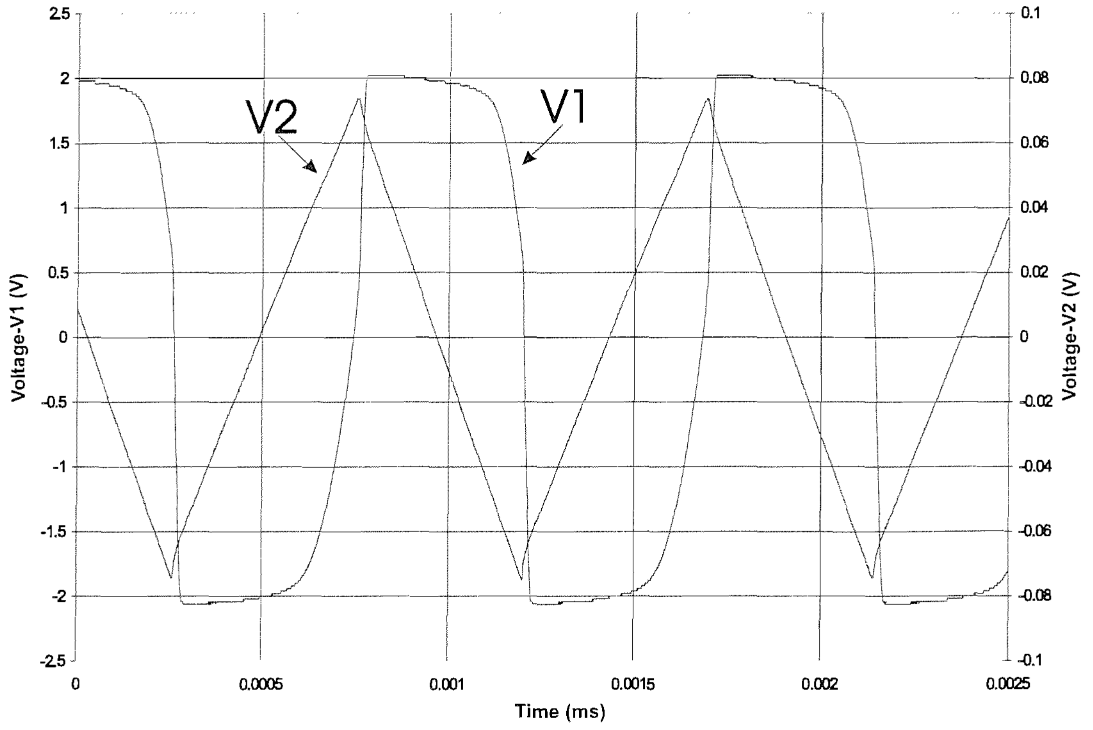


FIGURE 6.16: Measured VCO Waveforms (Scale: x-axis= 0.5ms/div, primary y-axis voltage V1 (Enable signal 'B')= 0.5V/div and secondary y-axis voltage V2 (Oscillating voltage in the 10nF capacitor)= 0.02V/div)

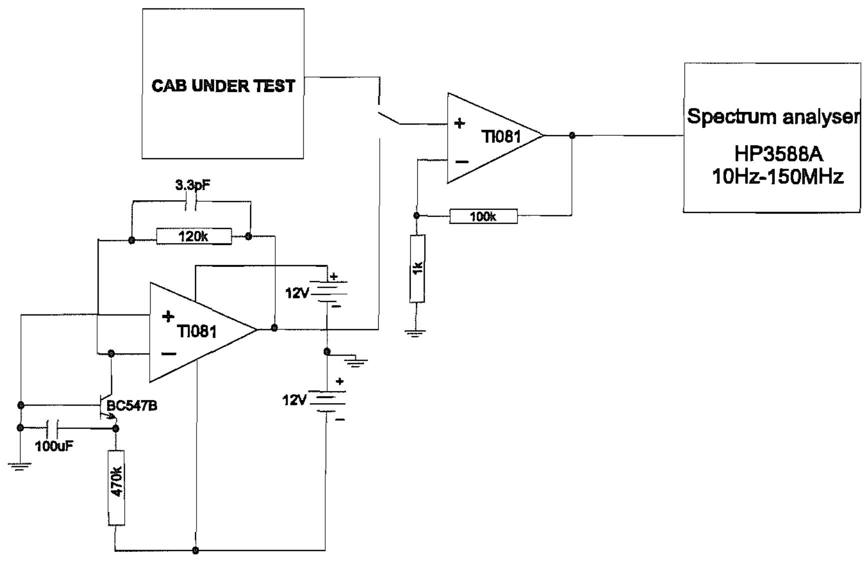


FIGURE 6.17: Circuit used for noise measurement

6.6 Noise Analysis

The test CAB on the chip was configured as a unity gain buffer for measurement of noise generated by the CAB alone. This measurement of the noise using a spectrum analyser was difficult without a reference noise level. Hence, a noise generator circuit, which uses an opamp set with an appropriate gain (100) to measure the collector noise current in a bipolar transistor (BC547B) was used. This measured noise was used as a reference noise level (A=-65dB), to identify the noise level of the CAB (B=-78dB) on the spectrum analyser (HP3588A 10Hz-150MHz) as shown in Figure 6.18. The difference in the noise levels of approximately 13dB was in used in calculating the noise generated by the test CAB configured as unity gain buffer, which was approximately $78\frac{nV}{\sqrt{Hz}}$. This noise is comparable to the simulated noise response of $74\frac{nV}{\sqrt{Hz}}$ shown earlier in Chapter 4. The theoretical expression used for calculating the noise of the noise calibration circuit is:

$$V_n^2 = 2qI_c\,(120K)^2 + 4kT\,(120K) + V_{no}^2 \tag{6.5}$$

where V_n^2 is the voltage noise power spectral density from the calibration circuit, first term is the collector current noise, second term is the noise from the resistor and the third term is the noise of the opamp (negligible). For the value of I_c=25.5μA, q the charge of an electron and k Boltzmans constant, the value of V_n = 346 x $10^{-9}\frac{V}{\sqrt{Hz}}$.

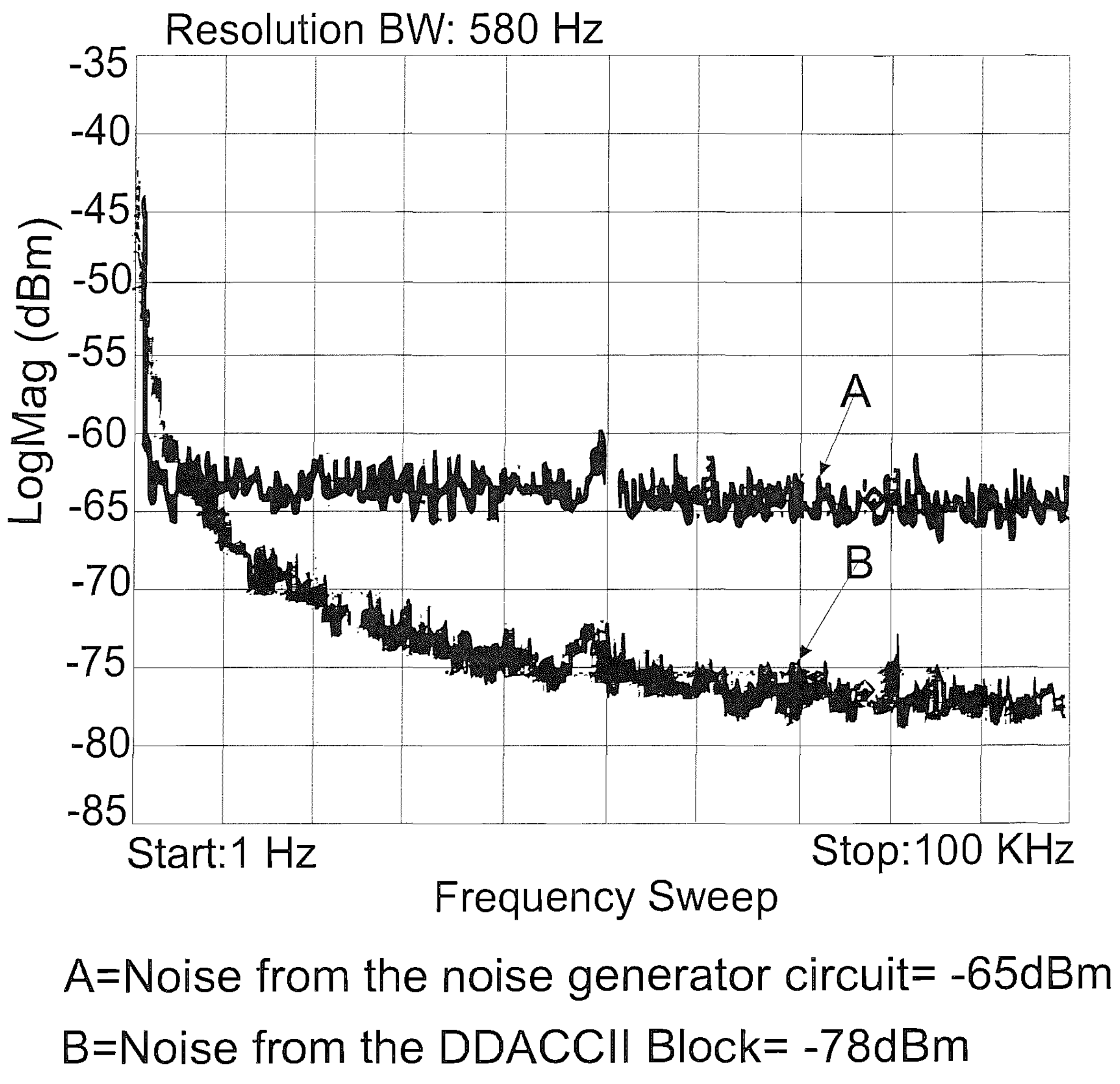


FIGURE 6.18: Measured Noise Response of the CAB

6.7 Summary of the DDACII CAB measured results

The following Table 6.1 summarises the measured DDACCII CAB performance parameters from the chip, compared with the simulation results earlier in Chapter4. Note that the measured differential gain is significantly less than that anticipated from simulation, but the reason for this has not been identified.

Parameter	Measured	Simulated
Differential Gain	40dB	63dB
Gain Bandwidth	900kHz	1.2MHz
CMRR	55 dB	78dB
Output Swing	-1.2V to 2.2 V	-2V to 2.4V
Offset Voltage	-6mV	-4mV
Settling time response with 10pF pad capacitance	3.5μs	0.9μs
Power Consumption	7.3mW	6.5 mW
Measured noise	78 $\frac{nV}{\sqrt{Hz}}$	74 $\frac{nV}{\sqrt{Hz}}$

TABLE 6.1: Summary of the Measured DDACCII CAB Parameters

6.8 Conclusions from the tested chip

From the tested applications on the FPAA, the functionality was verified and found to be similar to that of the simulated results obtained during the design. But in all the tested cases the pad capacitances seemed to be the limiting factor for achieving a performance similar to the simulation results. Hence, the performance could have been improved by minimising the off-chip connections for passive components particularly, by making these required passive components available internally as a part of the CAB or the chip. The power consumption of the FPAA was not being considered as a design strategy. But the power consumption of our CAB alone was measured and was found to be about 7.3mW (test CAB on the chip).

Chapter 7

Flexibility analysis of the Interconnection Architecture

7.1 Introduction

The earlier chapters have concentrated on examining the suitable CAB for a FPAA. They examined the design strategies and requirements for the CAB granularity and discussed the design of a CAB, which attempts to meet the requirements of continuous-time operation, minimising the number of external passive components required and providing a flexible building block that can achieve a wide range of analogue signal processing functions. In addition the case has been made for adopting a hierarchical architecture for the FPAA. The suitability of our CAB in a hierarchical architecture with maximum routability between CABs, depends entirely on the flexibility of the interconnection architecture. This flexibility is directly related to the CAB granularity and the switching structures used for the expected interconnectivity between CABs. Hence, the analysis of the flexibility of the HFPAA interconnection architecture is required. The relationship between the routability of our HFPAA and the flexibility of its hierarchical interconnection structures is yet to be explored. This analysis would determine the feasibility of accommodating higher levels of hierarchy in our HFPAA to suit larger and more complex analogue systems like Analogue Neural Networks (ANNs) and other forms of large signal processing arrays.

There has been relatively little work on the interconnection strategies for FPAAs. There has been prior work on the interconnection complexity for FPGAs and HFPGAs based on the Rent's Rule, used for evaluating chip architectures before they are manufactured. Rent's rule has also been used to generate synthetic benchmark circuits to evaluate CAD tools for FPGAs [88]. This rule is also efficient for prior wire length estimation, where these techniques use Rent's rule for evaluating the partitioning behaviour of digital circuits. This is later used for predicting variance of the terminal count distribution using a set of benchmark circuits for estimating Rent parameters, typically the `intrinsic` Rent exponent to characterise the quality of the partitioning algorithms. There are other methods for estimating the Rent exponent in [20] and references contained there in. But similar methods may not be applicable for analysing the interconnection complexity for our HFPAA architecture, due to the absence of any established set of analogue benchmark circuits. Existing partitioning algorithms may not be useful here because of the independent nature of our CAB to achieve the target application, such that a functionality cannot be partitioned on two or more CABs. This might also be due to the inherent difference between the structure/topology of analogue circuits and digital circuits. *Hence, we present here our methodology for establishing relationship between the routability of our HFPAA and the flexibility of its hierarchical interconnection structures, not relying on the quality of partitioning/routing algorithms as in digital circuits.*

The development of an HFPAA analogue router adopting a suitable routing algorithm was considered. But, this had less significance compared to the development of a methodology for an interconnectivity analysis of our HFPAA. The development of an FPAA router would depend on the methodology for studying the interconnection complexity of the hierarchical interconnection structure. Information on the final target HFPAA interconnection architecture is required to be embedded into the router, to enable routing of the analogue circuits. Hence, the development of the analogue router is proposed for future work.

7.2 The Hierarchical Architecture for FPAA

The CABs in an FPAA can be connected in a hierarchical structure as shown in Figure 7.1. Initially, k CABs are connected to form a sub-cluster with a switch block. Then k clusters are recursively connected together as a super-cluster. In the figure the sub-cluster size is 2, connected by the switch block S1. This sub-cluster is recursively connected using the switch blocks S2 to form a bigger cluster of 2 sub-clusters, and so on at the next hierarchical level using the switch blocks S3 and S4. This FPAA can be defined as a 2-HFPAA, since the cluster size is 2 at the lowest clustering level of the design hierarchy tree. Similar tree representation and other concepts related to this hierarchical architecture can be found in prior work on HFPGAs in [99] and references contained there in. They discuss the flexibility of HFPGAs over FPGAs in reducing the number of switches/pass devices (which contribute to the delay) for connecting the same number of logic blocks and still achieve high routability. Hence similar definitions and discussion of advantages are avoided here.

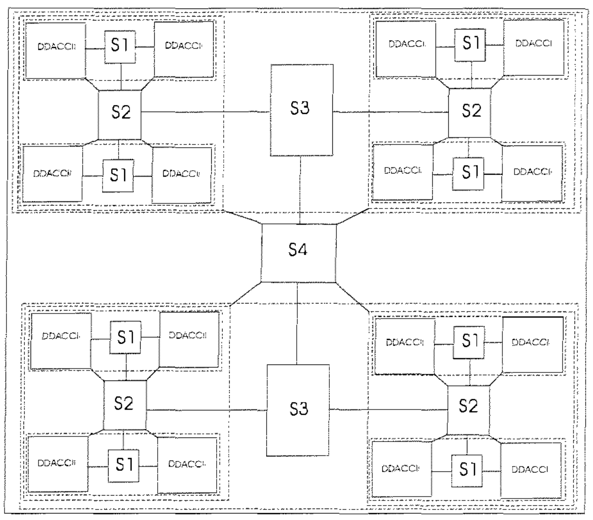


FIGURE 7.1: Hierarchical Architecture of a conceptual HFPAA

7.3 Experimental Definitions

Adopting the hierarchical architecture for connecting our CABs, the flexibility of connection of the switch blocks of a k-HFPAA is of prime interest as it is a deciding factor for achieving high routability of a design or set of circuits. Here the routability can be defined as the ratio of the expected number of connections successfully routed to the total number of connections. Considering a cluster of a 2-HFPAA, we define the flexibility associated with switch block in the sub-cluster(S1) and that with the switch block S2 of the cluster as Fs. *The flexibility of the switch block here is defined as the number of connection paths it can support between the CABs in a sub-cluster/cluster.* The switch count of a switch block is also assumed to be equal to the value Fs, which again reflects the required switch structure supported with appropriate programming technology, to facilitate the expected interconnectivity. To estimate Fs also requires the pin flexibility Pf of the CAB pins. *The pin flexibility Pf is defined as the number of other CAB pins each pin can be connected to; locally and hierarchically.* This Pf includes Pfs local pin flexibility (connectivity to the CAB pins of the same sub-cluster), Pfd downward pin flexibility (incoming connections from other sub-clusters) and Pfu upward pin flexibility (connectivity to other sub-clusters through S2). The total number of input/output terminals required for a sub-cluster/cluster is of interest, which would help in the estimation of the total number of input and output pins/pads required for the FPAA. The "pins" of a sub-cluster/cluster are distinguished from the "pins" of a CAB, by calling them the "terminals" of a sub-cluster/cluster.

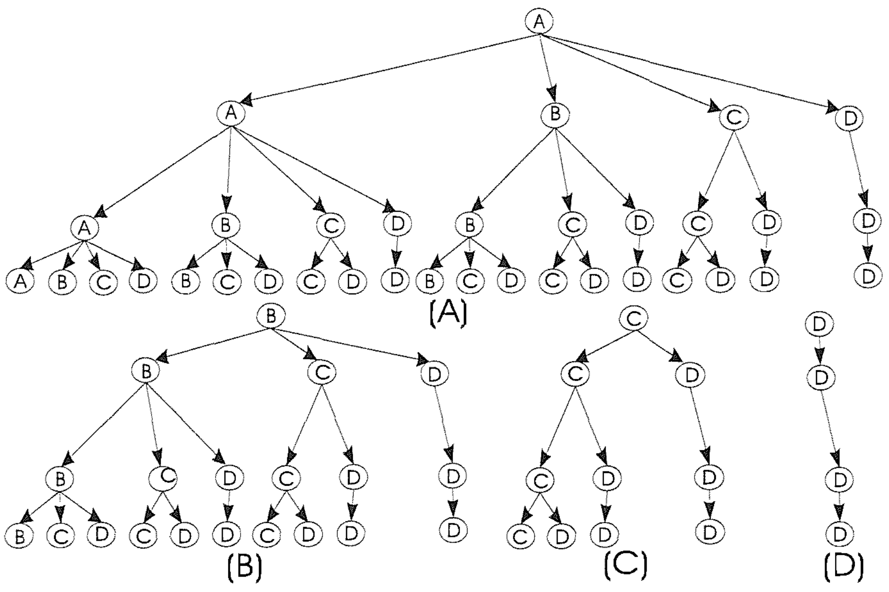


FIGURE 7.2: NPGCL for cluster size 4 & group size 4

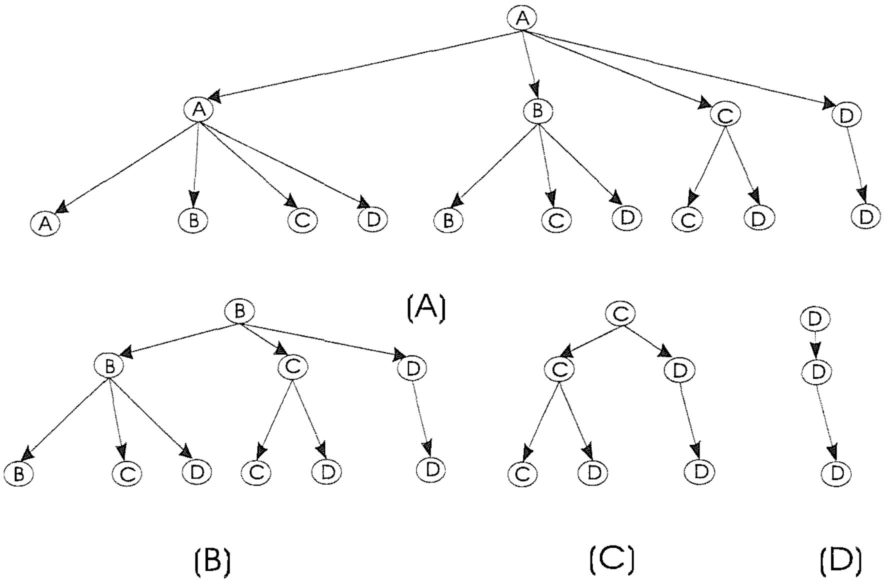


FIGURE 7.3: NPGCL for cluster size 4 & group size 3

7.4 Methodology

The flexibility of the hierarchical architecture and hence its effect on successfully routing analogue circuits, probably cannot be answered by following a set of benchmark circuits, and then extracting Rent exponents to predict the interconnection complexity. This is because we do not have an accepted set of benchmark circuits, which could be used to evaluate our architecture or the CAD tools to aid the analysis of partitioning and routing of analogue circuits on the HFPAA. Also, we believe that partitioning is less of an issue for analogue circuits as opposed to digital circuits. This is because the functionality of our CAB is such that there is less interdependence and interconnectivity between CABs. This would not necessarily be true for CABs of finer granularity.

The starting point for any attempt to analyse interconnectivity has to be a set of realistic circuits, which will perform the type of signal processing for which the FPAA may be used. A set of 33 basic analogue circuits that can be implemented by our CAB were identified and assigned a circuit identification value. Each circuit was classified according to the number of CABs (NOC) required, number of inputs (NOI), number of outputs (NOO), number of external connections (NEC), number of grounded connections (NGC), the mode of operation (MOO), and mapped on a k-HFPAA. Some of the considered circuits are shown in Table 7.1. Initially a cluster of a 2-HFPAA consisting of 4CABs or 2 sub-clusters was considered in our analysis. So the question which arises is what is the minimum number of terminals required for a sub-cluster/cluster for 100% utilisation of the CABs?

The objective is to identify all the combinations of circuits, which will fit in a cluster and utilise all the CABs within the cluster. For a cluster size of 4, a maximum group size of 4 circuits from our set of 33 circuits can be accommodated, while the minimum number is 1. For all combinations of circuits, which will fit in a single cluster, we need to identify the number of terminals (n) required. This has been achieved using our `NPGCL (Non-Permuting Grouped Combinations Listing)` Algorithm, to identify all possible combination of 4 circuits (irrespective of the NOC) and then filtering this set to identify those combinations, which will fit within 4CABs. For those combinations that will fit, the number of terminals (n) may be identified. Then the algorithm must be applied again for groups of 3 circuits, then groups of 2 circuits and for groups of 1 circuit. This will identify those combinations of multi-CAB circuits, which will fit in a cluster. The same algorithm may also be used to explore the consequences of changing the cluster size.

Our NPGCL algorithm is illustrated by an example. Let us consider an input circuit set of 4 identified as A, B, C and D. The task is to generate the non-permuting combinations of these 4 circuits for a cluster of 4CABS. The NPGCL is recursive in nature. The NPGCL is used to build 4 trees involving A, B, C and D as shown in Figure 7.2. These trees generate all the non-permuting combinations of interest. The number of levels (height) of these trees is equal to the group size 4. The first tree in Figure 7.2(A) contains the whole input data set. The subsequent trees in Figure 7.2(B),(C) and (D), contain the input data set reduced successively by one member, in order to avoid permutations. The apex of the tree is assigned to the first member the appropriate data set. The child nodes of the second level are assigned as the members of the current data set. The assignment of child nodes to the second and subsequent levels follow a pattern. This pattern involves a successive reduction of the data set by a member, while moving across from left to right. Similarly, this pattern is repeated at successive levels of the tree. Each sequence can be obtained by starting at the apex and traversing down each

Circuit ID	Circuit	NOC
A	Inverter	1
B	unity gain buffer	1
C	Adder/Subtractor	1
D	Level Shifter	1
E	Doubler	1
F	comparator with hysteresis	1
G	opamp with offset cancellation	1
H	voltage-controlled-current-source	1
I	Instrumentation Amplifier	1
J	Integrator with phase lag compensation	1
K	voltage-controlled-voltage-source	1
L	current-controlled-voltage-source	1
M	current amplifier	1
N	current-controlled-current-source	1
O	current differentiator	1
P	current integrator	1
Q	non-ideal gyrator	2
R	2nd order filter	4
S	Universal active filter	8
T	schmitt trigger	1
U	vco	3
V	dda-differential integrator	2
W	grounded resistor	1
X	dda-differential integrator	1
Y	4th order low pass filter	4
Z	Average circuit	1
a	State-variable filter	3
b	V/I converter	1
c	modulation cell	1
d	4 quadrant multiplier	3
e	Resonator filter	2
f	Full Wave Rectifier	2
g	Single to differential current converter	2

TABLE 7.1: The List of Circuits mapped on our HFPAA

parent/child nodes until the last level of the tree is reached. So the first set of sequences from Figure 7.2(A) are AAAA, AAAB, AAAC, AAAC and AAAD. The next set is AABB, AABC and AABD and so on. Hence, all the possible combinations containing A are obtained by traversing through the tree A. Similarly the combinations starting with B (excluding A), C (excluding A and B), D (excluding A, B and C) are obtained by traversing the corresponding trees shown in Figure 7.2(B),(C),(D) respectively. For a group size of 3 and for the same set of circuits A,B,C and D, the corresponding trees applying NPGCL is shown in Figure 7.3.

Cluster	Group Size	NOGC	filtered on NOC
4	4	12650	4CABs
4	4	46255	>4CABs
4	3	11875	4CABs
4	3	32689	>4CABs
4	2	81	4CABs
2	2	253	2CABs

TABLE 7.2: Results for varying cluster sizes & filters

7.5 Experimental Results

The NPGCL algorithm was implemented in C++ and was applied with an input set of 33 circuits, grouped in sizes of 4, 3 and 2, such that combinations generated were subsequently filtered to select those that occupy only 4CABs. The NPGCL was also applied for generating combinations for a group size of 2 such that the generated combinations occupy only a sub-cluster of 2CABs. The Table 7.2 shows the number of generated combinations (NOGC) from the algorithm with various filtering conditions. Along with the generation of the combinations with applied filtering conditions, parameters other than NOC (NOI, NOO, NGC, NEC, MOO) related to each circuit were included. The sum of NOI and NOO, giving `n` was used to estimate the total number of terminals required for cluster size of 4 and 2 from the generated circuit combinations, for a circuit coverage nearing 100%. Figures 7.4 (A) and (B) shows the number of circuit combinations as function of the total number of terminals `N(n)` and also the cumulative probability that a combination may be realised with `n` terminals and `N` CABs. The NPGCL was used to estimate the NEC for varying cluster sizes. Figure 7.4(C) shows the distribution of NEC for all N(n) for a cluster size of 4CABs. From Figures 7.4 (A) and (B) we observe that the number of terminals N(n) required for a cluster size of 4 is approximately double of that required for a cluster size of 2, for 100% circuit coverage. This is expected because the maximum number of connections corresponds to each CAB being a separate circuit with external terminals. The mean number of terminals also appears to double. This indicates that our illustrative input circuit set is biased towards single CAB circuits. Hence, the choice of the input circuit set will affect the shape of the distribution N(n) and this merits further investigation.

Another application of the NPGCL was achieved by applying it to a completely different data set, i.e the pins available in a CAB. This was done to get more information on the internal switch structure of a sub-cluster and cluster. The internal switch box structure for an expected interconnectivity between CABs in a sub-cluster or cluster, can be obtained by estimating the total number of switches in each hierarchical level. For this, the NPGCL was applied on an input set of terminal pins depending upon the cluster size. For example for a sub-cluster of 2CABs, consider the 7 pins of CAB1 as A(Vpp), B(Vpn), C(Vnp), D(Vnn), E(Vout), F(Iout1), G(Iout2) and similarly for CAB2

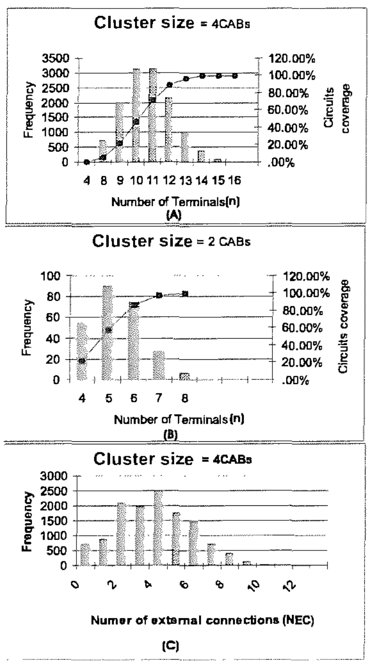


FIGURE 7.4: Distribution of N(n) and the integrated probability distribution

as a(Vpp), b(Vpn), c(Vnp), d(Vnn), e(Vout), f(Iout1), g(Iout2). Applying NPGCL on the input set of 14 pins in total, with a group size of 2 representing a connection between two pins. These will include the CAB1-CAB1, CAB2-CAB2 and CAB1-CAB2, to a total of 105 distinct connections, of which 49 were filtered to be CAB1-CAB2. Now out of the 49 CAB1-CAB2 connections other filters may be applied based on the allowed/required connections between voltage type and current type pins, which will determine the required Fs. For example connections like Ee, Ff, Fb may not be required as a connection to be routed in a circuit at all. Similarly the NPGCL applied on an input set of pins of 4CABs, and later filtered using the same constraints used earlier, will determine the Fs of switch box for achieving the required connectivity. Now the Pfd and Pfu again need not be assigned to every CAB pin as it might be a least possible case or might not be used at all. The required upward connectivity will give a good insight of the Fs for S2 and its structure. In all cases the pin flexibility was associated with only the pin types Vpp, Vout, Iout1 and Iout2, for keeping the discussion simple and is also similar to the pin flexibility implemented in the chip (Refer to Table 4.3).

For 2-HFPAA cluster, the Fs for S1 can be calculated as follows: Consider only three pins of CAB1 Vpp, Vout and Iout1 for the simplicity in discussing this example. Let Pfs=3 assigned to each of them for a connectivity from CAB1 to CAB2 only, as shown in Figure 7.5. The total number of switches for CAB1 to CAB2 connections is, is 3pins * Pfs=3*3= 9 switches/transistors. Now, consider the required upward connectivity (Pfu) for the pins of CAB1 and CAB2 (from sub-cluster1) to CAB3 and CAB4 pins (in sub-cluster2) through S1 (in sub-cluster1) and then through S2 (in the next hierarchical level, as in Figure 7.1. Let us assign Pfu=1 only for simplicity of the example to Vpp, Vout and Iout2 pins of CAB1 and CAB2, i.e they are allowed to connect to only one pin of the CABs in sub-cluster2. So the total number of switches required for this upward connectivity would be, 6 pins * Pfu= 6*1 =6 switches. Similarly let us again assign a Pfd=1 only for simplicity of the example, to Vpp, Vout and Iout2 pins of CAB1 and CAB2, i.e only one incoming connection from other sub-clusters is allowed. So the total number of switches required for this downward connectivity would be, 6 pins * Pfd= 6*1 =6 switches. Having considered and estimated the total number of switches for Pfs=3, Pfu=1 and Pfd=1, the Fs value for the switch box S1= 3pins * Pfs + 6pins * Pfu + 6pins * Pfd =9+6+6= 21 switches. This Fs value equal to 21 would be the same for the switch box S1 in sub-cluster2. Having, estimated the Fs for switch box S1 in sub-cluster1 and sub-cluster2, the value of Fs for the switch-box S2 here is, obtained from the total number of pins considered while estimating Fs for S1, and the corresponding Pfu and Pfd assigned. Hence Fs for S2 is 6 connections upward(from sub-cluster1) + 6 connections downward (from sub-cluster2). Therefore, we can say that the value of Fs for S2 is also=6 pins * Pfu + 6 pins * Pfd = 6+6 = 12 (for the earlier Pfu=Pfd=1).

Hence, the total switch count for a cluster of 2-HFPAA with this specific interconnectivity between the 4CABs is obtained by adding the Fs values obtained above for S1 and S2, which is 21+21+12=54 switches/transistors per cluster (does not include connections from CAB2-CAB1 and CAB4-CAB3). Hence, the complexity in facilitating the required interconnectivity between CABs, can be related to the structure or the number of switches in the switch boxes, which can be estimated from Pfs, Pfu, Pfd and Fs.

The estimate of the total number of switches required per cluster for a 2-HFPAA, for a specific routability can be compared to the switch count estimates for a k-HFPAA to facilitate the same routability. For example, the same routability of the 2-HFPAA discussed in the above example, can be achieved with a 4-HFPAA cluster and a single switch box with Fs = 30 switches/transistors. So, this shows that for implementing the same routability a 4-HFPAA would be more appropriate than a 2-HFPAA, for reducing the total number of switches required. Also, a complete connectivity between the same 3 pins of the four CABS in 4-HFPAA can be achieved with a Fs=108 switches (if the pin flexibility Pfs assigned to each of the three pins of four CABs considered is 9). The estimate of the switch counts also gives an insight of the involved switch capacitances and ON resistances that can affect the analogue signal behaviour.

Observing the large number of transistors required for implementing the switch boxes for a k-HFPAA for a specific interconnectivity, the effect of switch ON resistances and parasitic capacitances on signal transitions is suspected. This effect might be worse, when considering a signal transition from the lowest level to the highest hierarchical level available in a k-HFPAA. This might also restrict scaling of the switch transistors within a minimum and maximum size, while implementing switch boxes of larger Fs (eg:- Fs >50). Analogue circuit simulation was done in Cadence, for testing this suspected

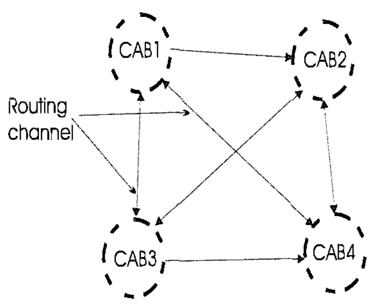


FIGURE 7.5: A Graph representation of the cluster interconnectivity

sub-cluster size	Switch	n	Pfs	Pfd	Pfu	Fs
2	S1	7	3	6	6	51
2	S2	14				12

TABLE 7.3: Switch Flexibility for 2-HFPAA

parasitic effect on signal behaviour during signal transitions from the lowest to the highest level in a k-HFPAA.

Circuit simulation results have showed that the value of Fs is not likely to be limited by its parasitic capacitances, as for switch sizes varying from 1μm to 10μm the total capacitances are only in the range of (10fF-50fF) and the time constant (RC) contributed by these switches again is negligible. All switches are NMOS switches and the ON resistances was observed to be in the range of (300Ω-900Ω) with varying input common mode voltages, which is negligible compared to the high output impedance contributed by the cascoded output stages of our DDACCII CAB. The two limitations are that

- the number of switches is limited by the area of the switches which is dependent primarily on the programming technology (much more chip area is needed for the programming registers and logic than for the switches).
- the possibility of the inbuilt offset increasing dangerously through the hierarchical levels, although this may be controlled by the offset control input available to our CABs.

The circuit set used for our experiments represents basic building blocks with our DDACCII CAB. It also includes basic circuits not larger systems. It is therefore applicable only at lower levels of the hierarchy. Application of our methodology to higher levels would require a circuit set based on applications of these building blocks, eg:- Analogue Neural Networks (ANNs), higher order filters or similar complex analogue systems. The choice of a suitable reference circuit set is an important and non-trivial issue for any attempt to investigate interconnectivity requirements.

7.6 Visualisation of the interconnection strategies

A visual analysis of the interconnection complexity involving the densely interconnected CABs (in a sub-cluster/cluster according to the user specified connectivity rules), was required to consider various hierarchical interconnection strategies. This visual aid would also help in verifying and inspecting these interconnection strategies along with the earlier analysis on estimating the effect of choosing various cluster sizes on the Fs value of the corresponding switch boxes. Use of existing custom IC design tool environment like Cadence, seemed to be inappropriate for such an interconnectivity analysis. Also such an analysis using the schematic feature of these tools would involve lot of time in manually or automatically create a complex interconnection strategy. For such an interconnectivity analysis, the behavioural/circuit simulation was also not significant here. Hence, a tool was developed for visualisation of hierarchical interconnection strategies for our HFPAA, which is discussed here.

The backbone of the tool exploits the recursive nature in building k-HFPAA architecture. The methodology used in our tool is shown in Figure 7.6. A clustering based technique is adopted for grouping CABs, sub-clusters and clusters according to the requirement of building k-HFPAA. The tool was developed using STL in C++, where data structures play an important role in the representation of CABs, sub-clusters, clusters and switch blocks containing switches used for the user-defined connectivity. At the end of the clustering and updating of the final data structure representing k-HFPAA, it is fed into our graph translator tool. The tool translates high-level data structure of the k-HFPAA into a Graph Visualisation grammar syntax of an existing tool called `dot/dotty` of the `Graphviz` package. Graphviz is used for drawing directed graphs and is very efficient in hierarchical visualisation of directed graphs using splines to represent edges, [35]. The translation of the k-HFPAA into the graph language grammar, exploits the hierarchical nature and hence descends hierarchically from the top-level of clusters to the bottom level of the CABs. Hence, the translation uses the opposite procedure of the clustering technique used while building the k-HFPAA.

The tool automatically assigns the required number of switches in the corresponding switch boxes of the sub-cluster/cluster, based on the user defined connectivity strategy of interest. The connectivity between CABs, sub-clusters and clusters of different hierarchies can be specified during run-time, which can be visualised later. Generic rules for the required interconnectivity between CABs, sub-clusters and clusters can also be used as an input to the tool. This interconnectivity rules file should specify the allowed connectivity between the source type pins and destination type pins, which can be applied as a generic rule for interconnectivity for a group of sub-clusters/clusters. Hence, the required number of switches with appropriate switch numbers and identification is automatically generated to support the user-defined connectivity. Thus, the estimates of Pfs, Pfu, Pfd, and Fs can be visually analysed.

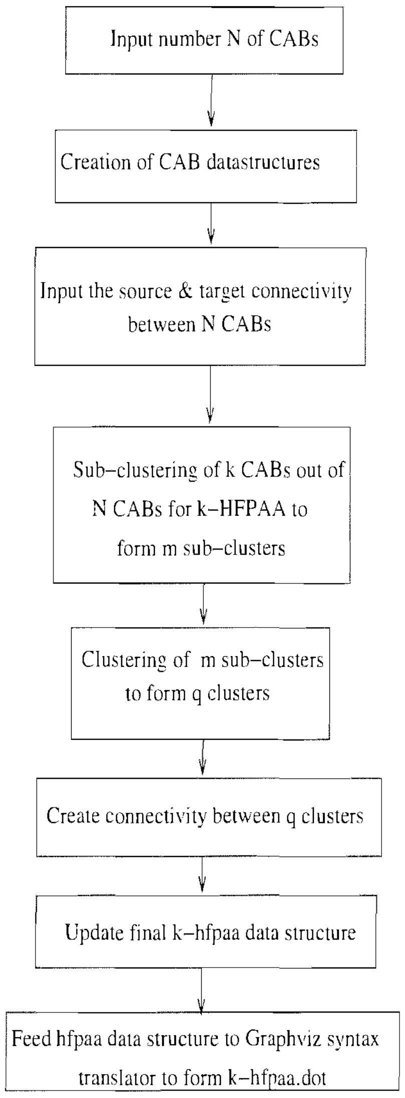

Input number N of CABs
Creation of CAB datastructures
Input the source & target connectivity between N CABs
Sub-clustering of k CABs out of N CABs for k-HFPAA to form m sub-clusters
Clustering of m sub-clusters to form q clusters
Create connectivity between q clusters
Update final k-hfpaa data structure
Feed hfpaa data structure to Graphviz syntax translator to form k-hfpaa.dot

FIGURE 7.6: Methodology adopted in our tool

A few of the simulated interconnection strategies are shown here. Figure 7.7 shows the lowest level, which is our CAB, with the corresponding ports assigned to the CAB. Figure 7.8 shows the next level in the hierarchy, which is a sub-cluster of 2-HFPAA, formed by two CABs and the user-defined interconnectivity between the CABs through the switches in the switch block. Figure 7.9 shows the simulated cluster strategy of a 2-HFPAA, with direction of connectivity as in Figure 7.5 and is similar to the topology of the fabricated chip. From the figure the value of Fs for S1 (Fs=8) and S2 (Fs=24) for the user-defined interconnectivity can be visually obtained by looking at the maximum value of the switch number in the corresponding switch boxes generated. The direction of connectivity from each source pin to the destination pin can also be visually confirmed. Similarly Figure 7.10 shows the results for the interconnection strategy obtained between two clusters with a user-defined connectivity. The connectivity between the two clusters at the next level of the hierarchy through a S3 switch box is automatically generated,

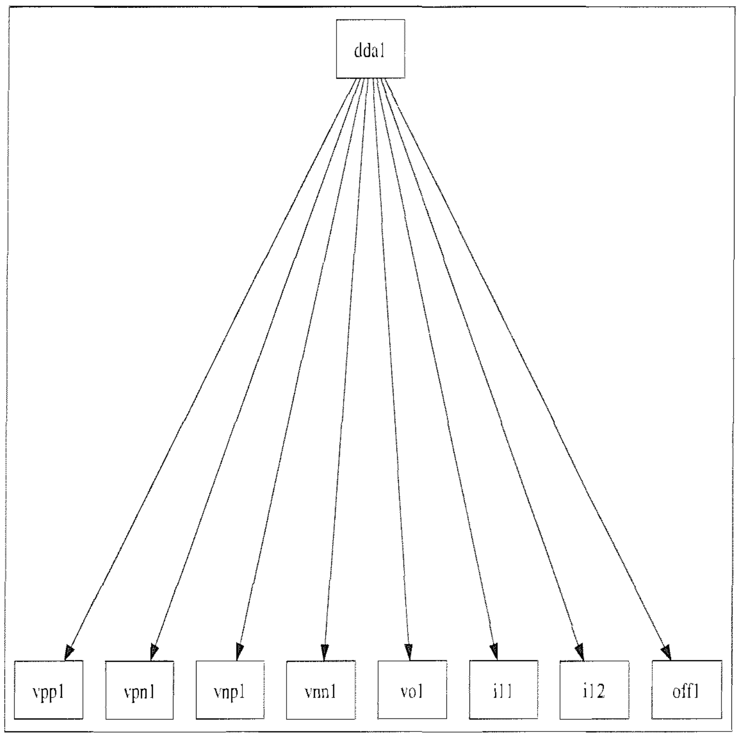


FIGURE 7.7: Our DDACCII CAB

as shown in Figure 7.1. Here, this connectivity through S3 was restricted through 8 switches at a time and hence these switch blocks are found dispersed outside the two clusters (Cluster1 and Cluster2). But it is these small blocks of 8 switches, which are likely to form a single S3 switch box structure, connecting Cluster1 and Cluster2.

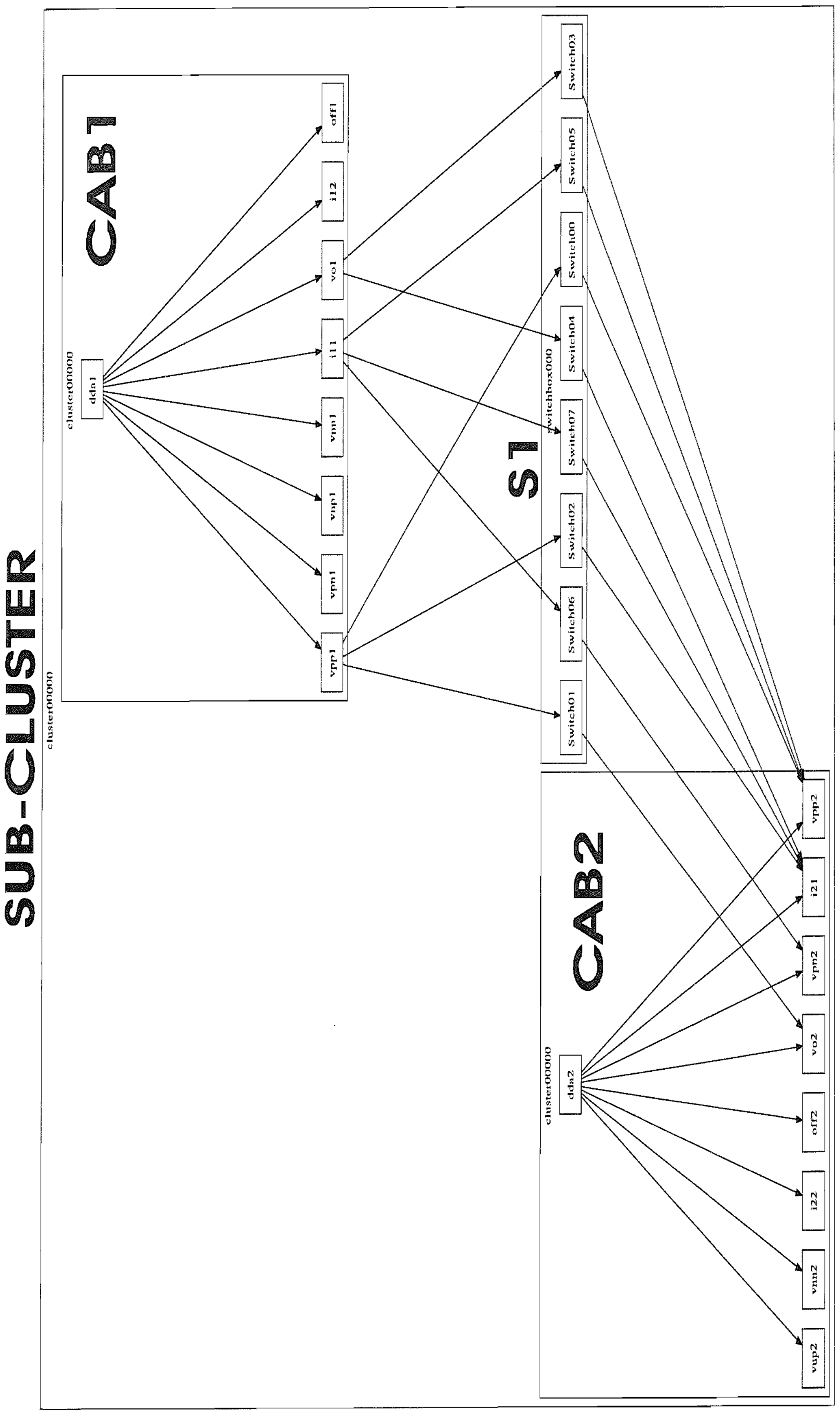


FIGURE 7.8: A sub-cluster containing 2 CABs

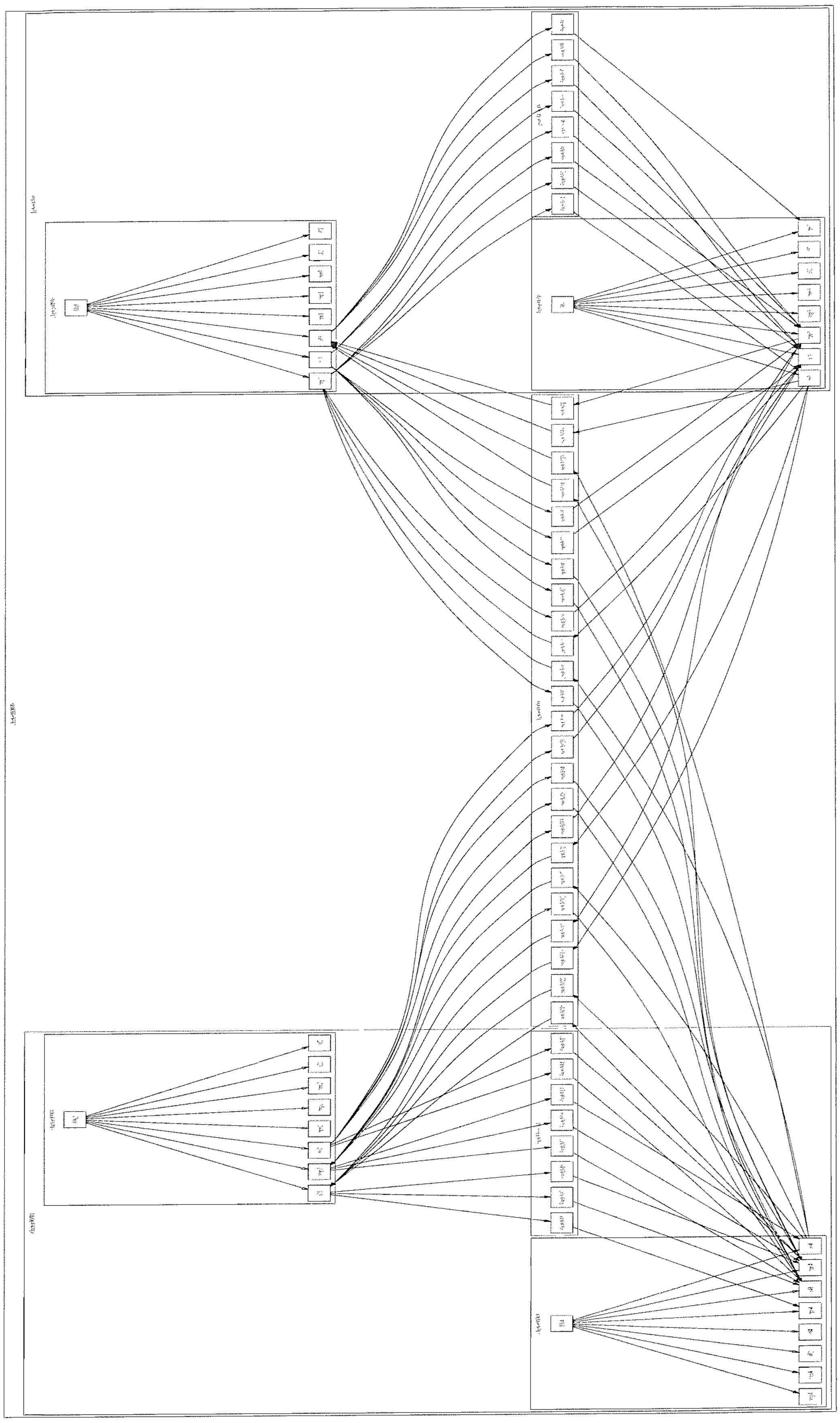


FIGURE 7.9: A cluster containing 2 sub-clusters

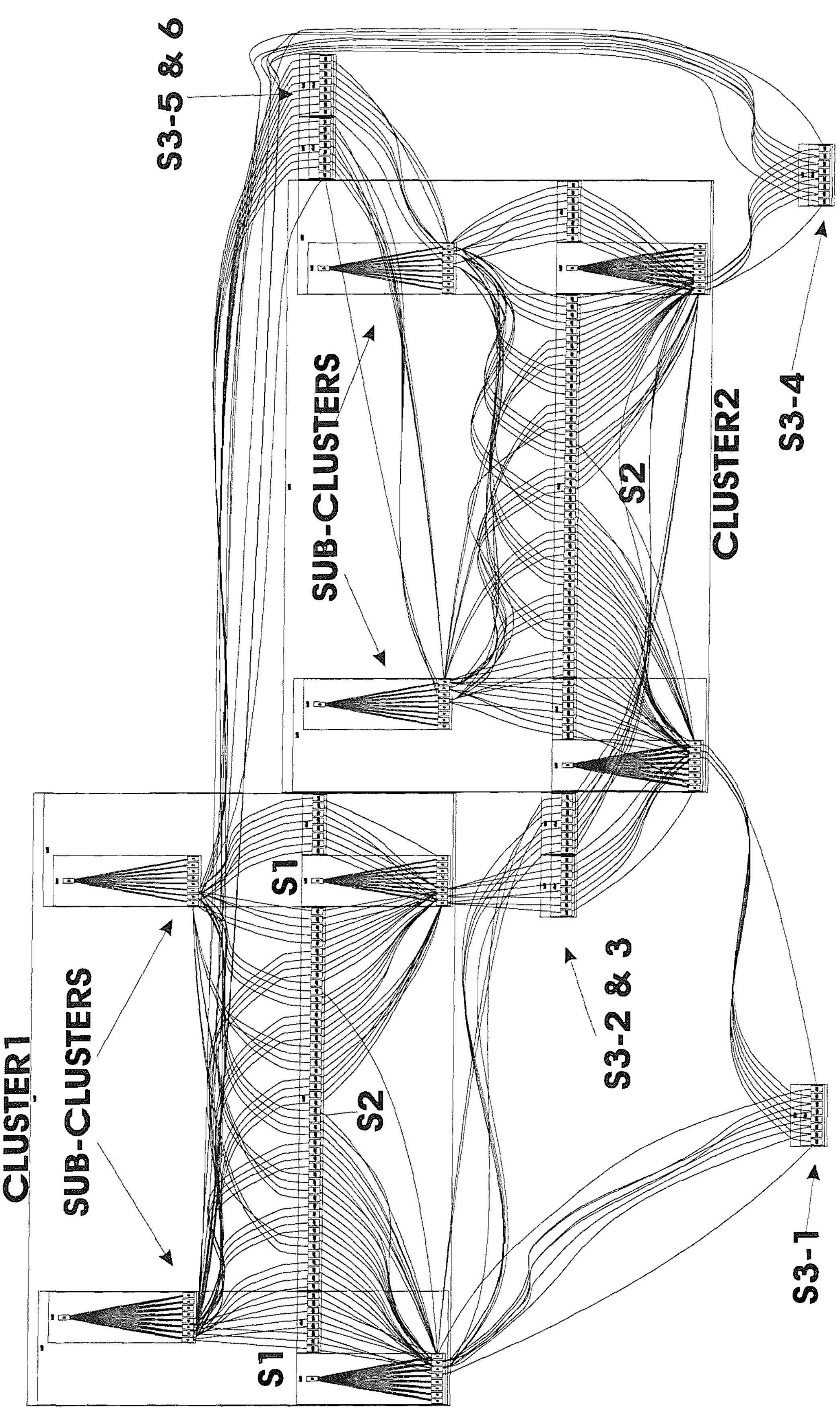


FIGURE 7.10: Simulated Interconnectivity between two clusters

7.7 Computation Time and complexity of NPGCL

The computation time for performing NPGCL on an input set of 33 circuits and a group size of 4 took 0.21 seconds, using a Intel Pentium(R)4 CPU,3.2GHz,1 GB RAM and Linux OS. Empirically the computation time of our NPGCL was observed to follow a power law of the form kn^4, where k is a constant that denotes group size and n is the input set size (Figure 7.11). This time is significantly dominated by the time for writing the generated combinations by the NPGCL to a file on disk. This limitation of the algorithm in computation time for larger input data sets could be improved, by exploiting the recursive nature and the similarity in tree patterns to derive the desired subsequent NPGCL trees from the first tree. In an effort to evaluate the complexity of the NPGCL algorithm, it was found to be similar to a well known classical problem in computer science theory called the "Knapsack Problem". The knapsack problem is a NP-complete (Non-deterministic Polynomial-time complete)problem in combinatorics, complexity theory, cryptography, and applied mathematics. Given a set of items, each with a cost and a value, determine the number of each item to include in a collection so that the total cost is less than some given cost and the total value is as large as possible. The knapsack problem is often solved using dynamic programming, though no polynomial-time algorithm is known for the general problem. The techniques used to solve such problems are found to have computational times growing exponentially with input size, the cost and value of each item.

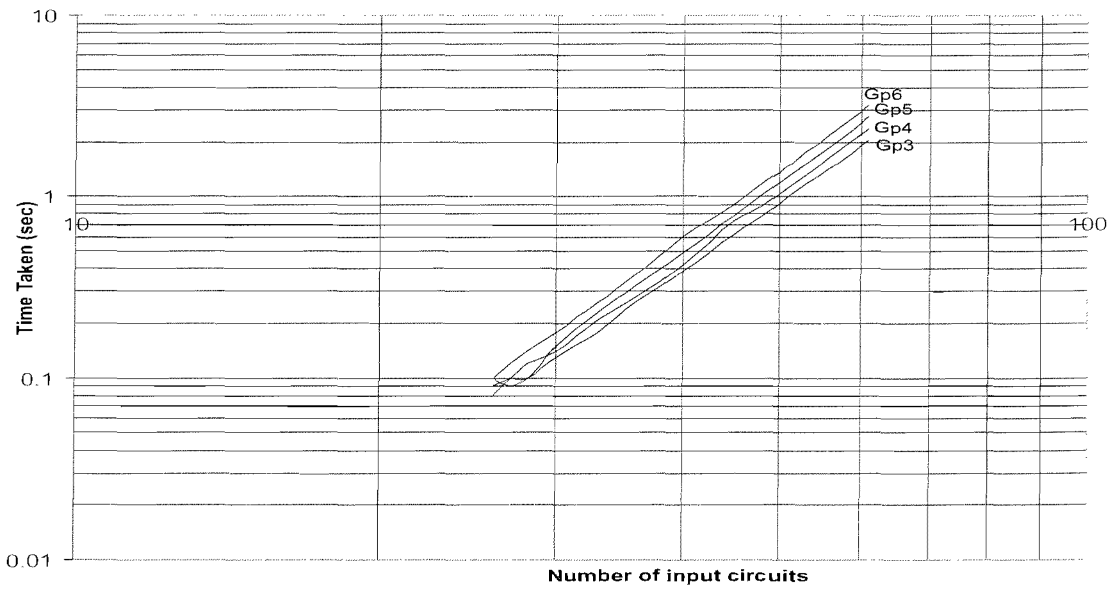


FIGURE 7.11: Power law form for the computational time of NPGCL

7.8 Conclusions

A methodology has been established to analyse the interconnection complexity and resource utilisation in hierarchical interconnection architectures for field programmable analogue circuits. This was applied on our HFPAA architecture for estimating the flexibility of its interconnection structures using switch flexibility (Fs), pin flexibility (Pf including Pfs, Pfd & Pfu) and number of terminals n. This methodology has also shown the flexibility of our DDACCII CAB granularity, our HFPAA architecture and its implications on interconnection complexities. A statistical comparison with other CAB implementations to statistically strengthen the argument on CAB flexibility was considered. But this is avoided in the discussion here because such a comparison on the resource usage, number of external components, etc. could be done only on our illustrative single CAB dominated set of analogue circuits. To apply the same analysis for other CABs, for comparison would require generating a set of circuits with the same functionality, but this would not give an objective comparison. To make such a comparison in an objective manner requires a defined set of functionality and an optimal set of circuits for each CAB design which provides these functions. The definition of a set or sets of analogue circuits for evaluating FPAA interconnection strategies is clearly necessary. This is a complex task very much dependant on the application areas envisaged.

Chapter 8

Summary, Conclusions and Future Work

This chapter includes the summary of the research work till now, along with the conclusions and future work.

8.1 Summary

The project started with the an investigation on the availability of FPAAs developed over the past several years and their significance in reducing the design cycles for analogue circuit design. This included a comparison with FPGAs, which have proved to be a significant development in modern VLSI design, reducing the design cycles and hence enabling rapid prototyping and testability of digital integrated circuits. The development of FPAAs has been very slow compared to FPGAs. The reason is that of the greater complexity to be considered in designing analogue circuits when compared to digital circuits. However, the significance of analogue circuits interacting between the real world analogue signals and digital circuits, has been realised by the industry and researchers. Hence the development of the FPAA, as a result of the efforts for faster design and test methodologies for analogue circuit design.

A survey of all the available FPAAs developed till now, by researchers and the industry has been undertaken. The FPAA architectures considered have been discussed in Chapter 2. These FPAAs were evaluated on the basis of applications covered and the ability to handle with good precision, the complexity of analogue signals with respect to frequency, signal levels, time and parasitics. This evaluation and comparison between the available FPAA architectures led to some interesting conclusions, which helped to clearly define the aims and objectives of this project. It was found that the technology used and the target applications covered were the prime concern for these FPAAs that have been developed. However, none of these papers on FPAAs explain how effectively these FPAAs can reduce the design cycle in analogue circuit design. Critically evaluating this aspect of the developed FPAAs, the following was concluded:

- These FPAAs do not explain the suitability of the chosen CAB for the target application, with respect to:
    - the amount of flexibility achieved in configuring the CAB for the target application.
    - the number of CABs and other routing resources required for configuring an application or a range of applications.
    - the flexibility achieved in switching between possible configurations with minimum number of external passive components.
    - trade-offs between the required flexibility and the tolerable complexity of the CAB.
- These FPAAs do not explain the suitability of the chosen CAB array architecture with respect to:
    - the amount of routability achieved between CABs.
    - the tolerable routing complexities between CABs for the maximum routability between CABs required to support more configurations.
    - the trade-offs between maximum routability/performance and the total number of allowed routing resources and its effect on chip area.
    - the possibility and feasibility of using other possible architectures(developed for FPGAs) for the FPAAs, other than the simple array-based architecture followed by all the FPAAs.
- They do not focus on, or explain, the suitability of the chosen technology for the FPAA, in reducing the total number of passive components. This is a very significant factor the developed FPAAs have not addressed or may have ignored. This is an issue which cannot be compromised, as they are required for the specific analogue behaviour and is also related to:
    - the total number of allowed wired input/output signal pads on the FPAA.
    - the allowed internal passive components, if any, as they affect the analogue behaviour with variation in process gradients.
    - the complexities to be handled in the interconnect architecture.
    - the amount of the FPAA chip area that can be allocated for any internal passive components, as they tend to occupy large area.

The above factors and the relationship between them, may be summarised as the need for finding the possible topologies for a programmable analogue circuit, that could speedup the design cycle of an analogue circuit. Hence, this research was titled as, *"Topologies for Programmable Analogue Circuits"*. As a part of this research it was decided to primarily focus on the:

- strategies that affect the choice of a CAB.
- development of a suitable CAB that can:
  - reduce the CAB dependency on passive components while achieving the target analogue behaviour
  - be flexible in switching between configurations.
  - is quite universal enough to achieve voltage and current mode operations.
- strategies that affect the choice of an appropriate interconnect architecture.
- development of a suitable interconnect architecture for arranging the CABs, that can:
  - incorporate maximum routability between CABs
  - minimise the complexity of routing but still achieve a considerable range of target applications.
- design and fabricate a test FPAA chip, that includes the designed CAB and interconnect architecture, that can explore:-
  - the effectiveness of the proposed CAB design.
  - the complexities of designing the FPAA architecture.
  - the use of a hierarchical architecture.
  - the complexities involved in planning the chip and implementing the layout of the chip for fabrication in a particular CMOS technology.
- range of applications that the FPAA would support was focused primarily on Instrumentation and Data Acquisition Systems, with a DC to at least 1-MHz frequency of operation.

As per the project plan and objectives of this project, an extensive analysis was done, on the available building blocks in analogue circuit design. The analysis suggested current conveyors to be an attractive building block, due to the compact nature in itself, the continuous-mode of operation, the flexibility in handling signals in current mode and also the high-bandwidth they provide. Current conveyors also require fewer external passive components. They also seemed to address the complexities in routing, resource utilisation and interconnect architecture. However, a CAB that could also support voltage-mode of operation along with additional features of applying differential input signals, was also of interest.

It was during the quest for a versatile CAB again, the research interest was directed towards a versatile building block, called the differential difference amplifier. Although not very well known in analogue circuit design, papers on the DDA and its applications indicated that, as a versatile building block in signal processing it was very attractive. The DDA, met all the defined requirements of the CAB in this project, except for the current-mode of operation. A considerable amount of time was spent on verifying the suitability of the DDA as a CAB, through Spice simulations with MOSFET transistor models for 0.5μm process.

It was found that, with additional circuitry in the existing DDA design, the current-mode of operation could also be achieved. The DDA block along with the integration of the current conveyor, was very appealing for satisfying the role of an ideal CAB, which could be arranged in a suitable architecture to form the target FPAA. A feasibility study of using the DDA as the CAB through spice simulations was done. The Spice simulations verified the features of the DDA and the DDA based analogue applications.

Foreseeing the fabrication of a FPAA chip and with the available support in funding the project, it was decided to choose the AMS CMOS 0.6μm CUP process, as the target technology. The Cadence 4.4.5, supporting the AMS CMOS 0.6μm CUP technology, was used as the tool for circuit design (Analogue Artist)and layout (Virtuoso) of the target chip. The plan for the design of the CAB and the chip was done, because the layout of the chip had to be completed well before the deadline for the AMS CMOS 0.6μm CUP process fabrication run scheduled by CMP, France.

The design of the DDACCII CAB was implemented for CMOS 0.6μm CUP process. The CAB was tested for the performance characteristics, although it was difficult to define appropriate measures to assess the performance due to the complexities in the number of parameters involved with the number of input ports. The integration of the current conveyor into the designed DDA CAB to form the DDACCII CAB, along with the design of an appropriate output stage for the block was a challenge. A complex analysis and design of the class AB stage using a compensation scheme, was done for the CAB. This analysis is comprehensive illustrating this efficient class AB output stage, as there is no paper or proper documentation about this compensation scheme useful for analogue circuits. The designed DDACCII CAB was tested and verified for the target applications defined in the project. These involved static and dynamic switching based applications as well.

The design of a suitable architecture, interconnecting the designed DDACCII CABs, with appropriate resources (switches), was a difficult task. It was observed that all the FPAA architectures developed till now followed a two-dimensional array concept. The array based architecture did not seem to be a very attractive concept, foreseeing the building of larger analogue systems with the FPAA, having a bigger array of CABs. Based on the investigation on the architectures developed for FPGAs, that efficiently minimises the routing complexities with maximum routability between the logic building blocks, the hierarchical architecture based FPGAs were of interest in this research. The routing algorithms developed for these hierarchical based architectures and its compatibility for programmable analogue blocks was also of interest in this research as a possible future work. However, it was obvious that the developed concepts for hierarchical FPGA architectures could not be applied directly for the FPAA. But due to the promising minimum amount of routing complexities and the efficient manner of connecting building blocks in the hierarchical manner, as in many investigated papers, it was decided to implement a similar concept with the DDACCII CABs.

The hierarchical based architecture was implemented connecting the DDACCII CABs to form clusters using appropriate number of switches in the hierarchy. However, a few modifications where made in the manner of representing the hierarchical structure. A graph based representation was found to be more appropriate for the hierarchical FPAAs than the normal tree based representation used for the hierarchical FPGAs. This graph based structure would also help in developing routing algorithms or an analogue router for the target FPAA, as a future work. Even though the research was interested in building a larger hierarchy of clusters for the FPAA, it was decided to implement only a cluster for the target FPAA chip. This decision was made with regard to the available funds to fabricate a chip of constrained area. Hence it was decided that a cluster (four CABs and three switch boxes), two double pole double throw switches (for dynamic switching) and a test CAB (backup CAB alone for testing purposes), is sufficient for the target chip.

As per the plan of the chip, the design of the chip was implemented at a transistor schematic level containing the blocks as planned and the appropriate input/output/bidirectional pins. The pin assignment and arrangement was done foreseeing the layout of the chip that could suit the JLCC68 pin package. This schematic of the chip was tested for applications and verified for functionality. The clocking scheme did not seem to be very complicated. A simple clock tree structure, with three branches, each buffered with clock buffers was implemented, for clocking the total 72-bit register (arranged as 24-bit x 3 rows).

The layout of the chip was also initially planned to obtain rough estimates about the complexity in implementing the routing with the available metal layers in CMOS 0.6μm CUP process. The planning of the layout through rough sketches was done. This is also helped in roughly assigning the placement of the main routing channels and also the rough coordinates for the placement of the other blocks. The top-down planning of the chip was followed by a bottom-up execution of the layout. The layout of each block was implemented and verified for functionality using the extracted net list and parasitics (capacitances and resistances) from the layout. Gradually the layout of the whole chip was completed satisfying all the DRC and LVS checks and other miscellaneous layout considerations and rules with respect to process constraints (coverage, antenna, wide metal slits). A considerable amount of time was spend for implementing the layout of the chip. The layout of the chip was also verified by simulating the extracted net list along with involved parasitics. Finally, the chip was fabricated by CMP, France.

The fabricated chip was put into testing for measuring and observing the realities of its performance, when compared to the simulation results during the design phase of the FPAA. The test DDACCII CAB block was working as expected. But there seemed to be a problem in configuring the cluster inside. The chip was drawing high currents from the power supply, which was later found due to the shorted inverted outputs of the configuration DFC flip-flops to ground. This behaviour was not spotted in the simulations earlier, because the currents drawn from the power supplies were not explicitly checked. So, the solution was to reduce the power supply range to 2.5V, so that less current is drawn while clearing the configuration register. Once the configuration register is cleared, the power supply voltage could then be raised to 5V. Hence, the four CABs in the cluster could still be used, as precaution was taken to bring out all the inputs/outputs of the CABs as separate pins on the chip. These CABs were connected for testing different applications for a comparison with the earlier obtained simulation results.

On chip tested applications included static and dynamic switching applications. The observed functional behaviour was found to be similar to that of the simulation results. But applications like modulation and multiplication could not be tested effectively with an external MOSFET transistor acting as a varying resistance. Hence, the MOSFET transistor should have been included in the chip to able to test the modulation and multiplier applications of our CAB. The effect of the high pad capacitances (4pF/pad) added to the internal compensation capacitance on the output responses of the static and dynamic switching applications was observed. This effect would worsen the switching speed if external load capacitances are also desired for the circuit connections. Hence, faster response will be achieved by keeping the required passive component connections internal (based on simulated results) to the CAB (locally accessible) or to the chip (globally accessible). A detailed noise analysis measurement was not done. The noise was measured only for the CAB configured in voltage mode, for a comparison with the simulated noise analysis results. The measured noise was found to be similar to the simulated response.

The work to date had established a design methodology and demonstrated the feasibility of our approach.The functional flexibility of our CAB was accounted in the research till then. But the suitability of our CAB in a hierarchical architecture with maximum routability between CABs, depends entirely on the flexibility of the interconnection architecture. This flexibility was found to be directly related to the CAB granularity and the switching structures used for the expected interconnectivity between CABs. Hence, the analysis of the flexibility of the HFPAA interconnection architecture was required as the next step in this research. On the basis of investigation, it was found that there has been relatively little work on the interconnection strategies for FPAAs. There has been prior work on the interconnection complexity for FPGAs and HFPGAs based on a deterministic method called the Rent's Rule. But similar methods may not be applicable for analysing the interconnection complexity for FPAA architectures, due to the absence of any established exhaustive set of analogue benchmark circuits. Also, a similar statistical homogeneity as in digital circuits, may not be definable in analogue circuits at a transistor level. Existing circuit partitioning and gate level placement algorithms for digital circuits may not be useful for the permitted granularity in analogue circuits. Hence, a new methodology was established for interconnectivity analysis of our hierarchical interconnection architecture, not relying on the quality of partitioning/routing algorithms as in digital circuits.

The methodology for interconnectivity analysis, involved the:-

- definition of 33 analogue circuits, which can be implemented by the DDACCII CAB, as a set of benchmark circuits. This was the starting point, as for any attempt to analyse interconnectivity has to be a set of realistic circuits, which will perform the type of signal processing for which the FPAA may be used.
- development of a new algorithm called *NPGCL (Non-Permuting Grouped Combinations Listing)* Algorithm, which uses the above set of 33 circuits as an input set for interconnectivity analysis. NPGCL was developed for generating all combinations of circuits, which will fit in a single cluster. For those combinations that will fit, the number of terminals (n) may be identified. The same algorithm was used to explore the consequences of changing the cluster size.

The flexibility of the interconnection architecture was characterised on the basis of the flexibility of the switch blocks (Fs) in the sub-cluster/cluster. The flexibility Fs, of the switch block is defined as the amount of interconnectivity it can support between the CABs in a sub-cluster/cluster. Hence, Fs of a switch block reflects the required switch structure supported with appropriate programming technology, to facilitate the expected interconnectivity. This Fs was again characterised using the pin flexibility Pf of the CAB pins. The pin flexibility Pf of the CAB pins is defined as the number of other CAB pins each pin can be connected to locally and hierarchically. This Pf includes Pfs (connectivity to the CAB pins of the same sub-cluster), Pfd (downward pin flexibility, incoming connections to A from other sub-clusters) and connectivity to other sub-clusters through S2 is Pfu (upward pin flexibility). The NPGCL was applied to get more information on the internal switch structure of a sub-cluster or cluster for different hierarchical interconnection strategies. The input set applied to the NPGCL here were the pins of the CABs for which the interconnectivity was analysed. Hence, the

NPGCL would compute the amount of interconnectivity between the pins of the CABs, by generating the combinations of these pins. This would give an insight on the required Pfs, Pfu and Pfd for defining the specific interconnectivity for calculating the Fs. This analysis resulted in obtaining large values of Fs for implementing a high interconnectivity between CABs/sub-clusters/clusters, although a majority of these interconnections may not be a useful.

As a result of the large values estimated for Fs for different hierarchical interconnection strategies, the effect of switch transistor ON resistances and parasitic capacitances on signal transitions was suspected. But, circuit simulation results showed that the value of Fs was not limited by its parasitic capacitances, ON resistances for varying switch sizes. The two limitations were:

- the number of switches is only limited by the area of the switches and the area consumed by the programming technology used (SRAM, EPROM, Shift-Registers etc).
- the possibility of the inbuilt offset increasing dangerously through the hierarchical levels, which can be controlled by the offset control input available to our CABs.

The limitation of our methodology for interconnectivity analysis based on the results indicated that our illustrative input circuit set is biased towards single CAB circuits. Hence, the choice of the input circuit set will affect the shape of the distribution under study (number of terminals, number of grounded connections, number of external connections etc.) and this merits further investigation. The circuit set used for our experiments represents basic building blocks with our DDACCII CAB. It also includes only basic circuits not larger systems. It is therefore applicable only at lower levels of the hierarchy. Application of our methodology to higher levels would require a circuit set based on applications of these building blocks, eg:- Analogue Neural Networks (ANNs), higher order filters or similar complex analogue systems. Hence, the definition and synthesis of a appropriate analogue benchmark circuit set for regressive interconnectivity analysis, requires further investigation and is proposed for future work. This proposed analogue benchmark circuit set could be used with the developed NPGCL algorithm and visualisation tool for a similar interconnectivity analysis implemented in this research project.

8.2 Conclusions

Having summarised the whole research progress, the following are the conclusions of this research project:-

- Reviewing the various available architectures for the FPAAs, a hierarchical architecture is proposed for the FPAA, which has not been explored by any research work for programmable analogue circuits.
- A CAB called the DDACCII, which integrates the features of the DDA and the second generation current conveyor, has been designed to meet the target applications with fewer external passive components.
- the objective to design a CAB, that is less dependent on external components for achieving the desired behaviour.
- A test chip containing the cluster, which defines the granularity of the FPAA, has been fabricated and tested for a comparison of results with simulated results during the design phase.
- A novel methodology was developed for the interconnectivity analysis on the basis of a set of DDACCII CAB based analogue circuits and the developed NPGCL algorithm.
- For the circuit set used for interconnectivity analysis, the number of combinations dominated by single CAB circuits required a relatively large number of terminals for the sub-cluster/cluster.
- The NPGCL analysis of the circuit set applied to 2CAB sub-cluster and 4CAB cluster showed the need for a large number of terminals to the sub-cluster/cluster, to accommodate most of the circuits. This is a consequence of the illustrative circuit set being dominated by single CAB circuits and may not reflect real requirements.

8.3 Future Work

The DDACCII CAB, the hierarchical architecture and methodology of discussing architectures can definitely be extended for research on the following:-

(1) Since the CABs can be operated in current mode as well, these could be used to build Analogue Neural Network (ANNs) which uses current mode signal processing. The project would address the complexity involved related to interconnectivity and the feasibility of such a hardware platform to support Neural computation. This would include speed, area and power related issues.

(2) This research has introduced a novel methodology of discussing interconnectivity analysis on the basis of empirical set of analogue circuits and the NPGCL algorithm. But this methodology clearly has limitation due to the smaller set of analogue benchmark circuits, which are again biased towards single CAB implementations. Hence, an

extensive investigation on deriving an appropriate set of analogue benchmark circuits is a likely topic for future work. An interconnectivity analysis on the basis of this new exhaustive set of analogue benchmark circuits, would also help in statistical comparison of various CAB type implementations. Such a statistical study would strengthen the suitability of certain CAB types for specific applications.

(3) Considering the significance of low power design and the need for an FPAA which can support bigger networks of CABs, the power scalability in the network is of interest. This would address the need for scaling networks like the ANNs for reducing power consumption, by considering the use of pruning theory (cutting/removing of those neural nodes which contributes the least in the hidden layers) , to consider the possibility of using techniques for dynamic voltage scaling and frequency scaling algorithms and also to investigate on any possible circuit techniques which could be incorporated locally within the CAB or globally for the FPAA to save power (like MTCMOS , RBB Reverse Body Biasing , ABB Adaptive Body Biasing,techniques to reduce leakage power). This would be useful to address issues of saving leakage power of a huge network. which could turn out to be greater than the active power, especially in space based applications prone to higher temperatures.

(4) Analogue Synthesis issues to support the interpretation of higher levels of abstraction for analogue systems design and to be able to map the design on FPAA. This would include a cell based synthesis of the target analogue functionality depending on the chosen granularity(could be existing CABs or my DDACCII cab), on the basis of an AHDL (Analogue Hardware Design Language) or a new library developed using VHDL-AMS/VerilogA/SystemC.

References

[1] *Totally reconfigurable analog circuit*. Zetex, 1999.

[2] *ispPAC30 In-System Programmable Analog Circuit*. Lattice Semiconductor Corporation, 2002.

[3] *AN120E04 Datasheet configurable FPAA*. Anadigm, 2003.

[4] *PSoC-Configurable Mixed-Signal Array with On-board Controller*. Cypress Microsystems, 2003.

[5] M. J. Alexander and G. Robins. New performance-driven fpga routing algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15:1505–1517, December 1996.

[6] P. E. Allen and D. R. Holberg. *CMOS Analog Circuit Design*. second edition, Oxford University Press, New York, 2003.

[7] H. A. Alzaher, H. O. Elwan, and M. Ismail. Cmos fully differential second-generation current conveyor. *Electronic Letters*, 36:1095–1096, June 2000.

[8] H. A. Alzaher, H. O. Elwan, and M. Ismail. A cmos fully balanced second-generation current conveyor. *IEEE Trans. Circuits and Systems II*, 50:278–287, June 2003.

[9] H. A. Alzaher and M. Ismail. A cmos full balanced differential difference amplifier and its applications. *IEEE Trans. Circuits and SystemsI*, 48:614–619, June 2001.

[10] D. Anderson and C. Marcjan. A field programmable analog array and its application. In *IEEE Custom Integrated Circuits Conference*, 1997.

[11] R. J. Baker, H. W. Li, and D. E. Boyce. *CMOS Circuit Design, Layout And Simulation*, chapter 20,Current Sources and Sinks. IEEE New York, 1998.

[12] R. J. Baker, H. W. Li, and D. E. Boyce. *CMOS Circuit Design, Layout And Simulation*, chapter 22,Amplifiers. IEEE New York, 1998.

[13] R. J. Baker, H. W. Li, and D. E. Boyce. *CMOS Circuit Design, Layout And Simulation*, chapter 7,CMOS Passive Elements. IEEE New York, 1998.

[14] M. Banu, J. M. Khoury, and Y. Tsividis. Fully differential operational amplifiers with accurate output balancing. *IEEE Journal of Solid-State Circuits*, 23:1410–1414, December 1988.

[15] J. Becker and Y. Manoli. A conitnuous-time field programmable analog array consisting of digitally reconfigurable gm-cells. In *IEEE International Symposium on Circuits And Systems (ISCAS)*, 2004.

[16] A. Bratt and I. Macbeth. Dpad2 - a field programmable analog array. *Kluwer Analog Integrated Circuits and Signal Processing - Special Issue on Field Programmable Analog Arrays*, 17:67–89, sep 1998.

[17] S. Brown, J. Rose, and Z. G. Vranesic. A detailed router for field-programmable gate arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11:620–628, May 1992.

[18] K. Bult and H. Wallinga. A class of analog cmos circuits based on the square-law characteristic of an mos transistor in saturation. *IEEE Journal of Solid-State Circuits*, 22:357–365, June 1987.

[19] S. Chang, B. Hayes-Gill, and C. Paull. Implementation of a multifunction signal detection block for a field programmable analogue array. In *5th EuroChip Workshop*, volume 17–19, pages 226–231. Dresden Germany, October 1994.

[20] P. Christie. The interpretation and application of rent's rule. *IEEE Transactions on VLSI Systems*, 8,(6):639–648, 2000.

[21] U. Cilingiroglu and S. K. Hoon. An accurate self-bias threshold voltage extractor using differential difference feedback amplifier. *IEEE Symposium on Circuits and Systems*, 5:209–212, May 2000.

[22] D. Clein and G. Shimokura. *CMOS IC LAYOUT: Concepts, Methodologies, and Tools*, chapter 6,Advanced Techniques for Building-Block Interconnect Layout Design. U.S.A Newnes, 2000.

[23] Z. Czarnul. A new compensated integrator structure with differential difference amplifier and its application to high frequency mosfet-c filter design. In *European Conference on Circuit Theory and Design*, pages 132–136, September 1989.

[24] J. F. Duque-Carrillo, G. Torelli, R. Perez-Aloe, J. M. Valverde, and F. Maloberti. A class of fully-differential basic building blocks based on unity-gain difference feedback. *IEEE International Symposium on Circuits and Systems*, 3:2245–2248, April 1995.

[25] J. F. Duque-Carrillo, G. Torelli, R. Perez-Aloe, J. M. Valverde, and F. Maloberti. Fully differential basic building blocks based on fully differential difference amplifiers with unity-gain difference feedback. *IEEE Trans. Circuits and Systems I*, 42:190–192, March 1995.

[26] R. Timothy Edwards, K. Strohben, and S.E. Jaskulek. A mixed-signal programmable array using antifuse interconnect. In *The Ninth NASA Symposium on VLSI Design*, 2000.

[27] A. A. El-Adawy, A. M. Soliman, and H. O. Elwan. A differential current-conveyor based buffer for high-bandwidth, low-signal applications. *The 6th IEEE International Conference on Electronics, Circuits and Systems*, 2:903–906, September 1999.

[28] A. A. El-Adawy, A. M. Soliman, and H. O. Elwan. A novel fully-differential current conveyor and applications for analog vlsi. *IEEE Trans. Circuits and Systems-II:Analog and Digital Signal Processing*, 47:306–313, April 2000.

[29] H. Elwan and M. Ismail. Cmos low noise class ab buffer. In *Electronic Letters*, volume 35, October 1999.

[30] A. Fabre and M. Alami. Universal current mode biquad implemented from second generation current conveyors. *IEEE Trans. on Circuits and SystemsI*, 42,(7):383–385, July 1995.

[31] E. G. Friedman. Clock distribution in general vlsi circuits. *IEEE International Symposium on Circuits and Systems*, 3:1475–1478, May 1993.

[32] S. Ganesan and R. Vemuri. Faar: A router for field-programmable analog arrays. *Twelfth International Conference On VLSI Design*, pages 556–563, January 1999.

[33] S. Ganesan and R. Vemuri. A methodology for rapid prototyping of analog systems. *International Conference On Computer Design*, pages 482–488, October 1999.

[34] A. J. Gano and J. E. Franca. New fully differential variable gain instrumentation amplifier based on a dda topology. *Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference*, 1:60–64, May 1999.

[35] Emden R. Gansner, E. Koutsofios, Stephen C. North, and Kiem-Phong Vo. A technique for drawing directed graphs. *IEEE Trans. on Software Engineering*, 19,(3):214–230, March 1993.

[36] V. C. Gaudet and P. G. Gulak. Cmos implementation of a current-conveyor based field programmable analog array. In *IEEE Custom Integrated Circuits Conference*, 1998.

[37] T. Georgantas, Y. Papananos, and Y. Tsividis. A comparitive study of five integrator structures for monolithic continuous-time filters. *IEEE International Symposium on Circuits and Systems*, pages 1259–1262, May 1993.

[38] L. R. Goke and G. J. Lipovski. Banyan networks for partitioning multiprocessing system. In *1st Annual Computer Architecture Conference*, pages 21–28, 1973.

[39] P. R. Gray, P. R. Hurst, S. H. Lewis, and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits.* fourth edition, John Wiley and Sons, Inc., New York, 2001.

[40] H. F. Hamed, A. El-Gaafary, and M. S. A. El-Hakeem. A new differential current conveyor and its application as a four quadrant multiplier. *The 8th IEEE International Conference on Electronics, Circuits and Systems*, 2:569–572, September 2001.

[41] B. Hochet, V. Peiris, and M. J. Ceclercq. Implementation of a learning kohonen neuron based on a new multilevel storage technique. *IEEE Journ. Soild-State Circuits*, SC–26,(3):262–267, 1991.

[42] S. C. Huang and M. Ismail. Novel full-integrated active filters using the cmos differential difference amplifier. *Proceedings of the 32nd Midwest Symposium on Circuits and Systems*, 1:173–176, August 1989.

[43] S. C. Huang and M. Ismail. Cmos multiplier design using the differential difference amplifier. *Proceedings of the 36nd Midwest Symposium on Circuits and Systems*, 2:1366–1368, August 1993.

[44] S. C. Huang and M. Ismail. A cmos differential difference amplifier with rail-to-rail fully-differential outputs. *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, 2:780–781, August 1994.

[45] S. C. Huang and M. Ismail. Design of a cmos differential difference amplifier and its applications in a/d and d/a converters. In *IEEE Asia-Pacific Conference on Circuits and Systems*, pages 478–483, December 1994.

[46] S. C. Huang, M. Ismail, and S. R. Zarabadi. A wide range differential difference amplifier:a basic block for analog signal processing in mos technology. *IEEE Trans. Circuits and SystemsII*, 40:289–301, May 1993.

[47] J. H. Huijsing. Instrumentation amplifiers:a comparitive study on behalf of monolithic integration. *IEEE Trans. Instrumentation and Measurement*, 25:227–231, September 1976.

[48] C. C. Hung, M. Ismail, K. Halonen, and V. Porra. Low-voltage rail-to-rail cmos differential difference amplifier. *IEEE International Symposium on Circuits and Systems*, 1:145–148, June 1997.

[49] D. A John and K. Martin. *Analog Integrated Circuit Design.* John Wiley and Sons, Inc., New York, 1997.

[50] D. A John and K. Martin. *Analog Integrated Circuit Design*, chapter 6, Advanced Current Mirrors. John Wiley and Sons, Inc., New York, 1997.

[51] M. Kayal and Z. Randjelovic. Auto-zero differential difference amplifier. *Electron Letters*, 36:695–696, April 2000.

[52] N. I. Khachab and M. Ismail. Mos multiplier/divider cell for analogue vlsi. *Electronic Letters*, 25,(23):1550–1552, 1989.

[53] E. Klumpernick, E. V. D. Zwan, and E. Seevinck. Cmos variable transconductance circuit with constant bandwidth. *Electron Letters*, 25:675–676, May 1989.

[54] E. Klumpernick, E. V. D. Zwan, and E. Seevinck. A cmos ota for hf filters with programmable transfer function. *IEEE Journal of Solid-State Circuits*, 26:1720–1724, November 1991.

[55] Y. T. Lai and P. T. Wang. Hierarchical interconnection structure for field programmable gate arrays. *IEEE Trans. VLSI*, 5:186–196, November 1997.

[56] F. Larsen and M. Ismail. A continuous-time offset compensated high speed bicmos comparator. *IEEE Symposium on Circuits and Systems*, 1:285–288, May 1996.

[57] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Trans.Compt.*, C–24:1145–1155, 1975.

[58] E. K. F. Lee. Low-voltage op amp design and differential difference amplifier design using linear transconductor with resistor input. *IEEE Trans. Circuits and Systems II*, 47:776–778, August 2000.

[59] E. K. F. Lee and P. G. Gulak. Error correction technique for multivalued mos memory. *Electronic Letters*, 27,(15):1321–1323, 1989.

[60] E. K. F. Lee and P. G. Gulak. A cmos field programmable analog array. In *IEEE Journal of Solid State Circuits*, volume 26, dec 1991.

[61] E. K. F. Lee and P. G. Gulak. Field programmable analogue array based on mosfet transconductors. In *Electronic Letters*, volume 28, January 1992.

[62] C. E. Leiserson. Fat trees:universal networks for hardware-effecient supercomputing. *IEEE Trans.Compt.*, C–34:892–901, 1985.

[63] S. Liu, J. S. Wang, H. W. Tsao, and J. Wu. Toca based electronically-tunable continuous-time filters. *Int. Journ. Electronics*, 71,(2):253–264, 1991.

[64] C. A. Looby and C. Lyden. A cmos continuous-time field programmable analogue array. In *FPGA 97*. Monterey California, February 1997.

[65] S. A. Mahmoud and A. M. Soliman. The current-feedback differential difference amplifier: new cmos realization with rail to rail class-ab output stage. *Proceedings of the 1999 IEEE International Symposium on on Circuits and Systems*, 2:120–123, May 1999.

[66] S. M. Mallya and J. H. Nevin. Design procedures for a fully differential folded-cascoded operational amplifier. *IEEE Journal of Solid-State Circuits*, 24:1737–1740, December 1989.

[67] K. Martin, L. Ozcolak, Y. S. Lee, and G. C. Temes. A differential switched-capacitor amplifier. *IEEE Journal of Solid-State Circuits*, 22:104–106, February 1987.

[68] R. Naiknaware and T. S. Fiez. Automated hierarchical cmos analog circuit stack generation with intramodule connectivity and matching considerations. *IEEE Journal of Solid-State Circuits*, 34:304–317, 1999.

[69] A. Nedungadi and T. R. Viswanathan. Design of linear cmos transconductance elements. *IEEE Trans. Circuits and Systems*, 31:891–894, October 1984.

[70] G. Nicollini, F. Moretti, and M. Conti. High-frequency fully differential filter using operational amplifiers without common-mode feedback. *IEEE Journal of Solid-State Circuits*, 24:803–813, June 1989.

[71] O. Oliaei and P. Loumeau. A low-input resistance class ab cmos current-conveyor. In *MidWest Symposium on Circuits and Systems,Ames,Iowa*, 1996.

[72] C. S. Park and R. Schaumann. A high-frequency cmos linear transconductance element. *IEEE Trans. Circuits and Systems*, CAS–33:1132–1138, November 1986.

[73] J. H. Patel. Processor-memory interconnections for multiprocessors. In *6th Annual Symp. Computer Architecture*, pages 168–177, 1979.

[74] E. Pierzchala and M.A. Perkowski. High speed field programmable analog array architecture design. In *Analogix Corporation*.

[75] X. Quan and S. H. K. Embabi. A current-mode based field programmable analog array architecture for signal processing applications. In *IEEE Custom Integrated Circuits Conference*, 1998.

[76] P. Ramanathan, A. J. Dupont, and K. G. Shin. Clock distribution in general vlsi circuits. *IEEE Trans. Circuits and Systems-I*, CAS–41:395–404, May 1994.

[77] J. Ramirez-Angulo and F. Ledesma. The multiple input linear weighted differential amplifier: a new versatile analog circuit building block. *IEEE Symposium on Circuits and Systems*, 3:747–750, May 2002.

[78] R. Rivoir and F. Maloberti. A 1 mv resolution 10 ms/s rail-to-rail comparator in 0.5 μm low-voltage cmos digital process. *IEEE International Symposium on Circuits and Systems*, 1:461–464, June 1997.

[79] G. W. Roberts, D. G. Nairn, and A. S. Sedra. On the implementation of fully differential switched-capacitor ladder filters. *IEEE Trans. Circuits and Systems*, 33:452–455, April 1986.

[80] E. Säkinger and W. Guggenbühl. A versatile building block:the cmos differential difference amplifier. *IEEE Trans. Solid-State Circuits*, SC–22:287–294, April 1987.

[81] M. C. Schneider, S. Noceti Filho, and R. N. G. Robert. A linear differential cmos vi converter. *IEEE International Symposium on Circuits and Systems*, 3:1717–1720, May 1990.

[82] A. S. Sedra, C. W. Roberts, and F.Gohh. The current-conveyors:history,progress and new results. In *IEE Proceedings 137G(2)*, April 1990.

[83] A. S. Sedra and K. C. Smith. A second-generation current conveyor and its applications. *IEEE Trans. Circuit Theory*, CT–17:132–134, February 1970.

[84] E. Seevinck and R. F. Wassenaar. A versatile cmos linear transconductor/square-law function cicuit. *IEEE Journal of Solid-State Circuits*, 22:366–377, June 1987.

[85] C. Shi, Y. Wu, H. O. Elwan, and M. Ismail. A class of fully-differential basic building blocks based on unity-gain difference feedback. *IEEE International Symposium on Circuits and Systems*, 2:152–155, May 2000.

[86] J. Silva-Martinez, M. S. J. Steyaert, and W. M. C. Sansen. A large signal very low-distortion transconductor for high-frequency continuous-time filters. *IEEE Journal of Solid-State Circuits*, 26:946–955, July 1991.

[87] A. Stoica, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, G. Klimeck, Y. Jin, R. Tawel, and V. Duong. Evolution of analog circuits on field programmable transistor arrays. In *The Second NASA/DoD Workshop on Evolvable Hardware (EH'00)*. Palo Alto, California, July 2000.

[88] D. Stroobandt, P.Verplaetse, and J.V. Campenhout. Towards synthetic benchmark circuits for evaluating timing-driven cad tools. In *ISPD 99 Monterey CA USA*, pages 60–66, 1999.

[89] S. Szezepanski, R. Schaumann, and P. Wu. Linear transconductor based on cross-coupled cmos pairs. *Electron Letters*, 27:783–785, April 1991.

[90] D. E. Tiliute. Second-order active filter using a single current conveyor. Technical report, Electrical Engineering Dept.,University of Suceava,Romania.

[91] A. H. Titus and T. J. Drabik. An improved silicon retina chip with optical input and optical output. *IEEE International ASIC Conference and Exhibit*, 1:88–91, September 1997.

[92] A. H. Titus and A. Gopalan. A differential summing amplifier for analog vlsi systems. *IEEE Symposium on Circuits and Systems*, 4:747–750, May 2002.

[93] R. Torrance, T. Viswanathan, and J. Hanson. Cmos voltage to current transducers. *IEEE Trans. Circuits and Systems*, 32:1097–1104, November 1985.

[94] C. Toumazou, F. J. Lidjey, and D. G. Haigh. *Analog IC design: the current-mode approach*, chapter 3,Current Conveyor Theory and Practise. London Peregrinus, 1990.

[95] Y. Tsividis, Z. Czarnul, and S. C. Fang. Mos transconductors and integrators with high linearity. *Electron Letters*, 22:245–246, May 1986.

[96] P. M. VanPetegham and J. F Duque-Carrillio. A general description of common-mode feedback in fully-differential amplifiers. *IEEE International Symposium on Circuits and Systems*, 3:3209–3212, 1990.

[97] P. D. Walker and M. G. Green. An approach to fully differential circuit design without common-mode feedback. *IEEE Trans. Circuits and SystemsII Analog and Digital Signal Processing*, 43:752–762, November 1996.

[98] P. D. Walker and M. M. Green. Cmos half-wave and full-wave precision voltage rectification circuits. *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, 2:901–904, August 1995.

[99] P. T. Wang, K. N. Chen, and Y. T. Lai. A high performance fpga with hierarchical interconnection structure. In *Proceedings of ISCAS*, pages 239–242, 1994.

[100] P. T. Wang and J. J. Tang. Graph-based detailed router for hierarchical field-programmable gate arrays. *IEE Proceedings Computers and Digital Techniques*, 146:57–67, January 1999.

[101] Z. Wang. Novel linearisation technique for implementing large-signal mos tunable transconductor. *Electron Letters*, 26:138–139, January 1990.

[102] S. Wenxiao, H. Qingquan, and C. Wang. Fully differential current-mode filter based on mddcc. In *IEEE Asia-Pacific Conference on Circuits and Systems*, pages 674–677, December 2000.

[103] B. Wilson. Recent developments in current-conveyors and current-mode circuits. In *IEE Proceedings 137G(2)*, April 1990.

[104] F. You, S. H. K. Embabi, and E. Sanchez-Sinencio. Multistage amplifier topologies with nested gm-c compensation. *IEEE Journal of Solid-State Circuits*, 32:2000–2011, December 1997.

[105] S. R. Zarabadi, F. Larsen, and M. Ismail. A reconfigurable op-amp/dda cmos amplifier architecture. *IEEE Trans. Circuits and SystemsI*, 39:484–487, June 1992.

[106] X. Zhang, B. J. Maundy, E. I. El-Masry, and I. G. Finvers. A novel low-voltage operational transconductance amplifier and its applications. In *IEEE International Symposium on Circuits and Systems-II, ISCAS '2000*, volume 3, pages 661–664, May 2000.

Appendix A

Examples of DDA Based Simulated Applications in PSpice

This Appendix contains the following:

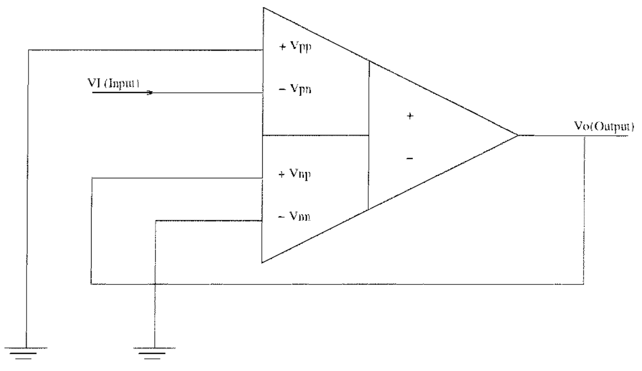


FIGURE A.1: DDA Inverter

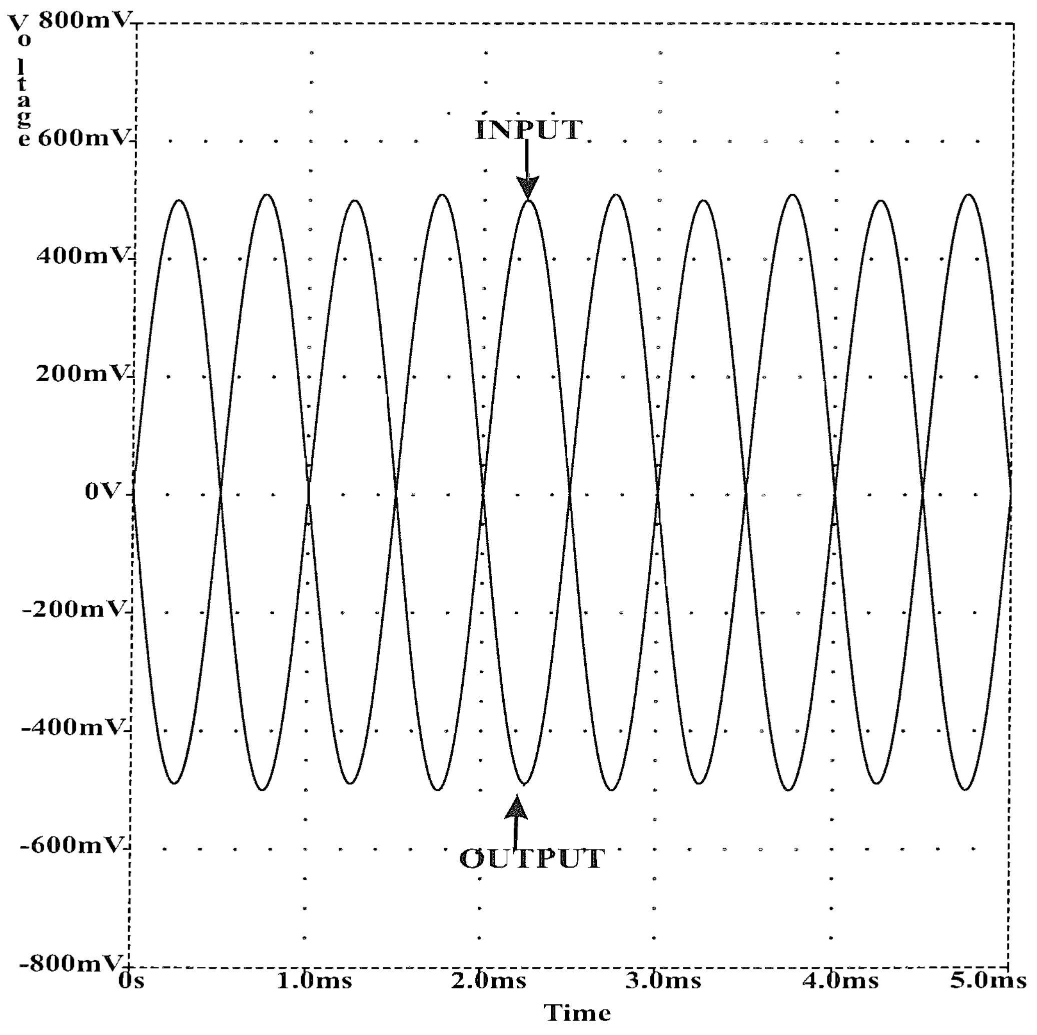


FIGURE A.2: Simulation of the DDA Inverter

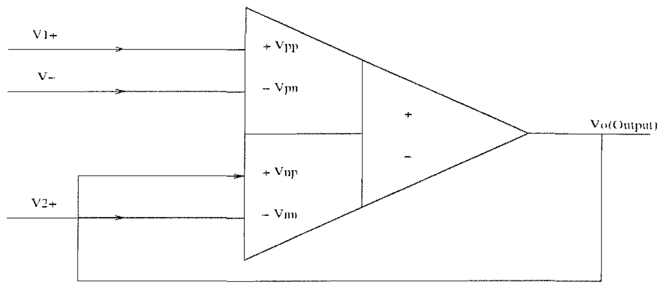


FIGURE A.3: DDA Adder/Subtractor

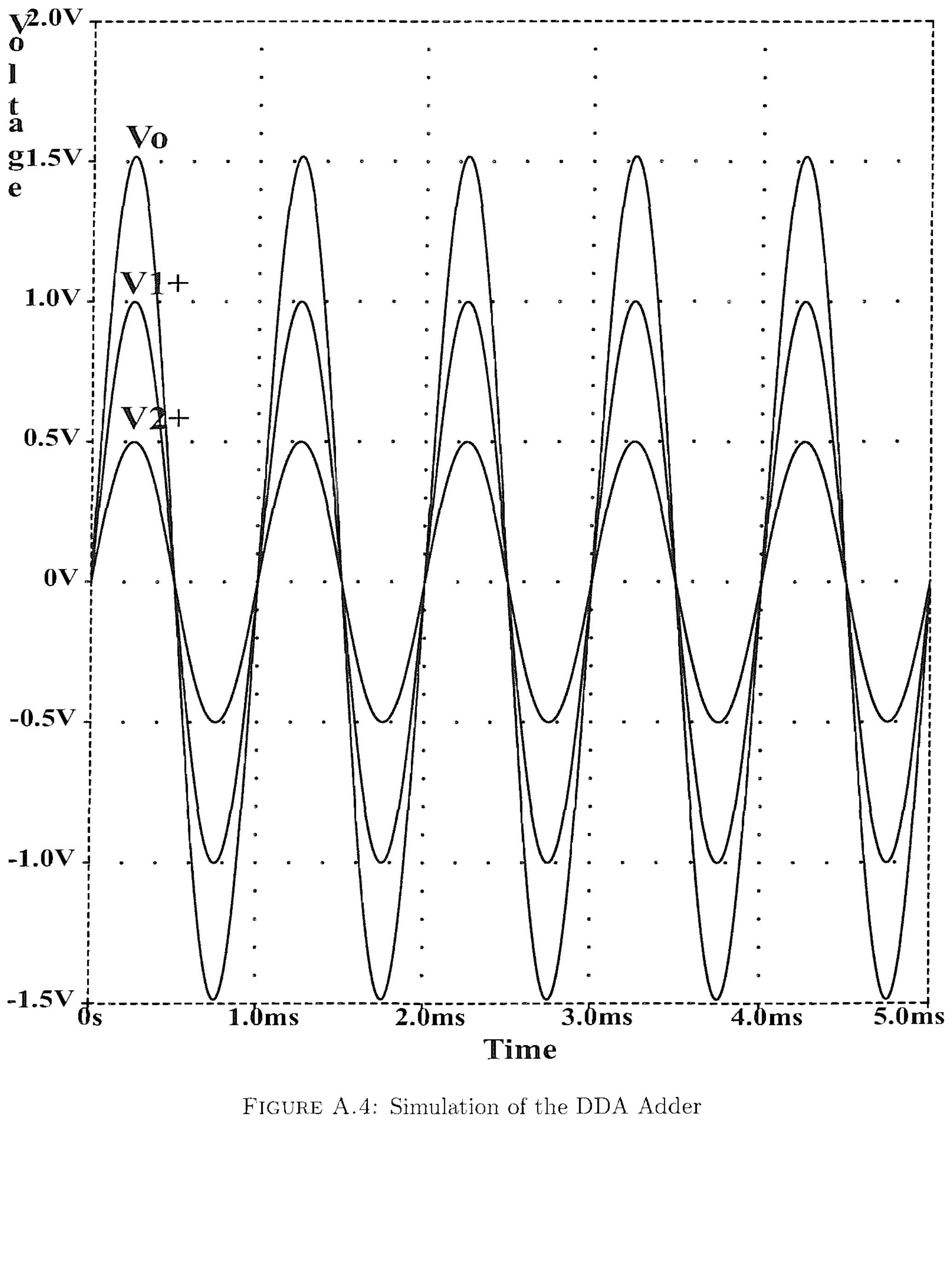


FIGURE A.4: Simulation of the DDA Adder

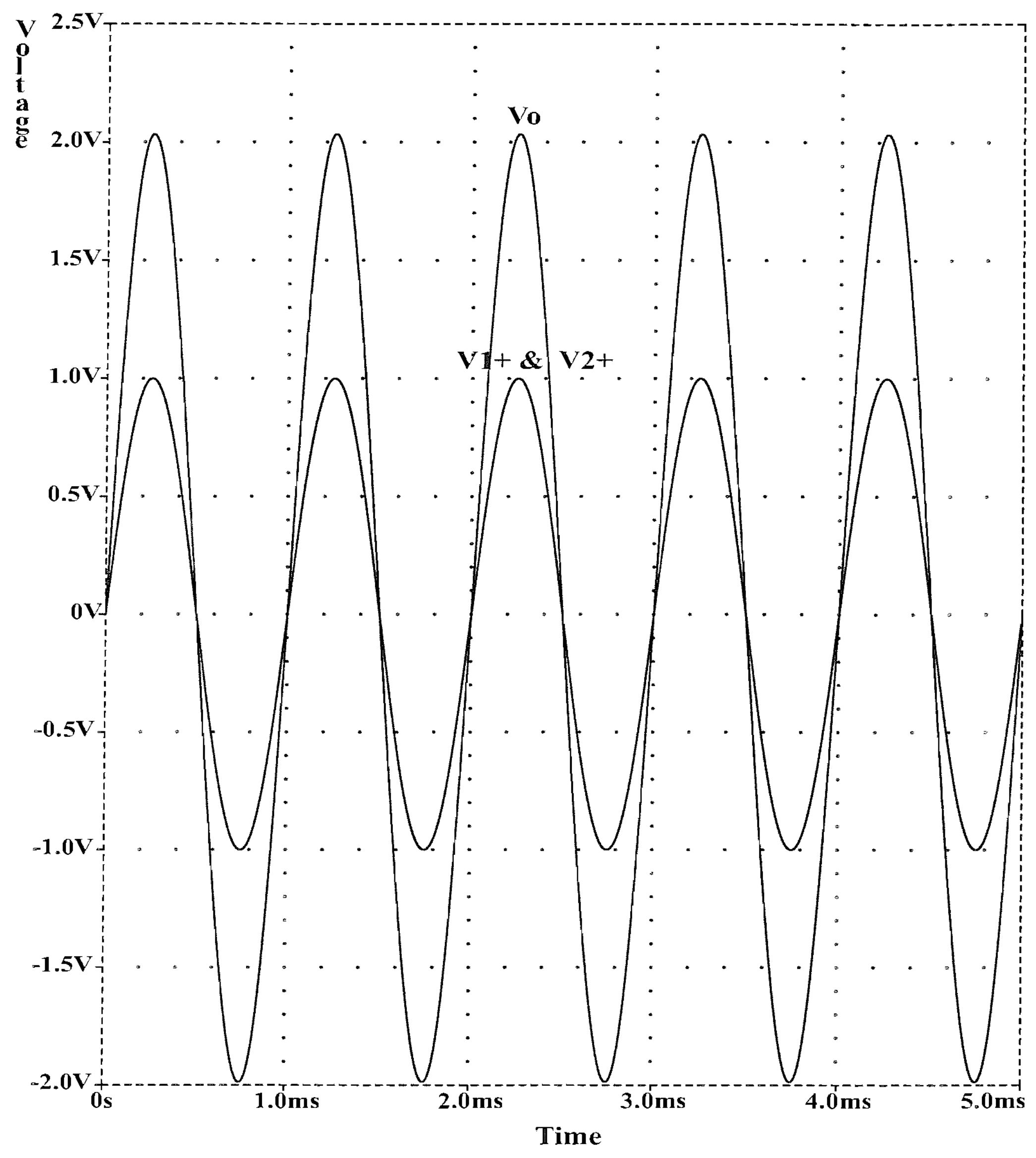


FIGURE A.5: Simulation of the DDA Voltage Doubler

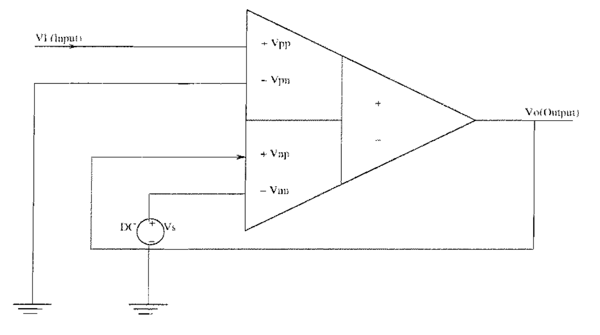


FIGURE A.6: DDA Level Shifter

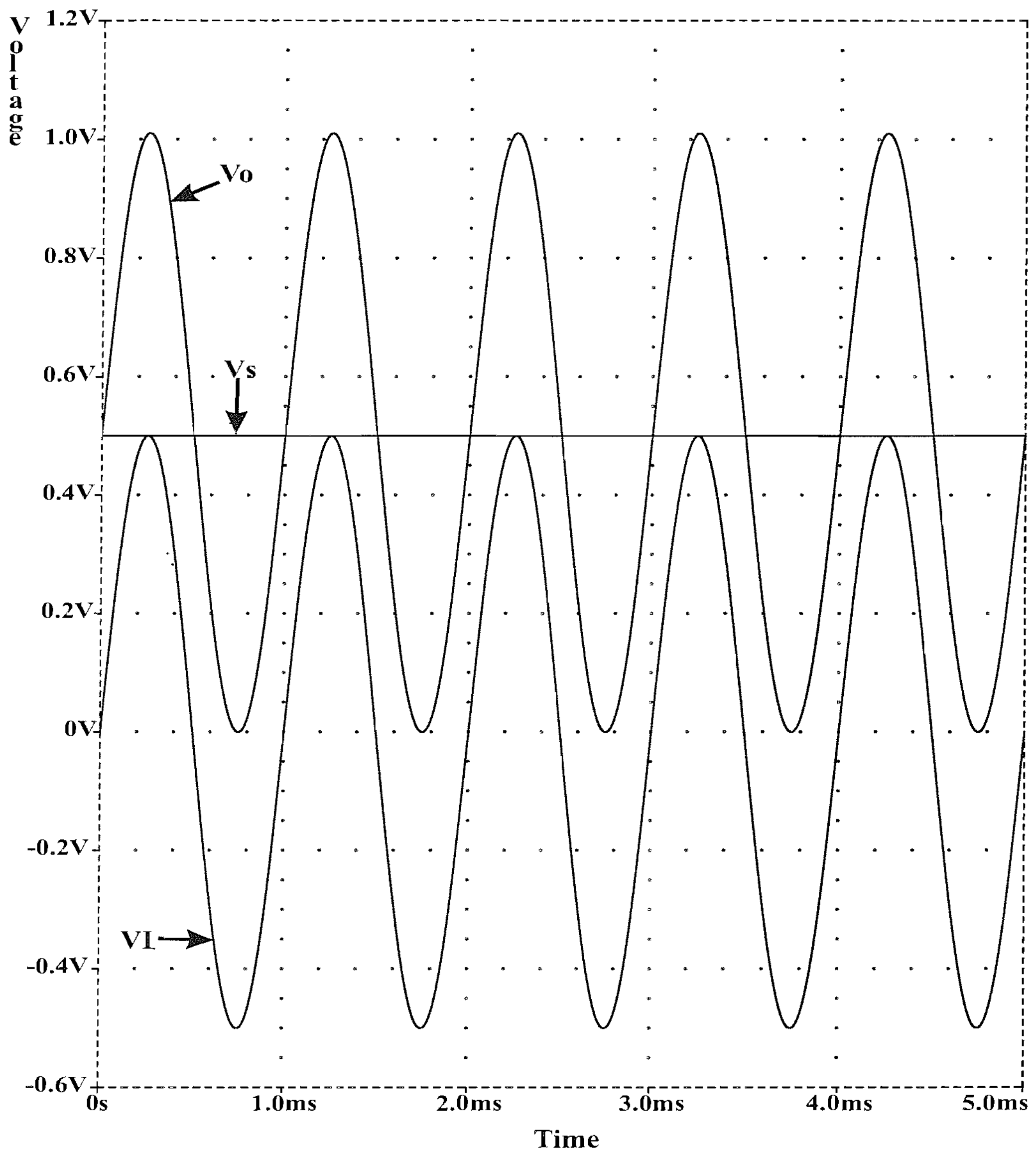


FIGURE A.7: Simulation of the DDA Level Shifter

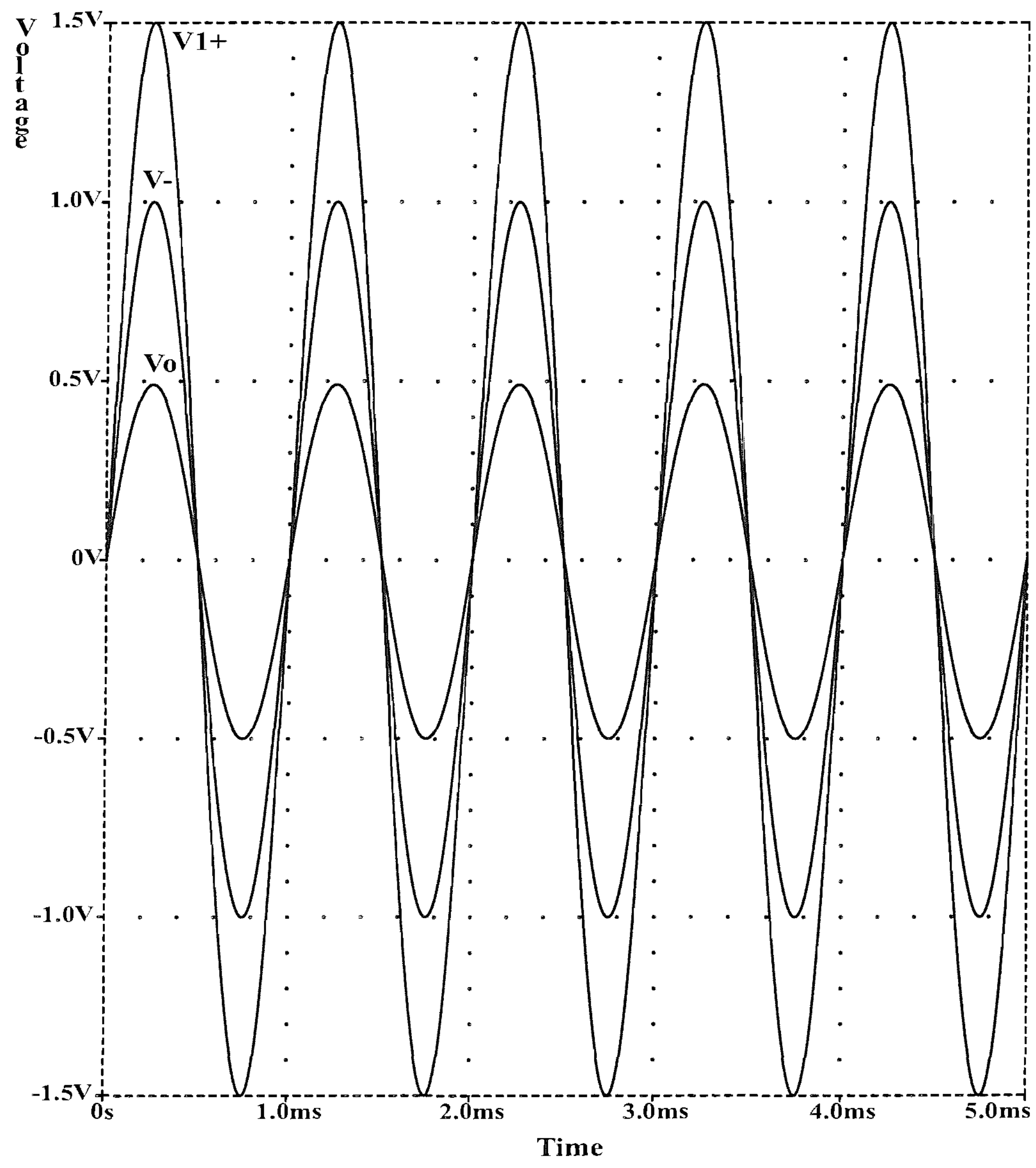


FIGURE A.8: Simulation of the DDA Subtractor

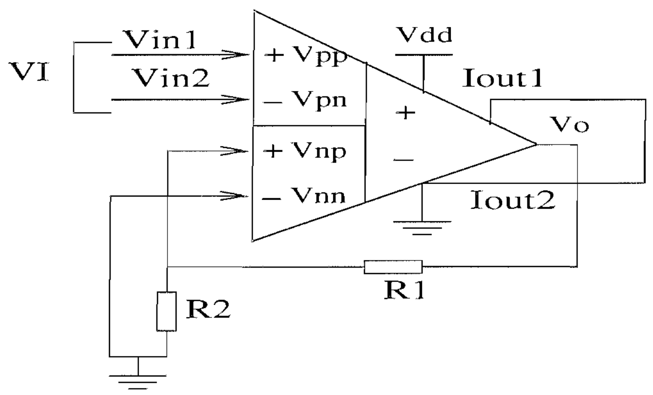


FIGURE A.9: DDA based Instrumentation Amplifier

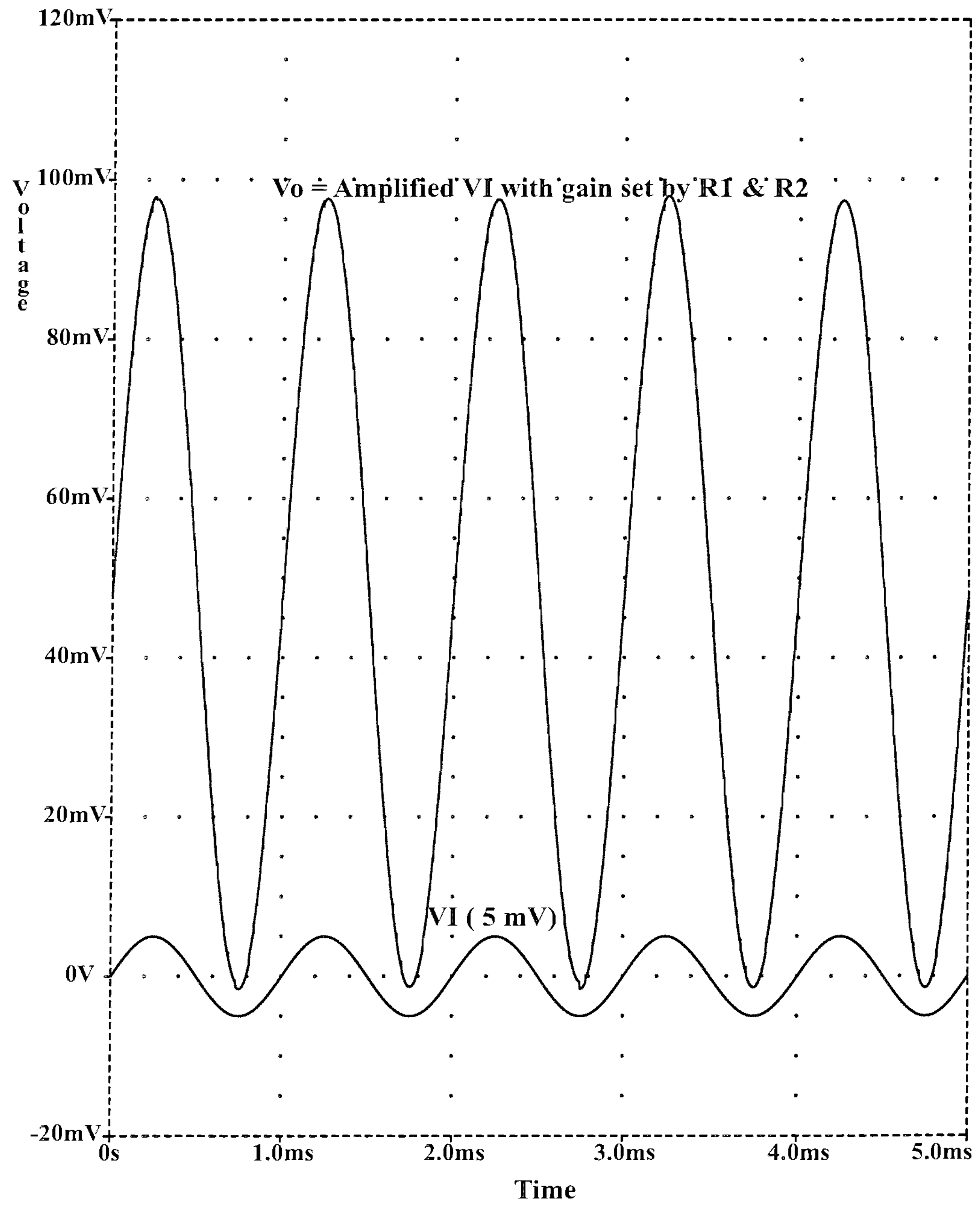


FIGURE A.10: Simulation of the DDA Instrumentation Amplifier

Appendix B

Schematics and Graphs

This Appendix contains the following:

- Class AB Amplifier Analysis Figure B.1
- DDA Design with the different class AB output stage Figure B.2
- DC Response of the DDA Figure B.3
- Modified DDA circuit Figure B.4
- Alternative Class AB stage Analysis Figure B.5
- DC Response of the modified DDA Figure B.6
- Compensated DDA Figure B.7
- Transient Response of the DDA without external load resistance Figure B.8
- Transient Response of the DDA with 10pF load capacitance Figure B.9
- Open-loop compensated frequency response of the DDA Figure B.10
- Open-loop gain and phase margin of the DDA driving a 10pF capacitive load Figure B.11
- CMRR of the DDA Figure B.12
- Modified DDA Figure B.13
- Test circuit for simulating DDVR of the DDA Figure B.14
- DDVR of the DDA Figure B.15
- DDA with offset compensation Figure B.16
- DDA slew Response and Settling time Response for 10pF load Figure B.17
- DDA slew Response and Settling time Response for 1pF load Figure B.18
- DDA/Current Conveyor Figure B.19
- Simulation setup for transient analysis Figure B.20
- Transient Response of the DDACCII with varying external resistance Figure B.21
- Transient Response of the DDACCII with Idc=30μA and varying external resistance Figure B.22

- Noise Response of DDACCII Figure B.23
- Test circuit for harmonic distortion and its obtained response Figure B.24
- Symbol of the final design of DDACCII CAB Figure B.25
- Schematic of the final design of DDACCII CAB Figure B.26
- Cluster Design Figure B.27
- Block level Schematic of the target FPAA chip Figure B.28
- Simulation setup for testing the FPAA chip Figure B.29
- Transient responses of chip simulation Figure B.30
- Simulated transient responses of the Schmitt Trigger Figure B.31 Simulated transient responses of the VCO Figure B.32

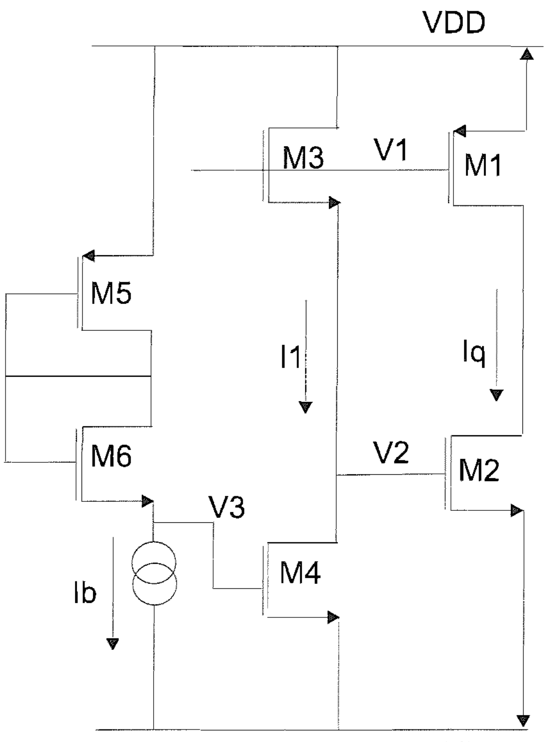


FIGURE B.1: Class AB amplifier Analysis

B.1 Class AB amplifier analysis

From the Figure B.1, the following can be derived. For the transistor M2, the quiescent current is:

$$I_q = K_n \left(\frac{W_2}{2L_2}\right)(V_2 - V_{Tn})^2 \tag{B.1}$$

Similarly, the quiescent current for the transistor M1 is:

$$I_q = K_p \left(\frac{W_1}{2L_1}\right)(V_1 - VDD - V_{Tp})^2 \tag{B.2}$$

Thus the voltages V_1 and V_2 can be written as:

$$V_1 = VDD + V_{Tp} - \sqrt{\frac{I_q 2L_1}{K_p W_1}} \tag{B.3}$$

$$V_2 = V_{Tn} + \sqrt{\frac{I_q 2L_2}{K_n W_2}} \tag{B.4}$$

The voltage (V_1 - V_2) may be equated to the gate-to-source voltage for the transistor M3 (neglecting the effect of substrate potential) as:

$$V_1 - V_2 = VDD - V_{Tn} + V_{Tp} - \sqrt{I_q}\left(\sqrt{\frac{2L_1}{K_p W_1}} + \sqrt{\frac{2L_2}{K_n W_2}}\right) = V_{Tn} + \sqrt{I_1}\sqrt{\frac{2L_3}{K_n W_3}} \tag{B.5}$$

Hence, I$_q$ may be expressed in terms of I$_1$ as:

$$\sqrt{I_q}\left(\sqrt{\frac{2L_1}{K_pW_1}}+\sqrt{\frac{2L_2}{K_nW_2}}\right) = VDD - 2V_{Tn} + V_{Tp} - \sqrt{I_1}\sqrt{\frac{2L_3}{K_nW_3}} \tag{B.6}$$

In the above equation the current I$_1$ is given by:

$$I_1 = \frac{K_nW_4}{2L_4}(V_3 - V_{Tn})^2 \tag{B.7}$$

Hence, the equation B.6 becomes:

$$\sqrt{I_q}\left(\sqrt{\frac{2L_1}{K_pW_1}}+\sqrt{\frac{2L_2}{K_nW_2}}\right) = VDD - 2V_{Tn} + V_{Tp} - \sqrt{\frac{2L_3}{K_nW_3}}\sqrt{\frac{K_nW_4}{2L_4}}(V_3 - V_{Tn}) \tag{B.8}$$

Now, considering the transistors M5 and M6, the voltage V$_3$ is given by:

$$V_3 = VDD - V_{Tn} + V_{Tp} - \sqrt{I_b}\left(\sqrt{\frac{2L_5}{K_pW_5}}+\sqrt{\frac{2L_6}{K_nW_6}}\right) \tag{B.9}$$

Transistor	Type	W	L
M1 M2 M3 M4	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M10	N	28um	2um
M15	P	88um	2um
M16	N	28um	2um
M17	P	4um	2um
M18	N	2um	2um

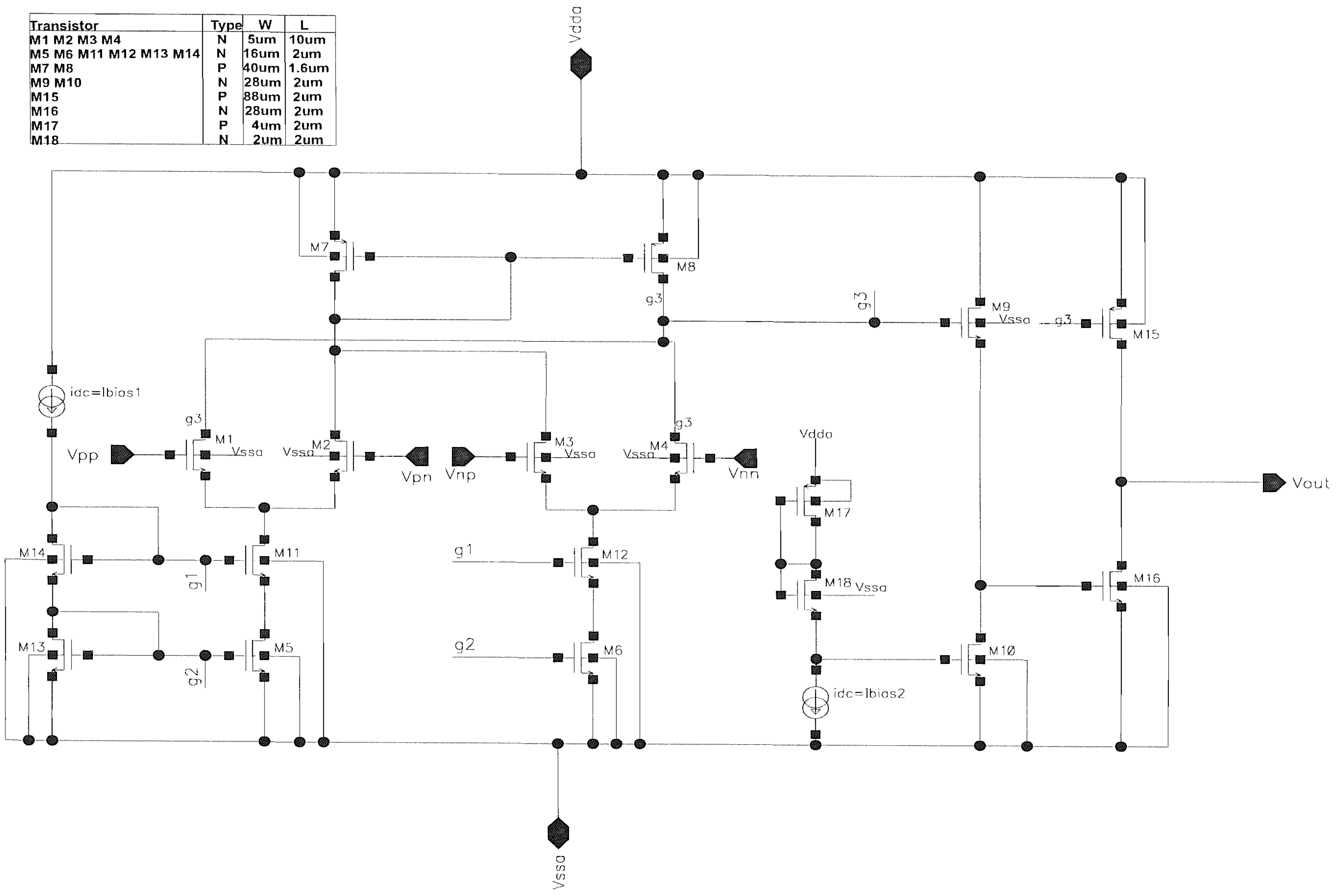


FIGURE B.2: DDA Design with the different class AB output stage

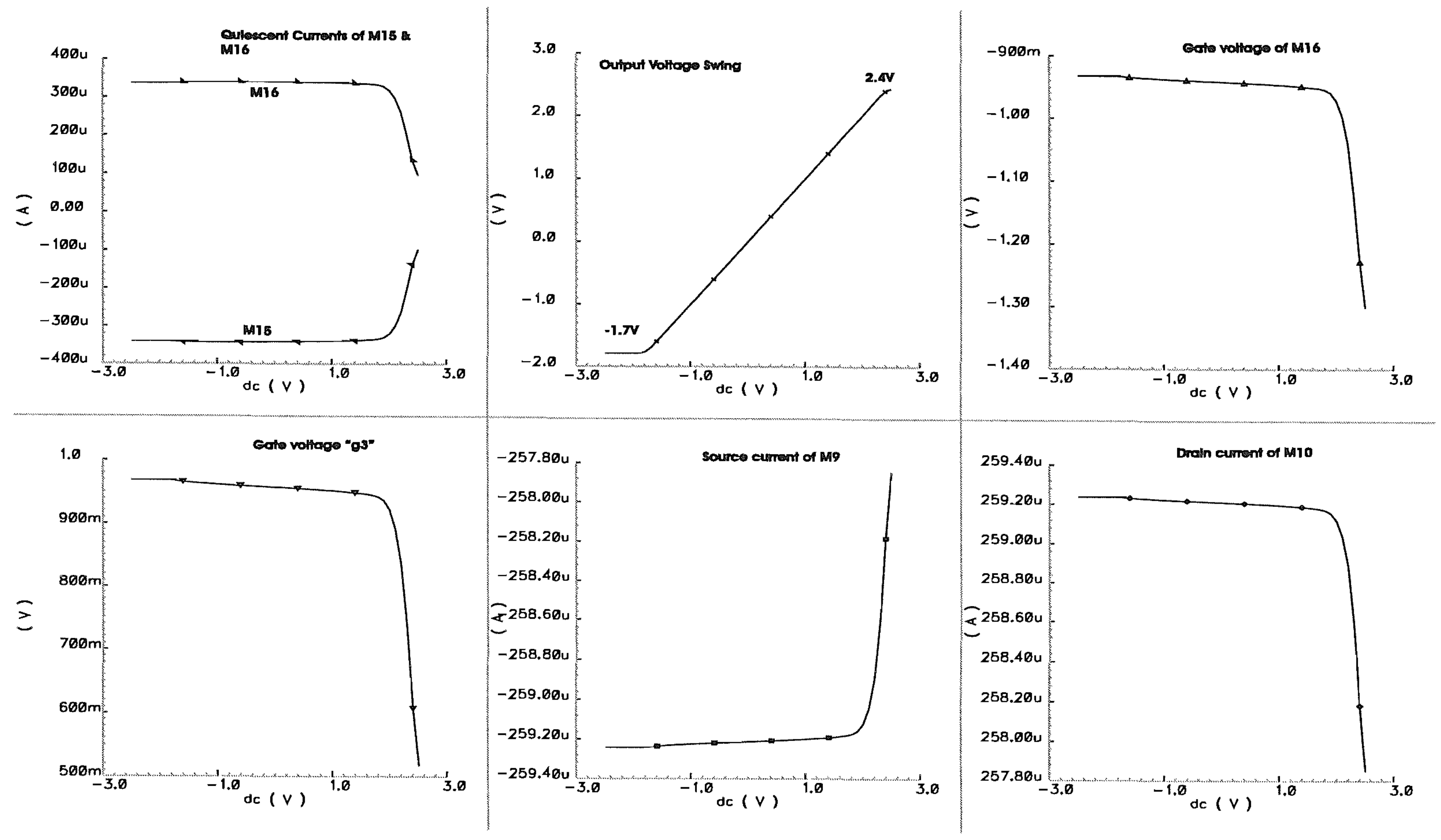


FIGURE B.3: DC Response of the DDA

Transistor	Type	W	L
M1 M2 M3 M4	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M19	N	28um	12um
M10	N	28um	20um
M15	P	88um	2um
M16	N	28um	2um
M17	P	88um	2um
M18 M20	N	28um	2um

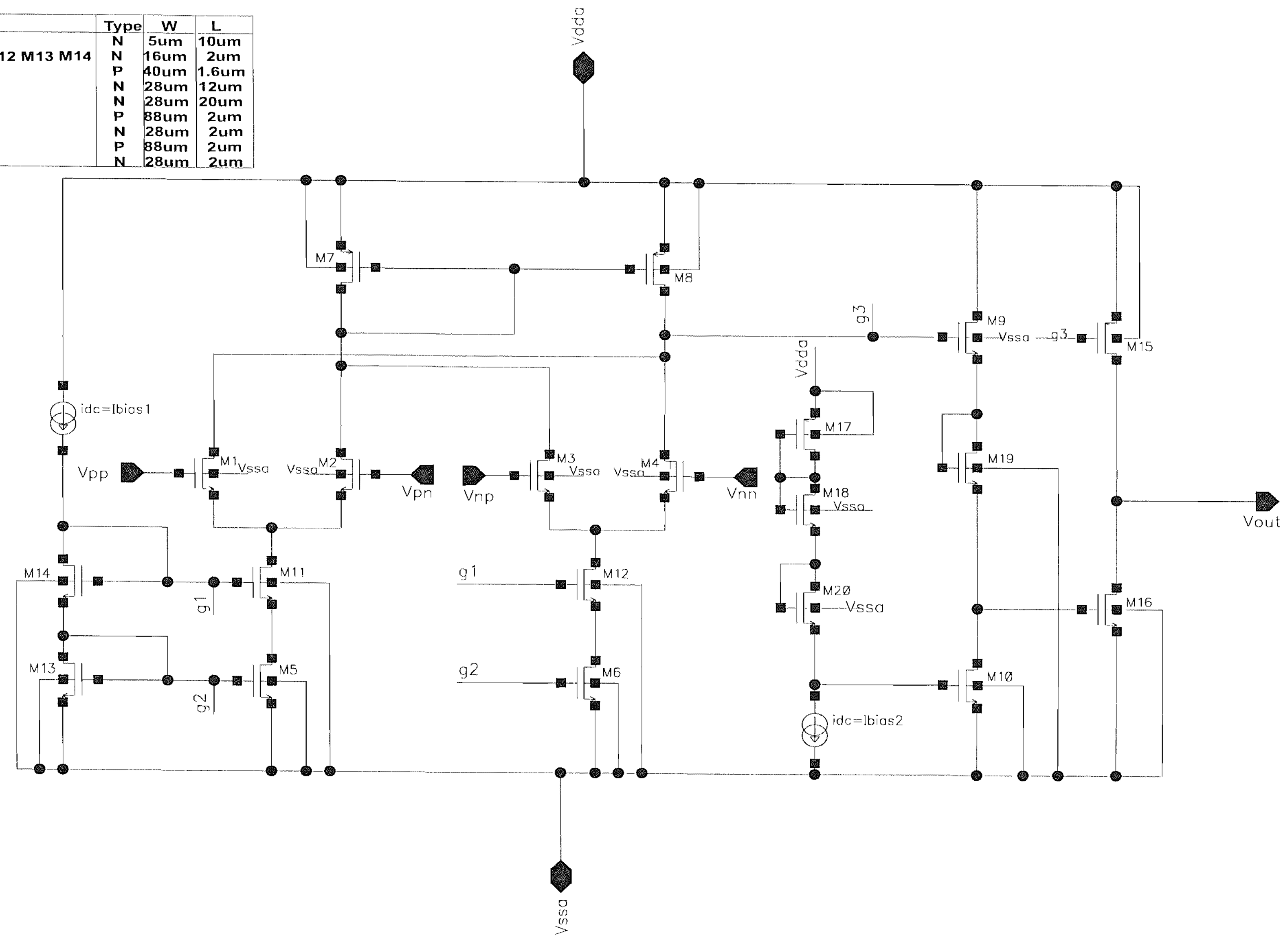


FIGURE B.4: Modified DDA circuit

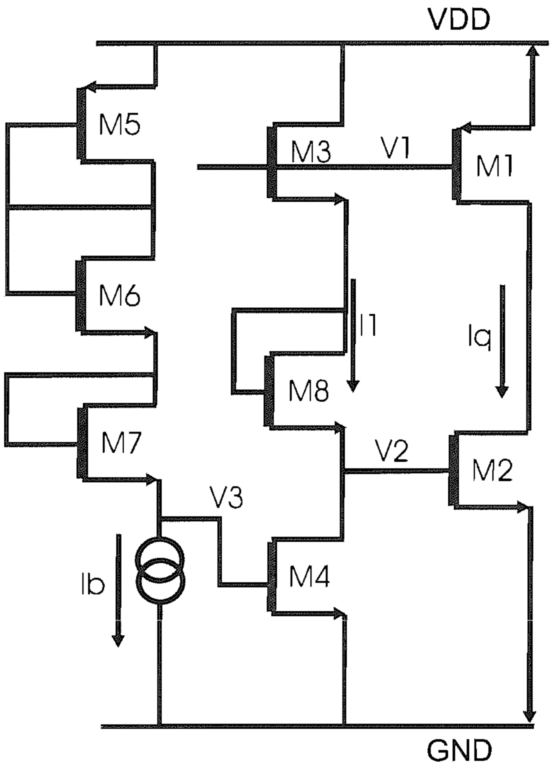


FIGURE B.5: Alternative Class AB stage Analysis

B.2 Alternate Class AB stage analysis

We know that for a transistor in saturation, the drain current can be written as:

$$I = K\left(\frac{W}{2L}\right)(Vgs - V_T)^2 = \frac{\beta}{2}(Vgs - V_T)^2 \tag{B.10}$$

The threshold voltage V_T in the above equation is a function of the source to substrate voltage, equal to V_{Tn0} or V_{Tp0} at V_{sb}=0. Considering the quiescent currents in the output transistors M1 and M2, the current I_q can be written as:

$$I_q = \frac{\beta_1}{2}(V_1 - V_{Tp0})^2 and I_q = \frac{\beta_2}{2}(V_2 - V_{Tn0})^2 \tag{B.11}$$

Rearranging the terms in the above equation, V_1 and V_2 can be written as:

$$V_1 = -\sqrt{\frac{2I_q}{\beta_1}} + VDD + V_{Tp0} \tag{B.12}$$

$$V_2 = \sqrt{\frac{2I_q}{\beta_2}} - V_{Tn0} \tag{B.13}$$

Thus the difference in voltages between the gates of M1 and M2 is given by:

$$V_1 - V_2 = VDD - V_{Tn0} + V_{Tp0} - \sqrt{2I_q}\left(\frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}\right) \tag{B.14}$$

The difference between V_1 and V_2 is also determined by the gate-to-source voltages for M3 and M8. Hence the difference between V_1 and V_2 can also be written as:

$$V_1 - V_2 = V_{T3} + \sqrt{2I_1}\frac{1}{\sqrt{\beta_3}} + V_{T8} + \sqrt{2I_1}\frac{1}{\sqrt{\beta_8}} \tag{B.15}$$

Combining equations B.14 and B.15, we get:

$$VDD - V_{Tn0} + V_{Tp0} - \sqrt{2I_q}\left(\frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}\right) = V_{T3} + V_{T8} + \sqrt{2I_1}\left(\frac{1}{\sqrt{\beta_3}} + \frac{1}{\sqrt{\beta_8}}\right) \tag{B.16}$$

The above equation can be rearranged as:

$$\sqrt{2I_q}\left(\frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}\right) = VDD - V_{Tn0} - V_{T3} - V_{T8} + V_{Tp0} - \sqrt{2I_1}\left(\frac{1}{\sqrt{\beta_3}} + \frac{1}{\sqrt{\beta_8}}\right) \tag{B.17}$$

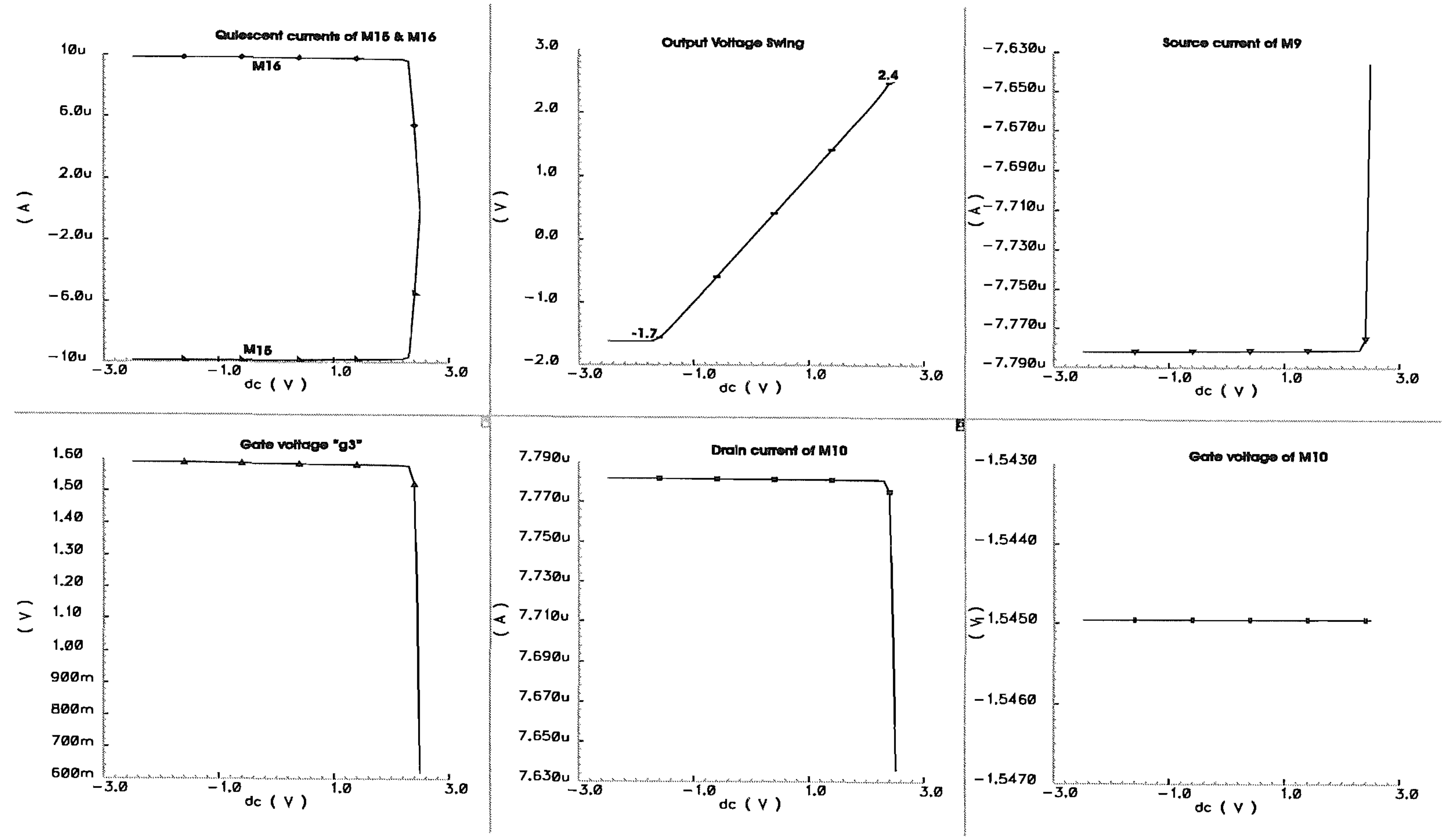


FIGURE B.6: DC Response of the modified DDA

Transistor	Type	W	L
M1 M2 M3 M4	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M19	N	28um	12um
M10	N	28um	20um
M15	P	88um	2um
M16	N	28um	2um
M17	P	88um	2um
M18 M20	N	28um	2um
M21	P	10um	0.6um

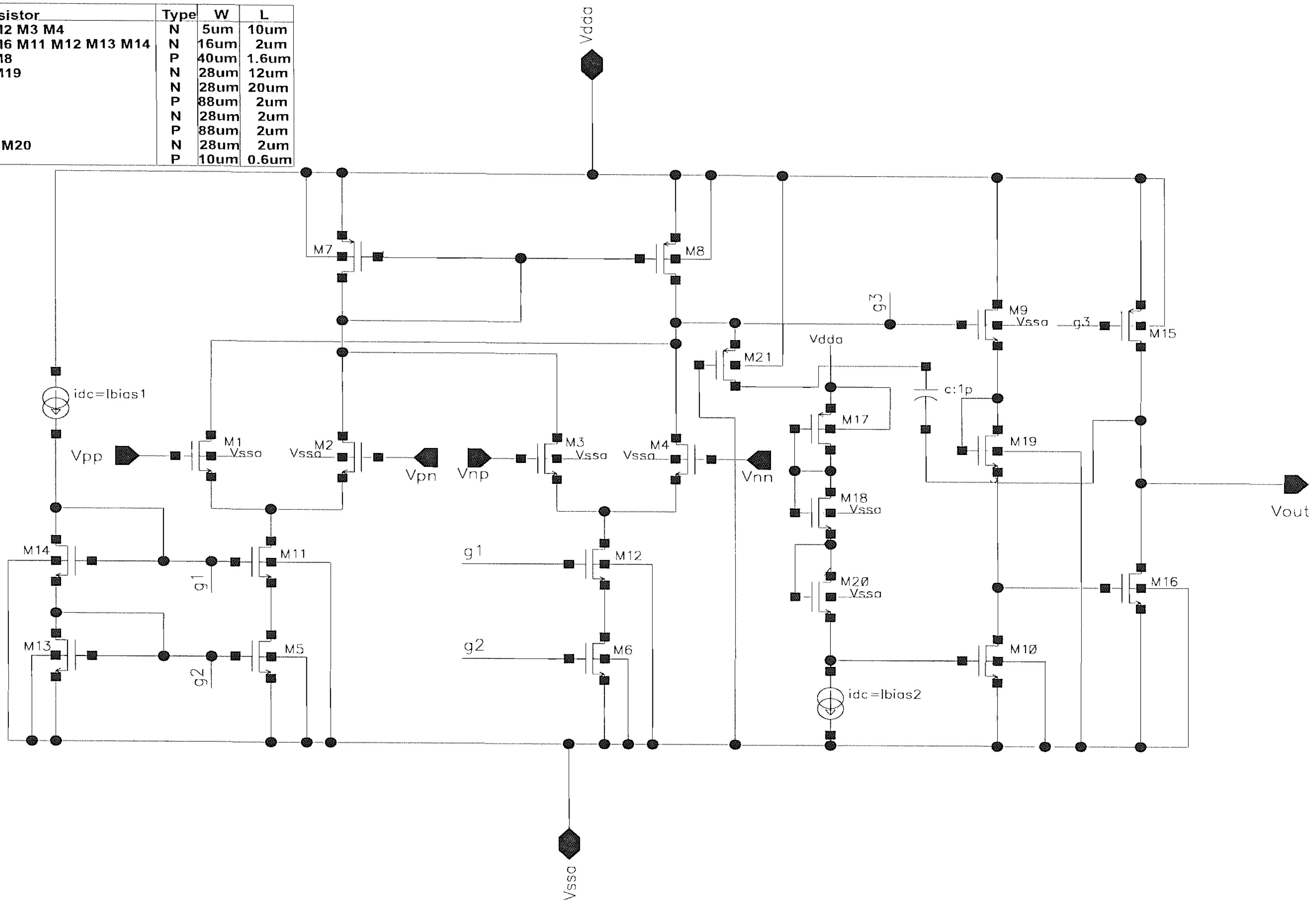


FIGURE B.7: Compensated DDA

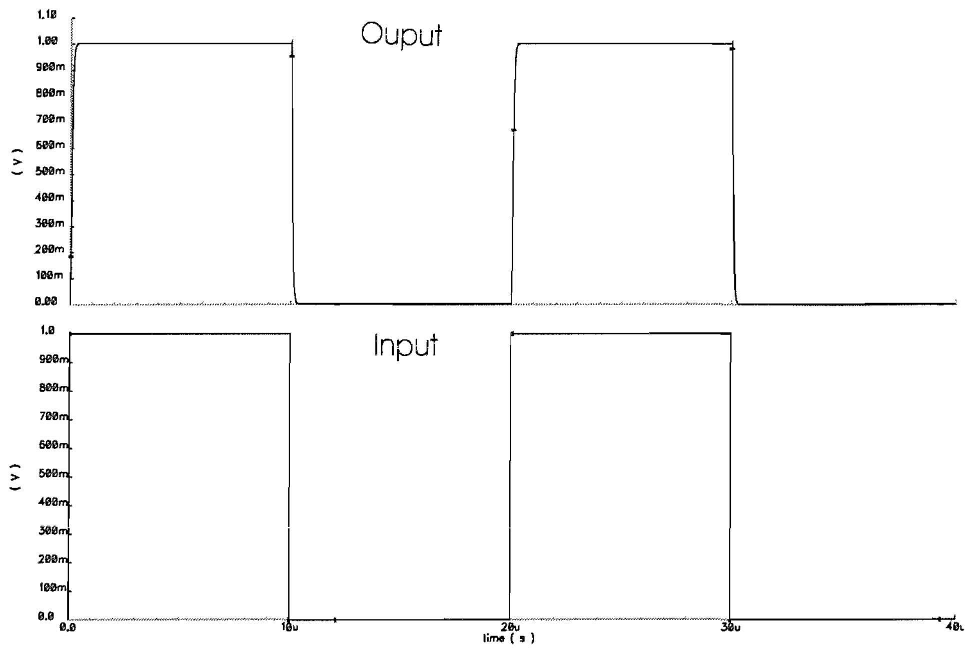


FIGURE B.8: Transient Response of the DDA without external load resistance

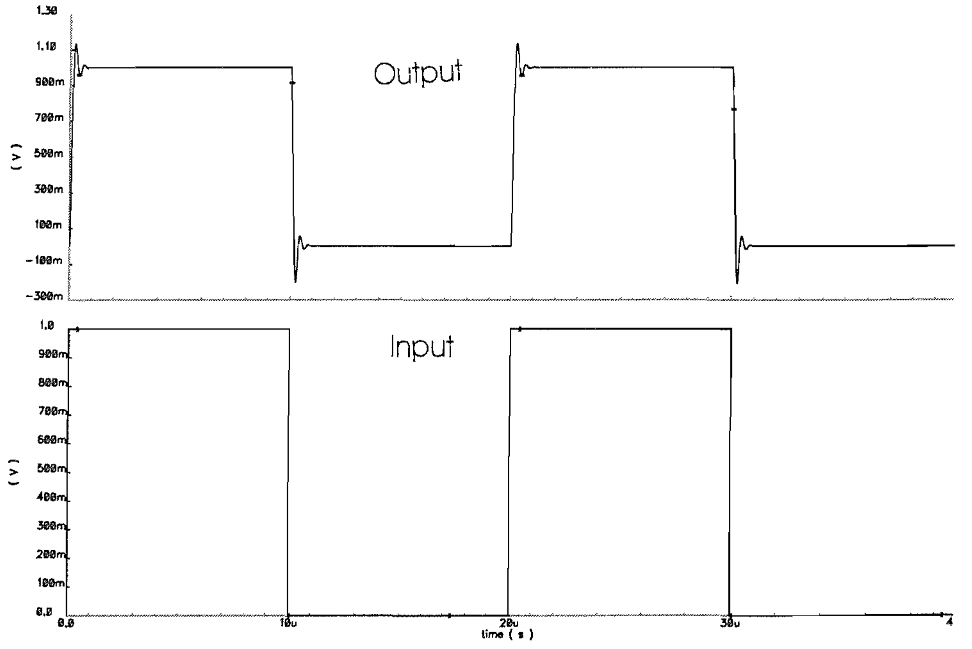


FIGURE B.9: Transient Response of the DDA with 10pF load capacitance

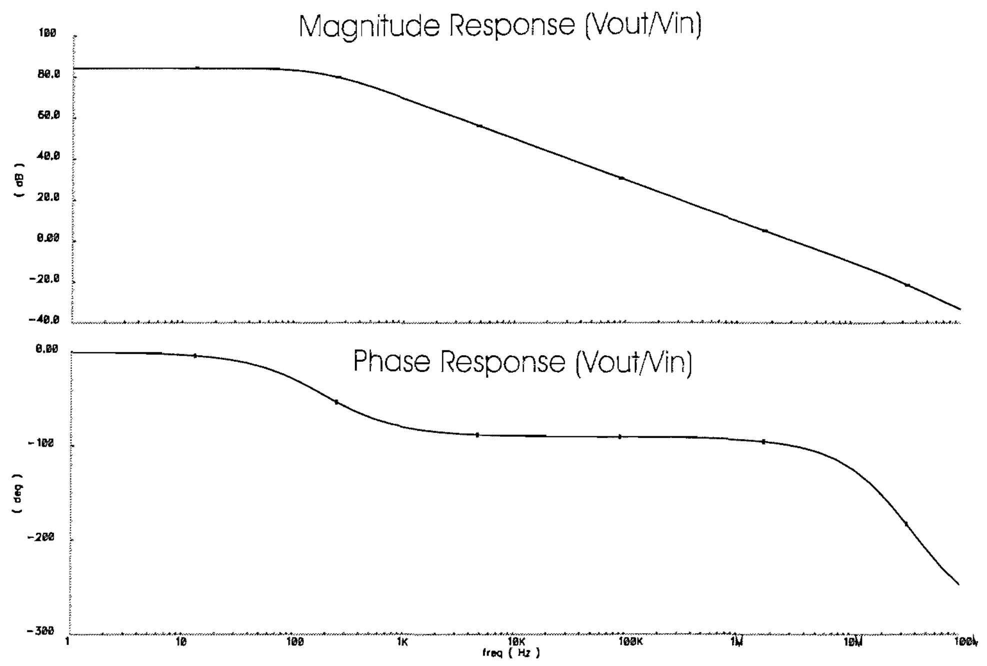


FIGURE B.10: Open-loop compensated frequency response of the DDA

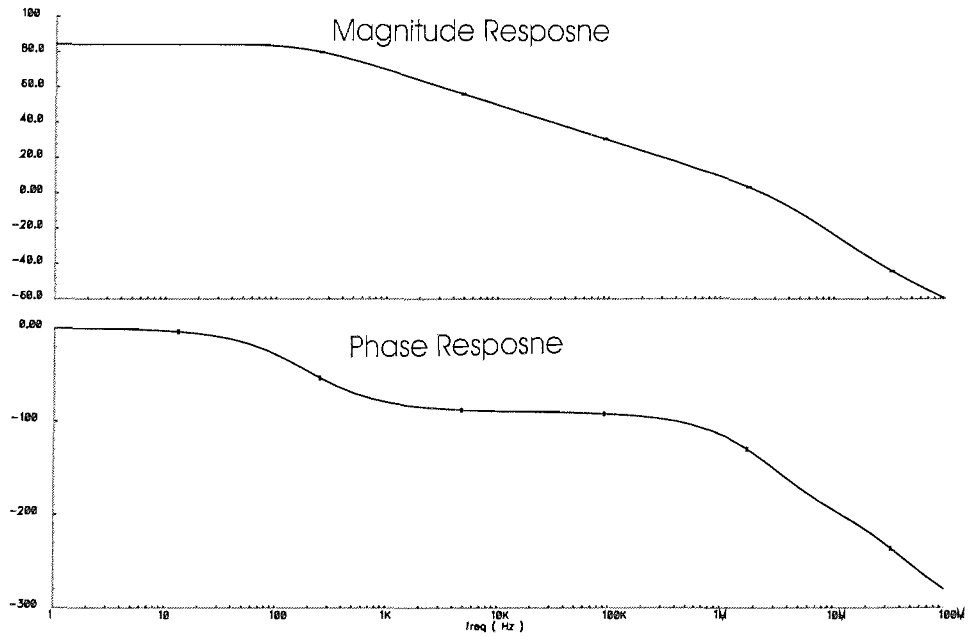


FIGURE B.11: Open-loop gain and phase margin of the DDA driving a 10pF capacitive load

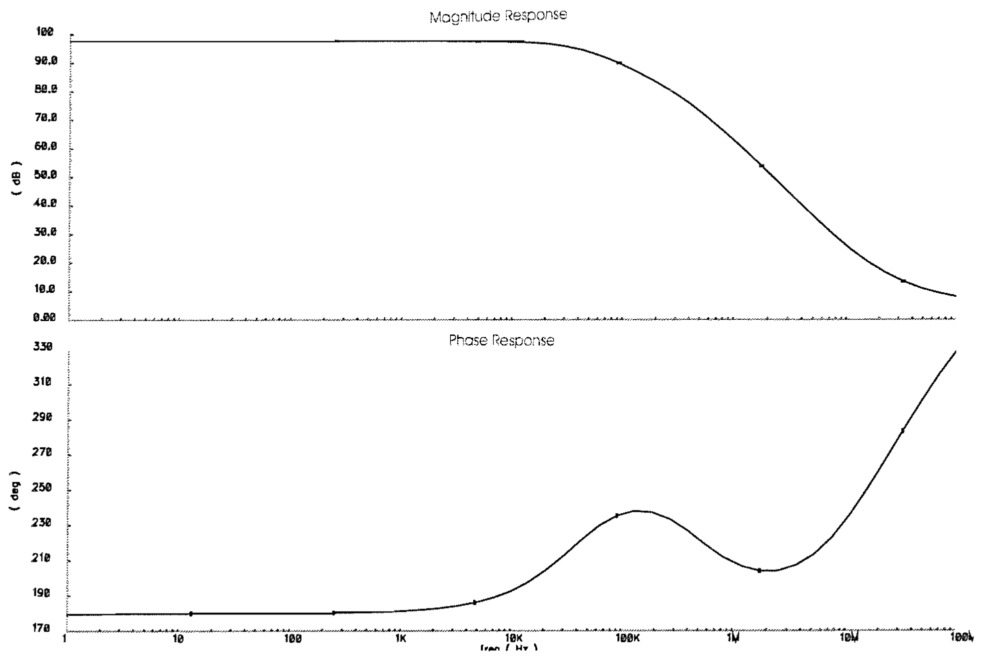


FIGURE B.12: CMRR of the DDA

Transistor	Type	W	L
M1 M2 M3 M4	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M19	N	28um	12um
M10	N	28um	20um
M15	P	88um	2um
M16	N	28um	2um
M17	P	88um	2um
M18 M20	N	28um	2um
M21	P	10um	0.6um
M22	N	4um	2um

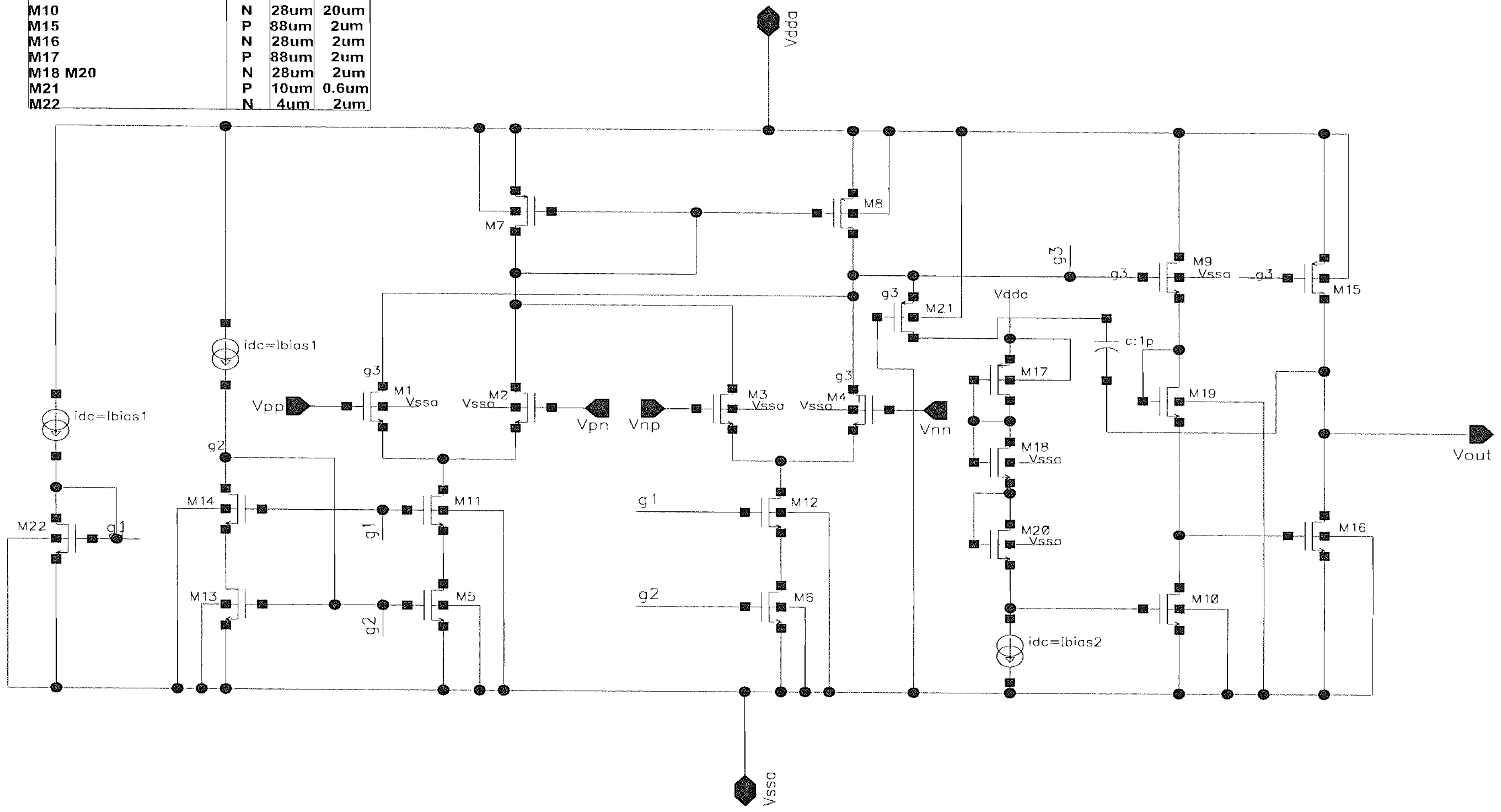

Vdda
Vssa
idc=Ibias1
idc=Ibias1
idc=Ibias2
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15 M16 M17 M18 M19 M20 M21 M22
Vpp
Vpn
Vnp
Vnn
Vout
g1
g2
g3
c:1p

FIGURE B.13: Modified DDA

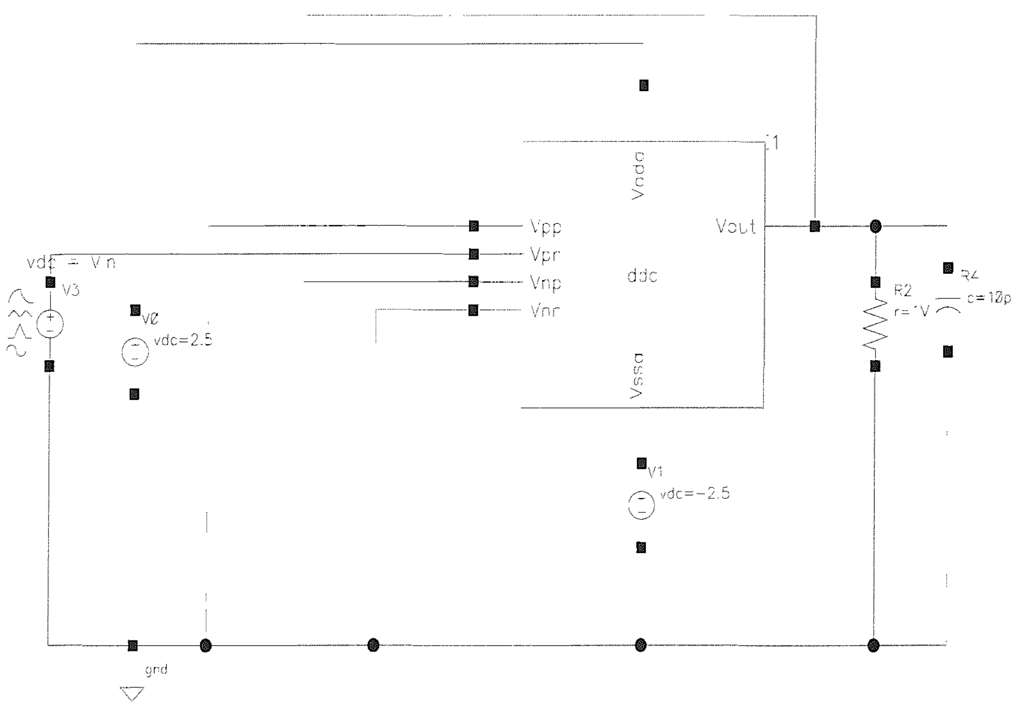


FIGURE B.14: Test circuit for simulating DDVR of the DDA

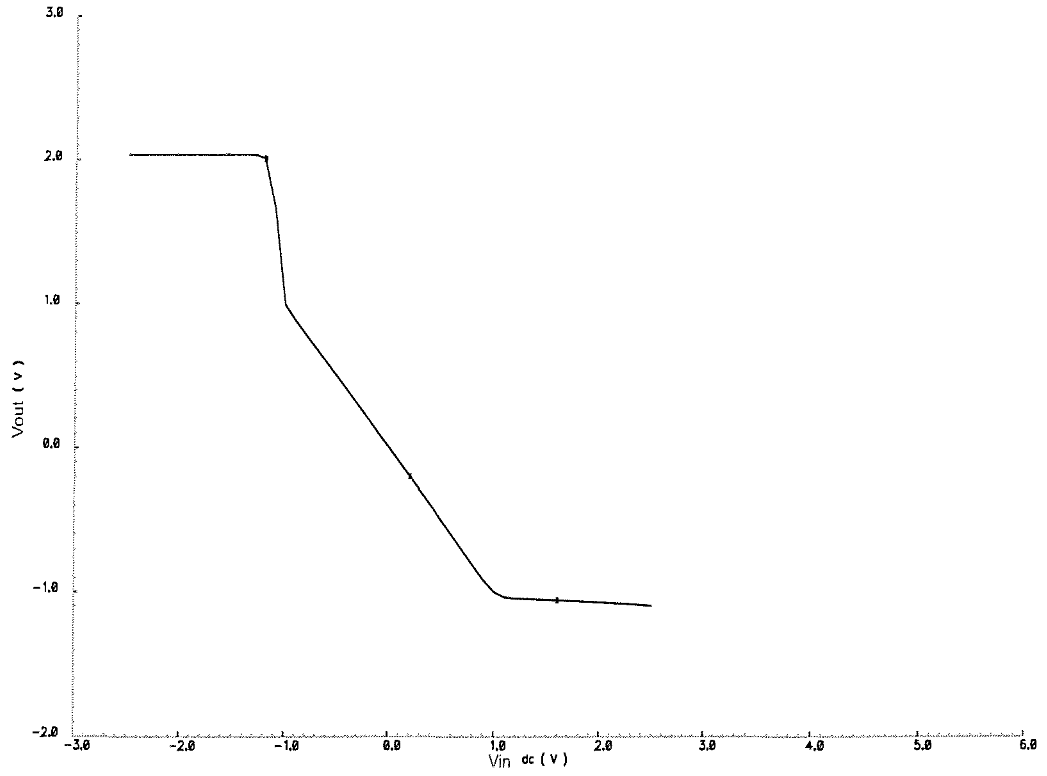


FIGURE B.15: DDVR of the DDA

Transistor	Type	W	L
M1 M2 M3 M4 M23 M24 M26	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M19	N	28um	12um
M10	N	28um	20um
M21	P	10um	0.6um
M22	N	4um	2um
M25 M27 M28	N	10um	0.6um
M15 M17	P	88um	2um
M16 M18 M20	N	28um	2um

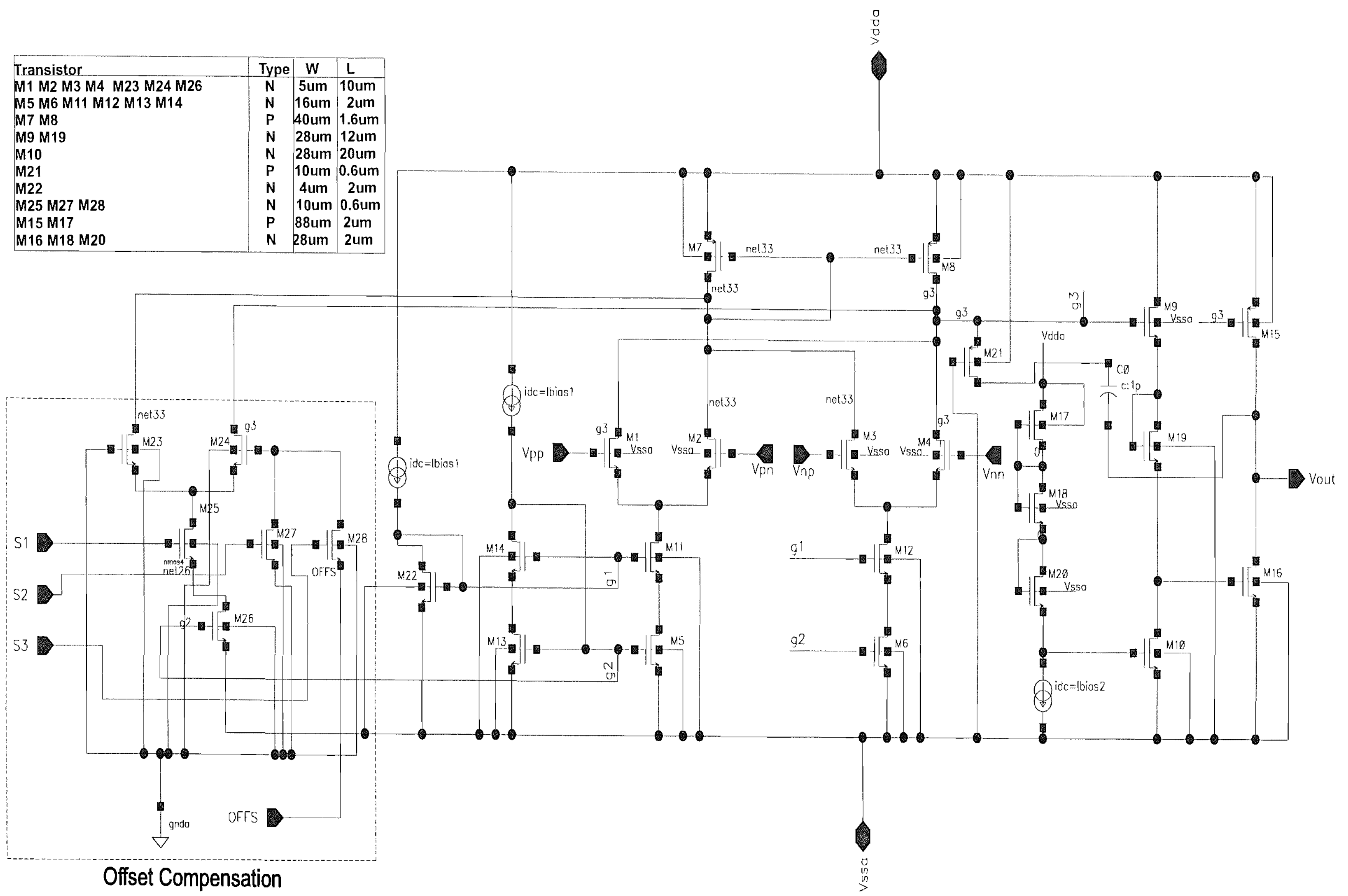
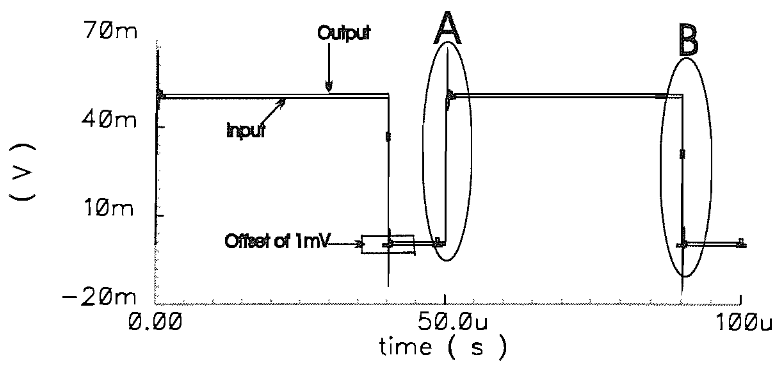
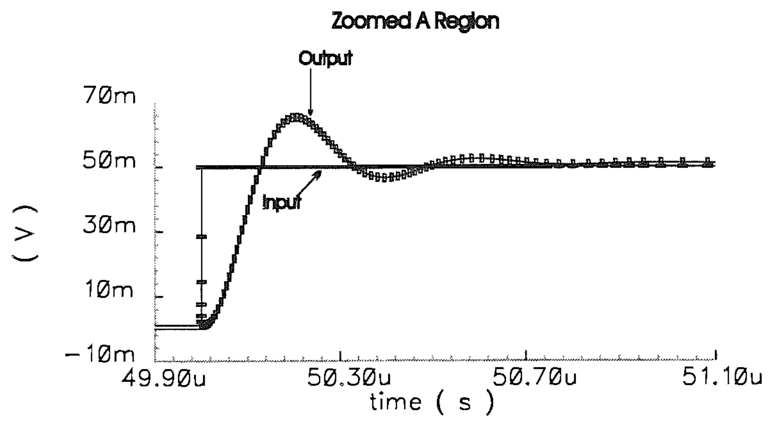


FIGURE B.16: DDA with offset compensation





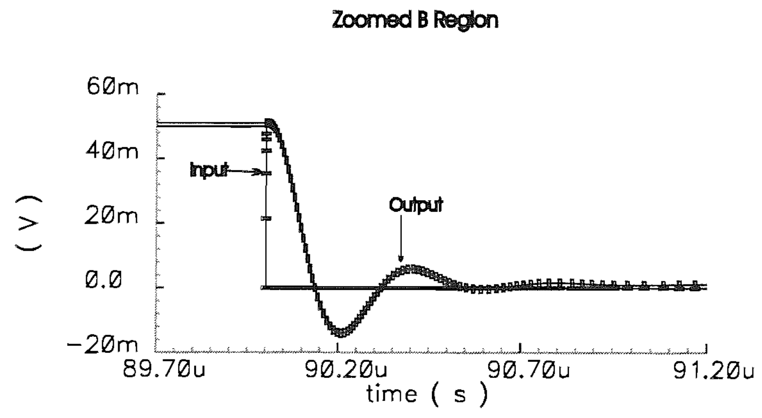
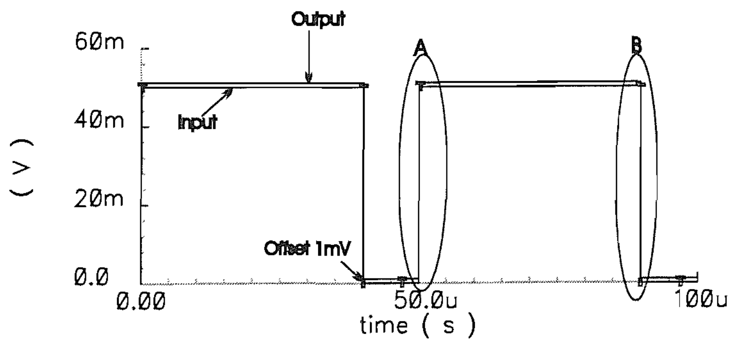
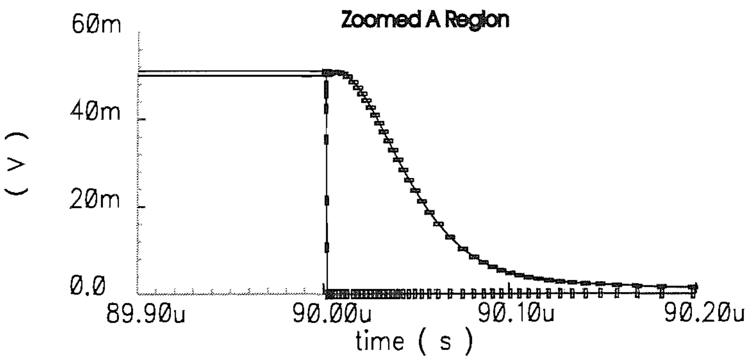


FIGURE B.17: DDA slew Response and Settling time Response for 10pF load





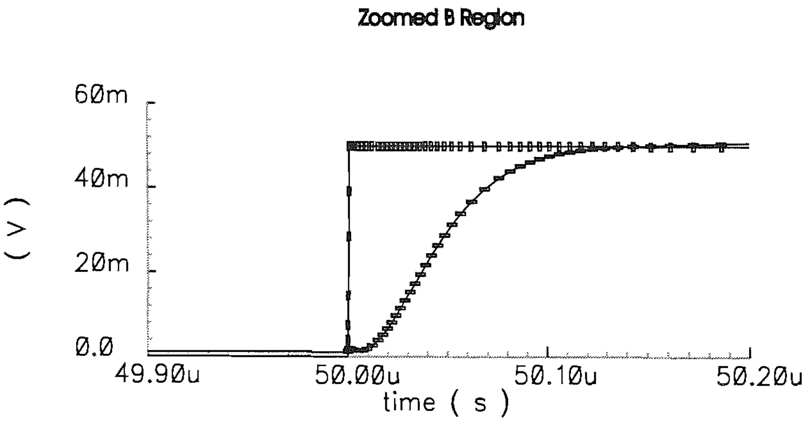


FIGURE B.18: DDA slew Response and Settling time Response for 1pF load

Transistor	Type	W	L
M1 M2 M3 M4 M23 M24 M26	N	5um	10um
M5 M6 M11 M12 M13 M14	N	16um	2um
M7 M8	P	40um	1.6um
M9 M19	N	28um	12um
M10	N	28um	20um
M21	P	10um	0.6um
M22	N	4um	2um
M25 M27 M28	N	10um	0.6um
M15 M17 M29 M31 M33 M37 M38	P	88um	2um
M16 M18 M20 M30 M32 M34 M35 M36	N	28um	2um
M39	N	7um	2um
M40	N	22um	2um

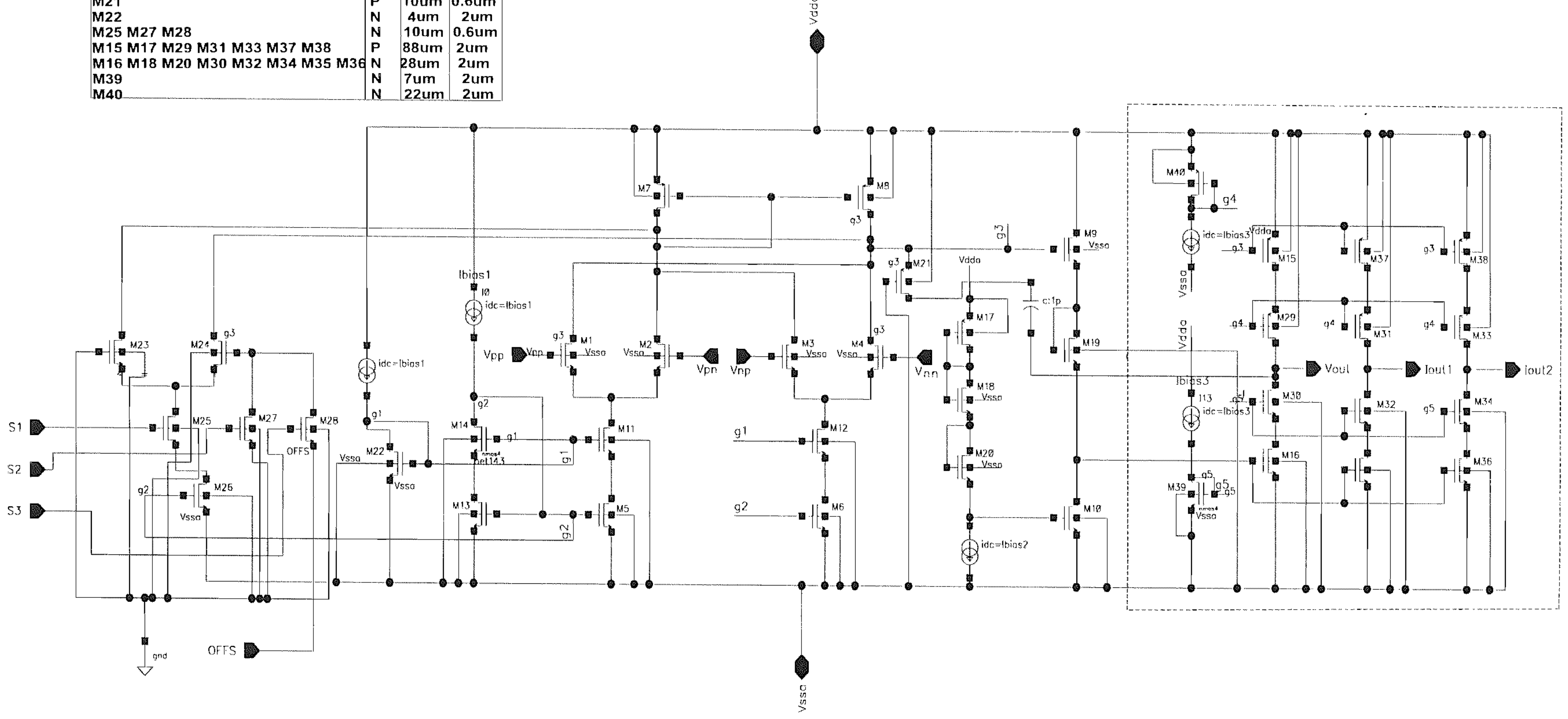


FIGURE B.19: DDA/Current Conveyor

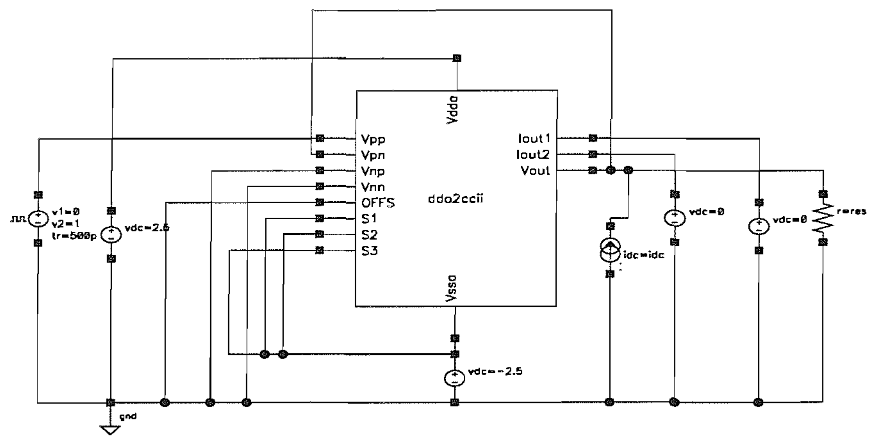


FIGURE B.20: Simulation setup for transient analysis

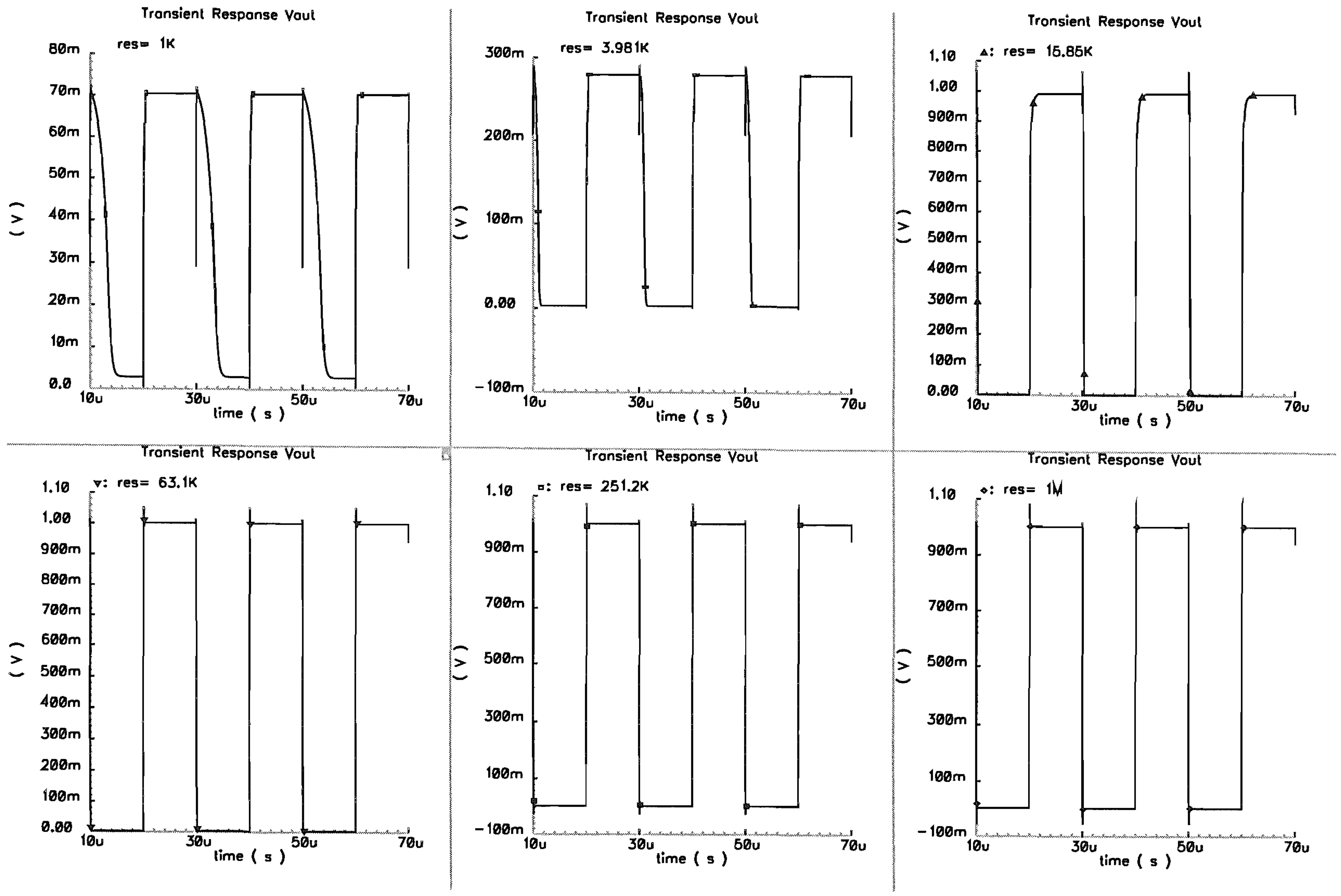


FIGURE B.21: Transient Response of the DDACCII with varying external resistance "res" 1kΩ-1MΩ

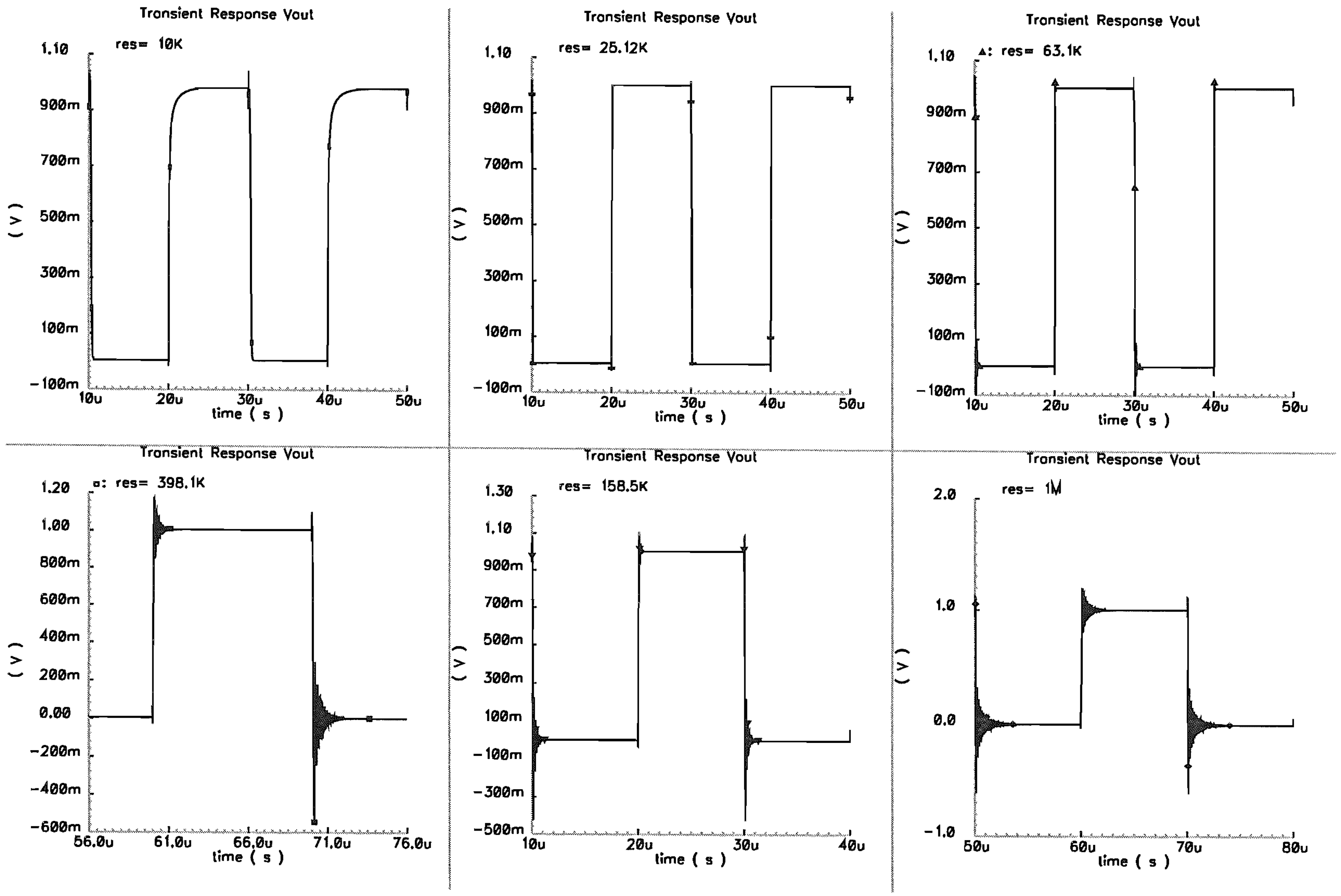
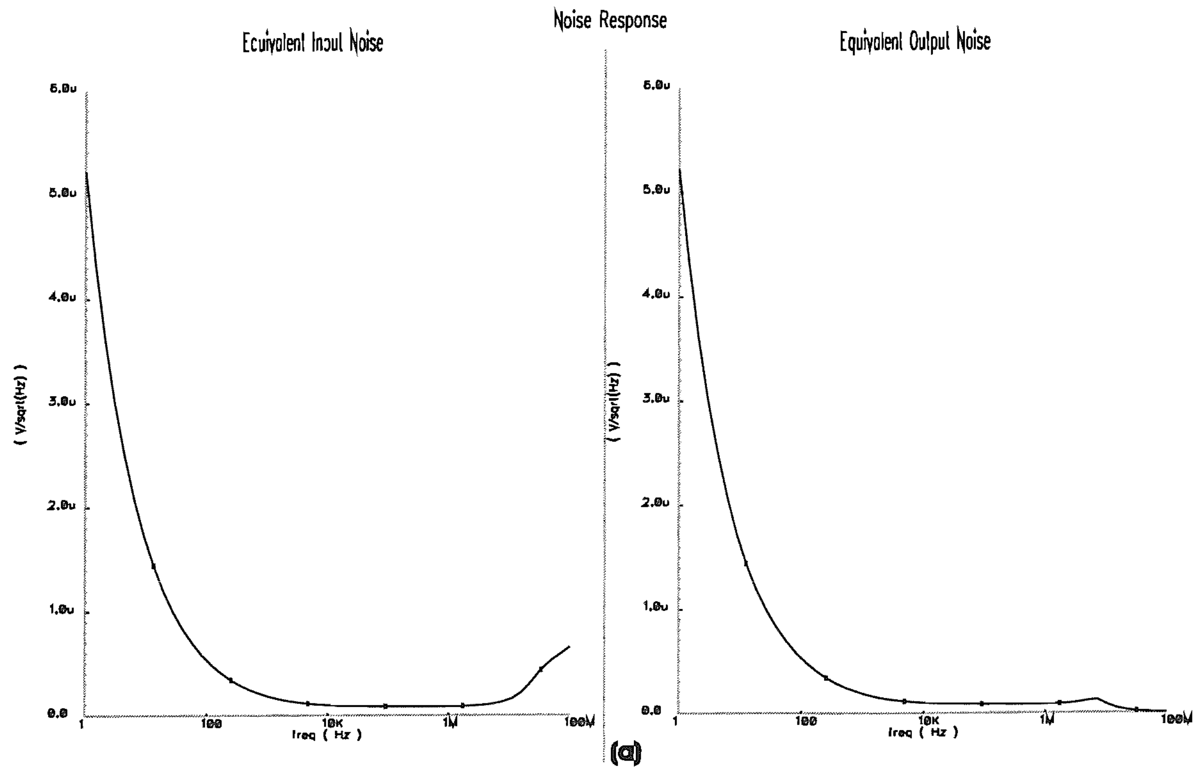


FIGURE B.22: Transient Response of the DDACCII fed with Idc=30μA and varying external resistance "res" 10kΩ-1MΩ



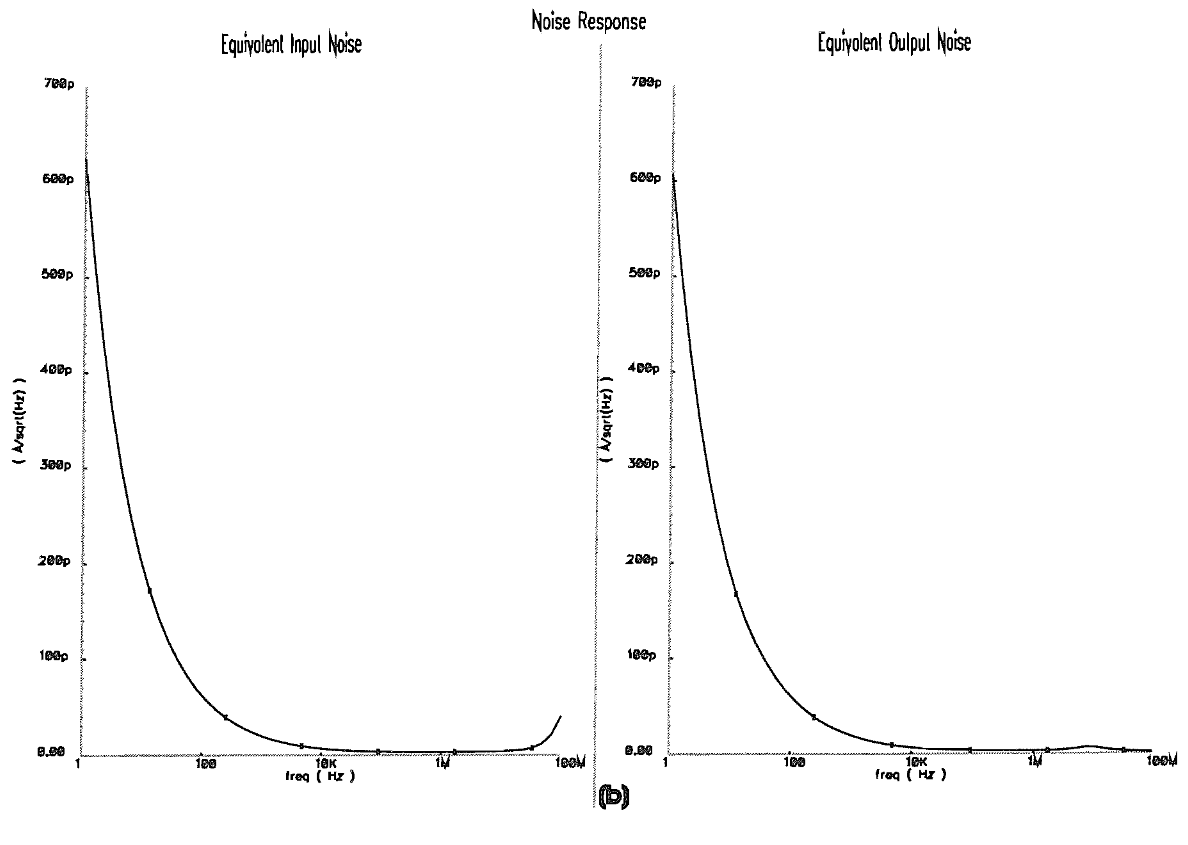


FIGURE B.23: Noise Response of DDACCII as a (a)unity gain voltage buffer, (b)current conveyor

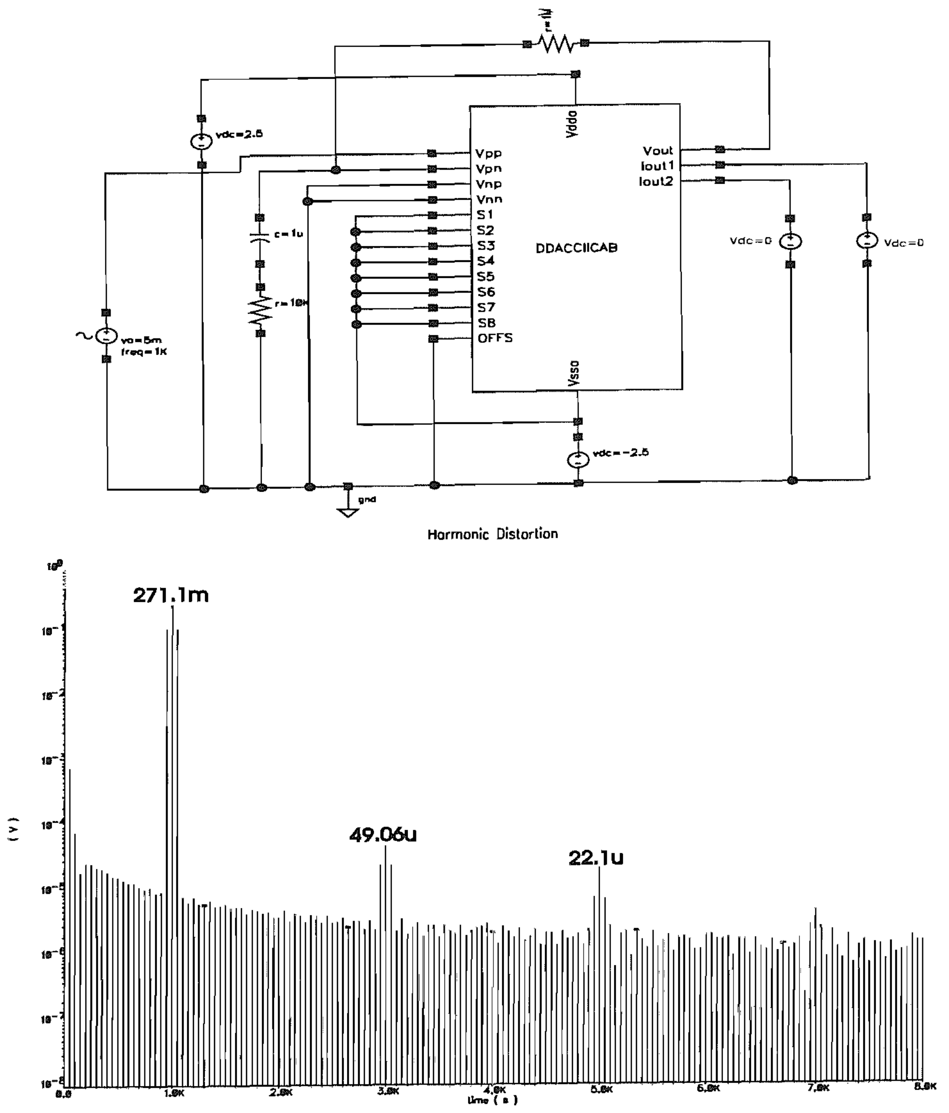


FIGURE B.24: Test circuit for harmonic distortion and its obtained response

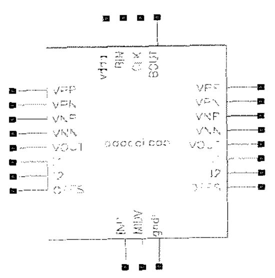


FIGURE B.25: Symbol of the final design of DDACCII CAB

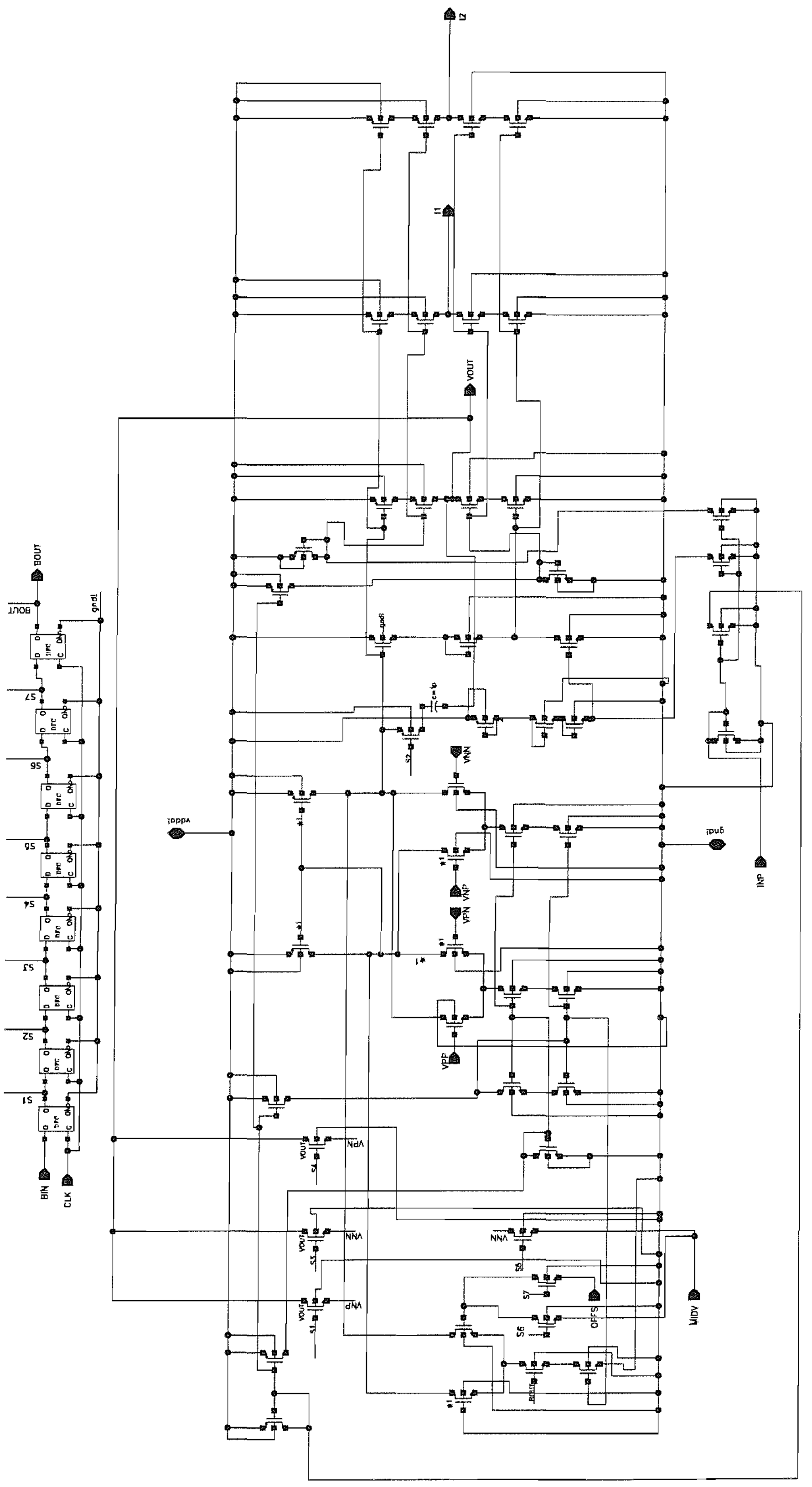


FIGURE B.26: Schematic of the final design of DDACCII CAB

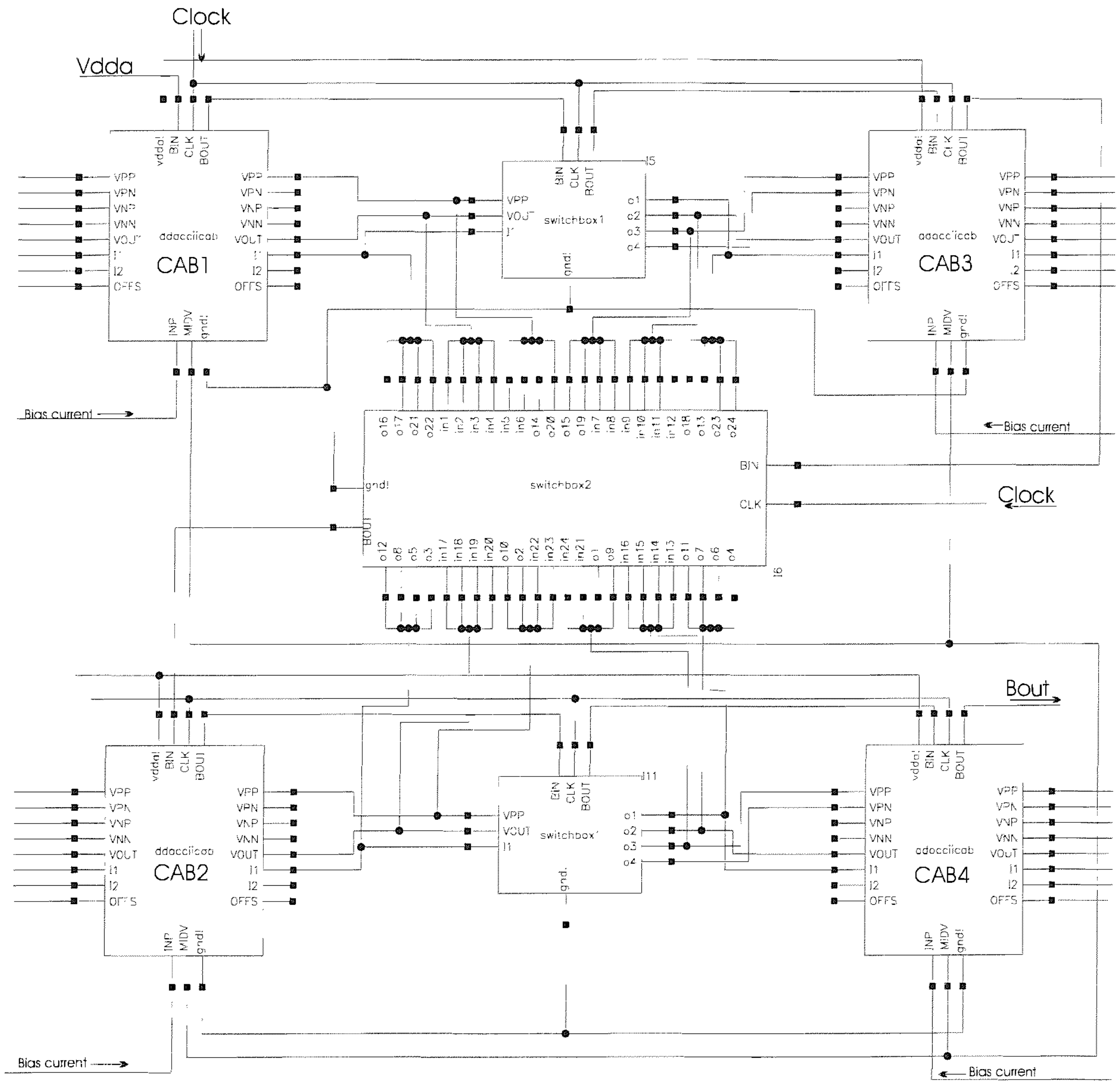


FIGURE B.27: Cluster Design

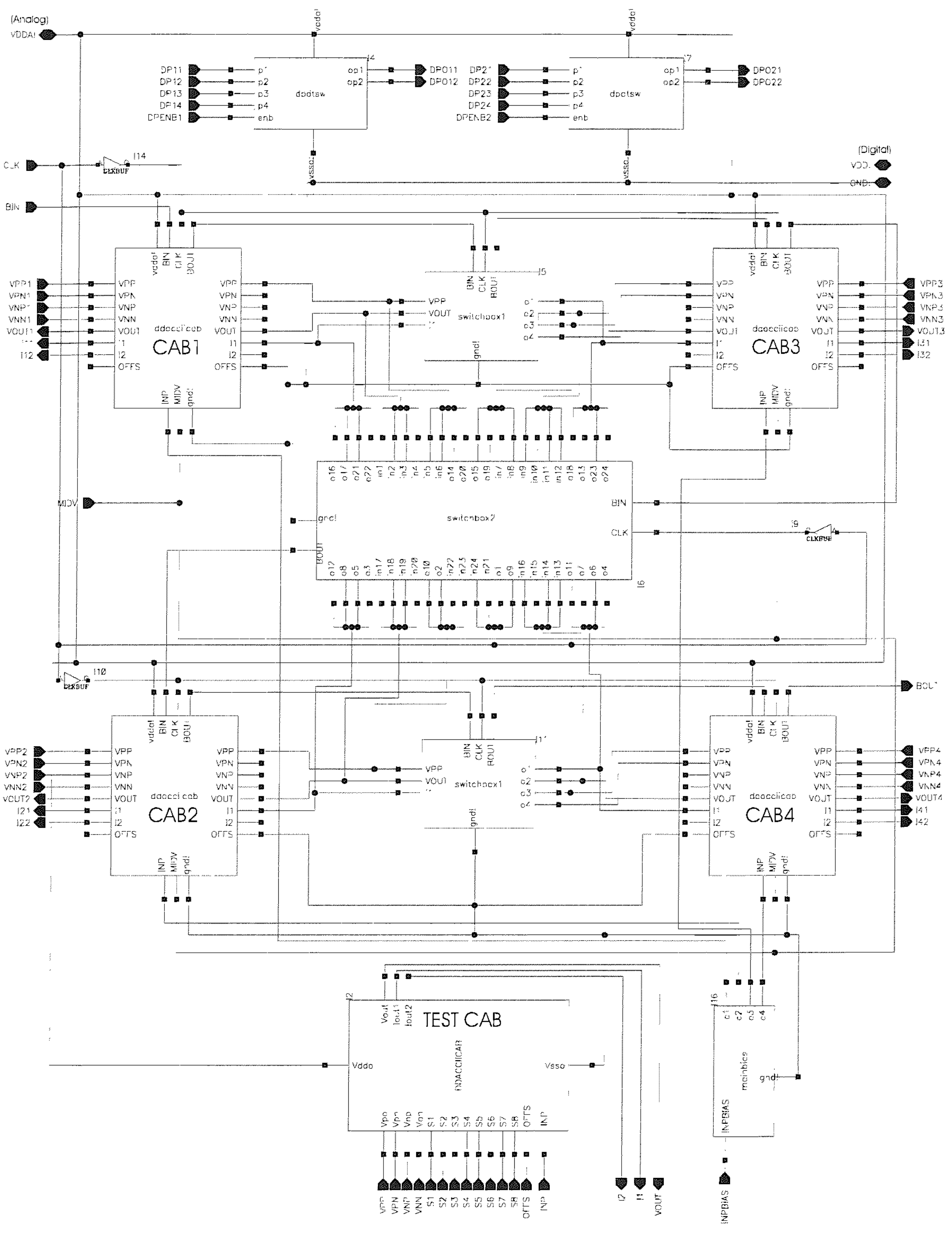


FIGURE B.28: Block level Schematic of the target FPAA chip

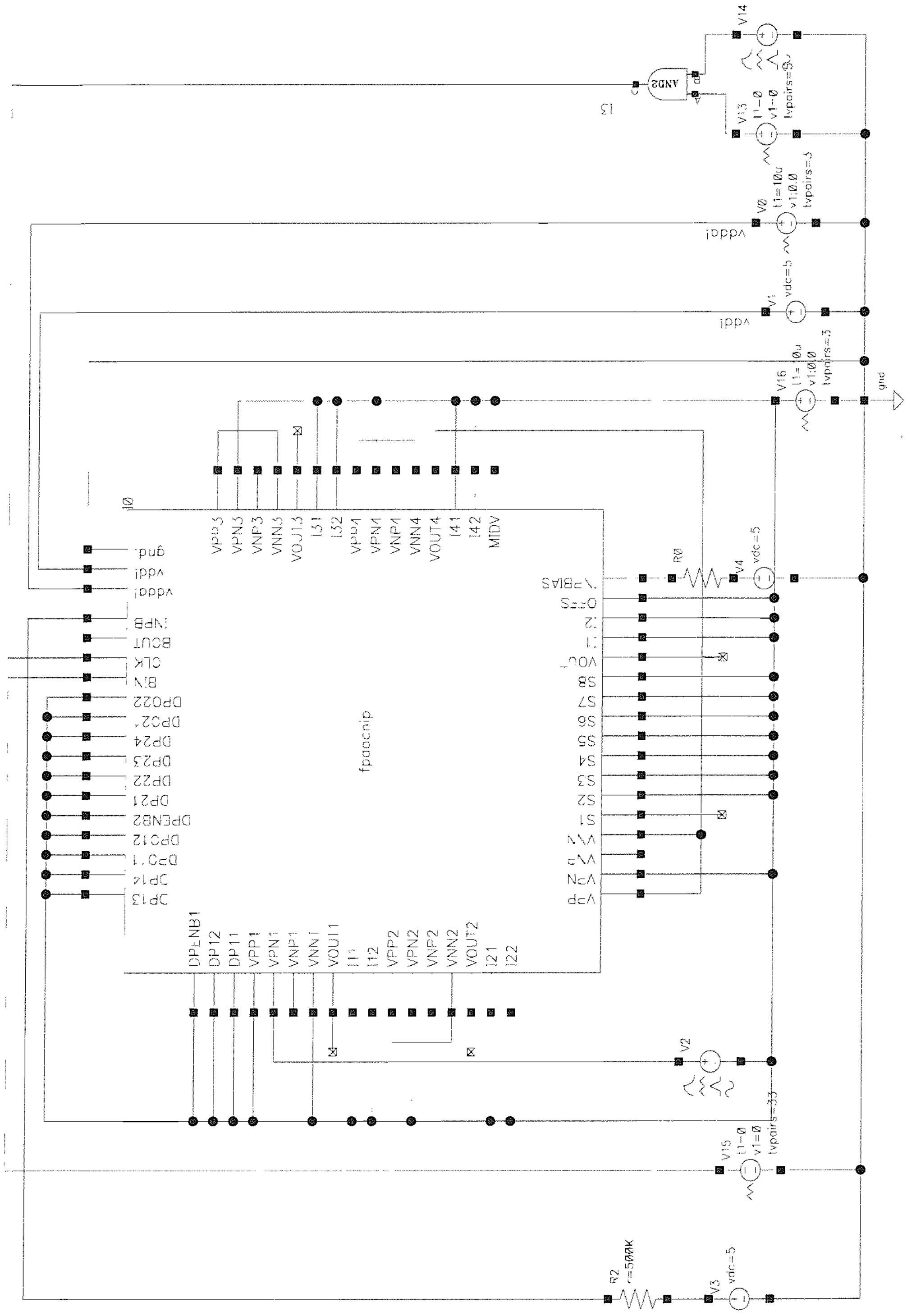


FIGURE B.29: Simulation setup for testing the FPAA chip

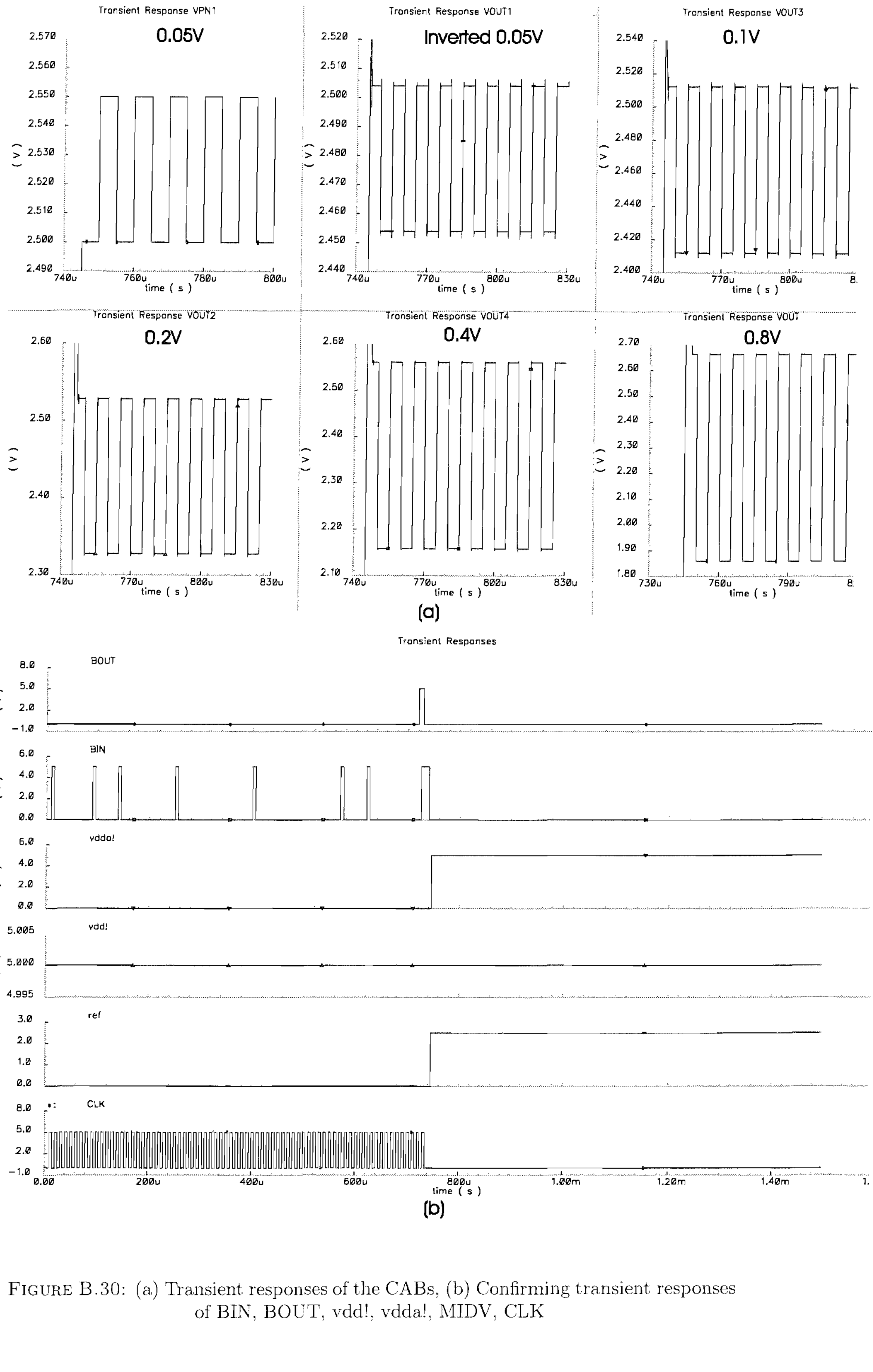


FIGURE B.30: (a) Transient responses of the CABs, (b) Confirming transient responses of BIN, BOUT, vdd!, vdda!, MIDV, CLK

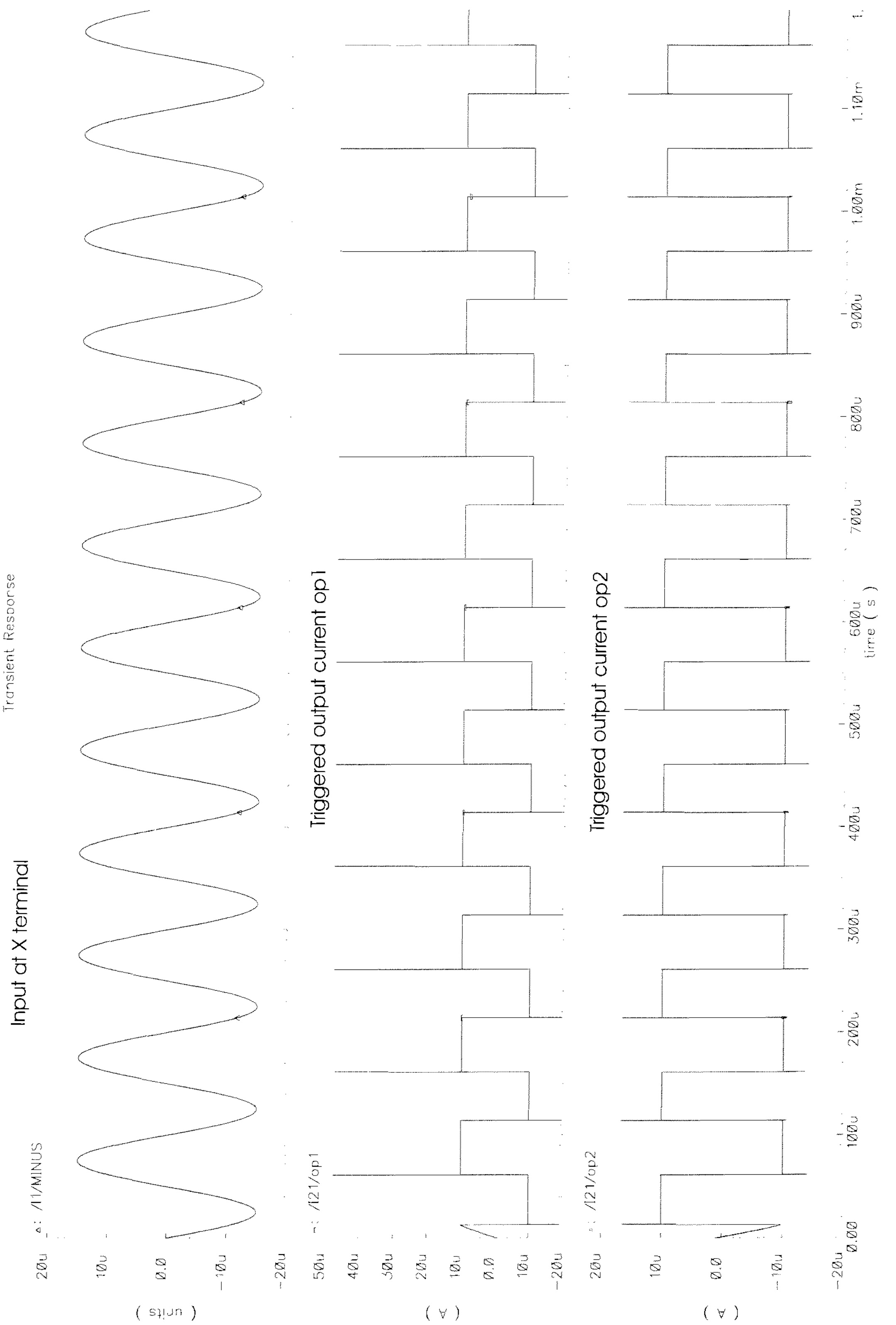


FIGURE B.31: Simulated transient responses of the Schmitt Trigger

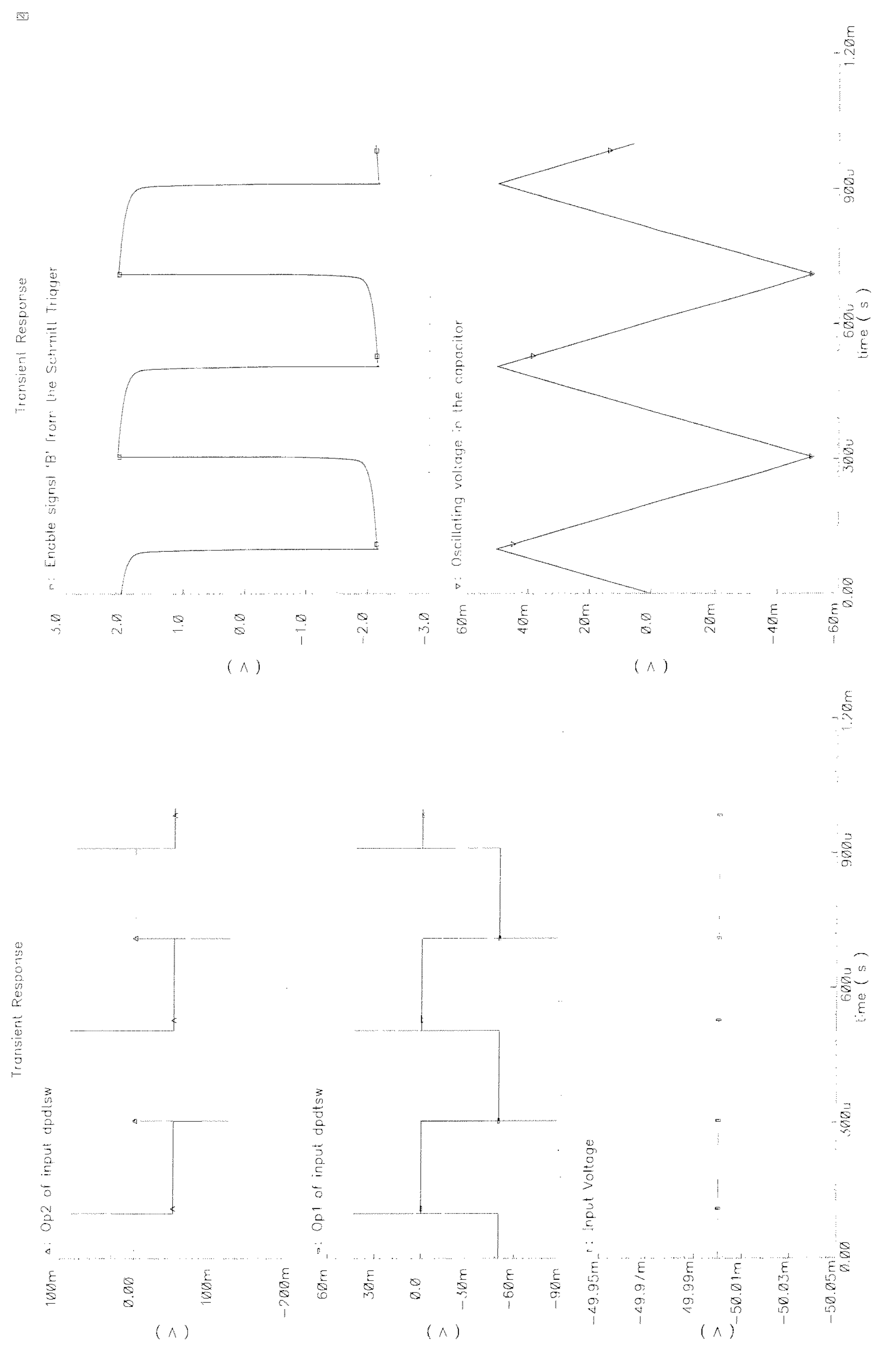


FIGURE B.32: Simulated transient responses of the VCO

Appendix C

Layouts

This Appendix contains the following:

- Plan of the chip Figure C.1
- The basic DDA Block Figure C.2
- The Final DDA Block Figure C.3
- The Switch Block S1 Figure C.4
- The Switch Block S2 Figure C.5
- The DPDTSW Block Figure C.6
- Layout of the FPAA core Figure C.7
- Layout of the chip Figure C.8

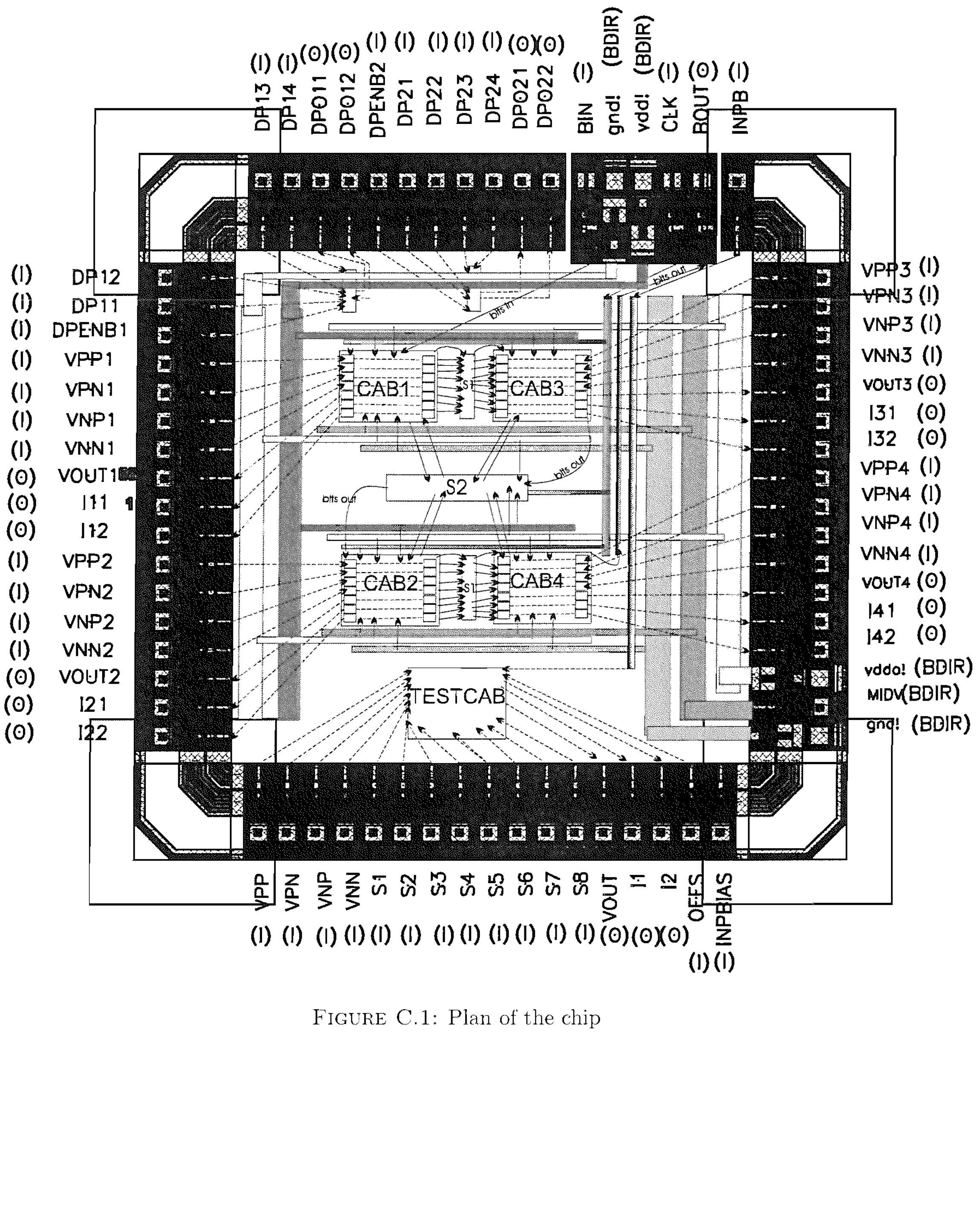


FIGURE C.1: Plan of the chip

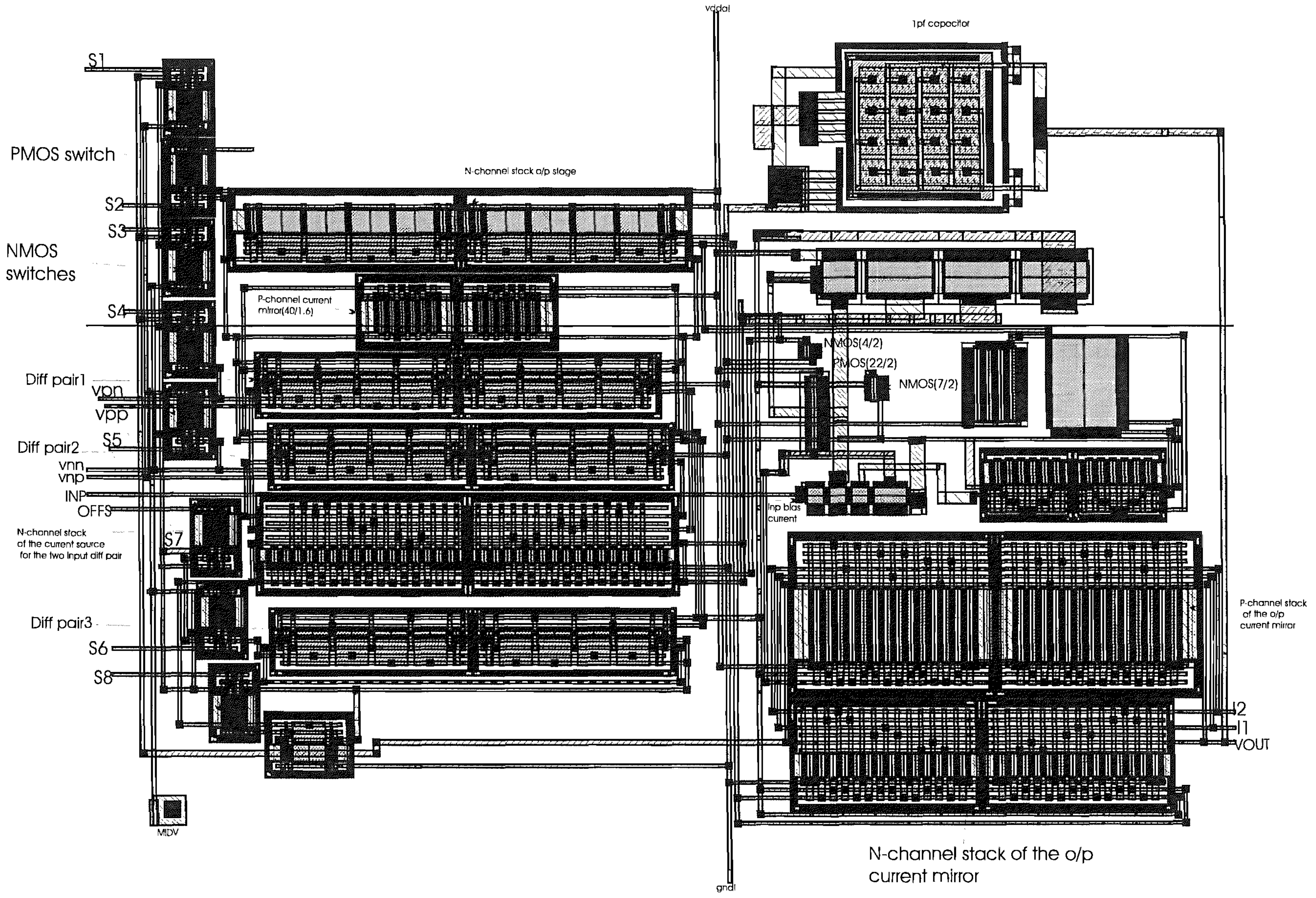

vddal
1pf capacitor
S1
PMOS switch
N-channel stack o/p stage
S2
S3
NMOS switches
P-channel current mirror(40/1.6)
S4
NMOS(4/2)
PMOS(22/2)
NMOS(7/2)
Diff pair1
vpn
vpp
Diff pair2
S5
vnn
vnp
INP
OFFS
Inp bias current
N-channel stack of the current source for the two input diff pair
S7
P-channel stack of the o/p current mirror
Diff pair3
S6
S8
I2
I1
VOUT
MIDV
N-channel stack of the o/p current mirror
gndl

FIGURE C.2: The basic DDA Block

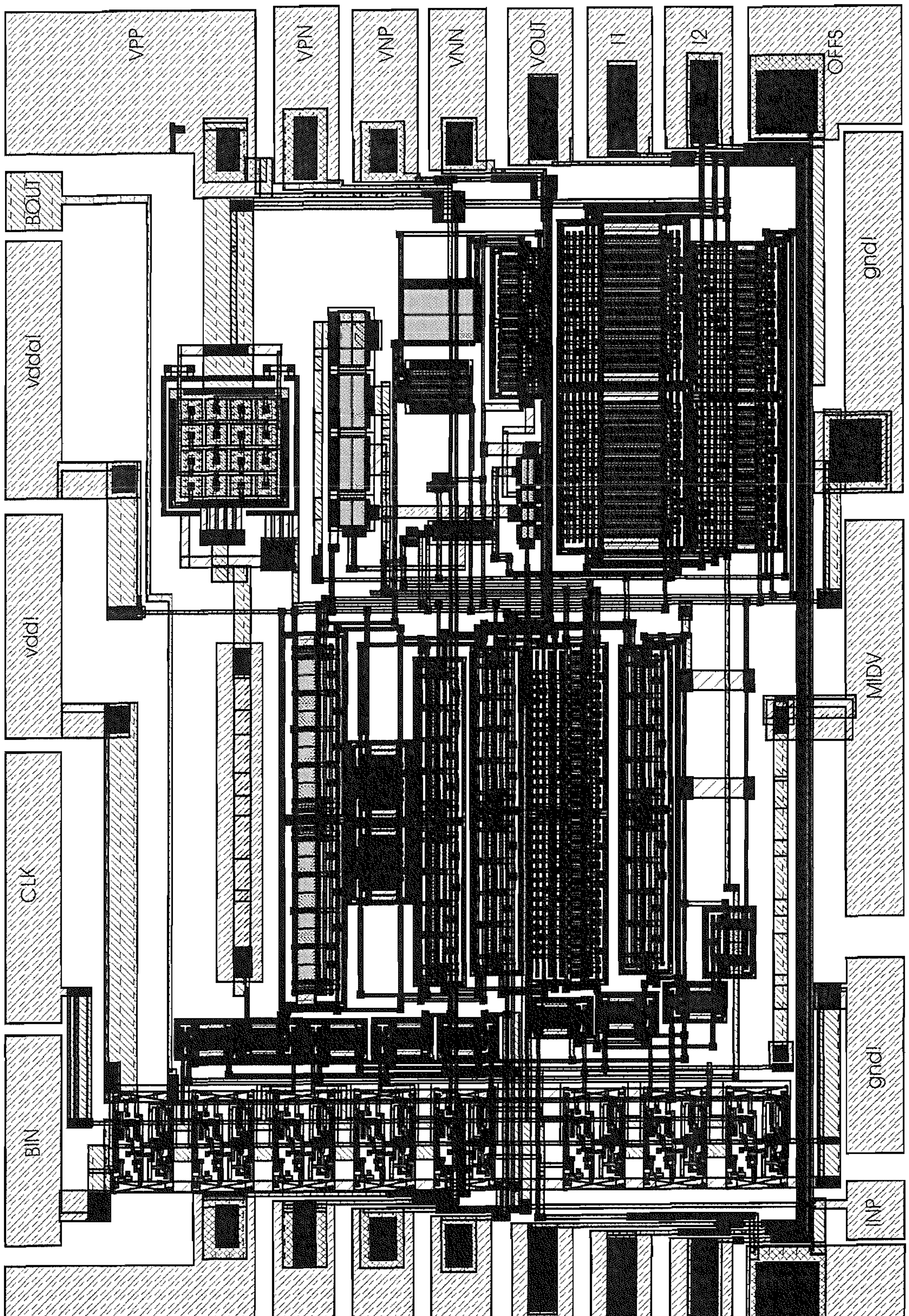


FIGURE C.3: The Final DDA Block

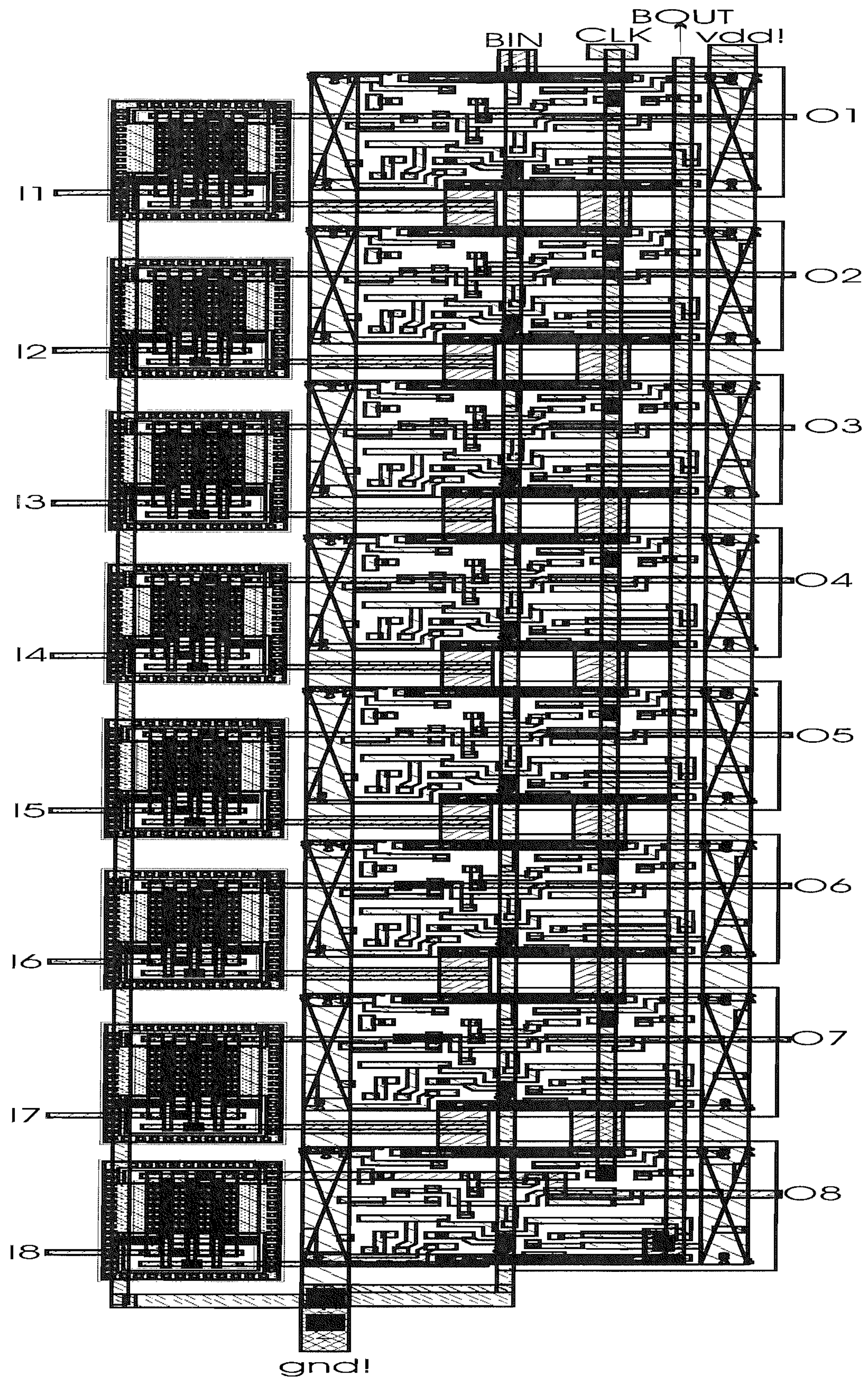


FIGURE C.4: The S1 Block

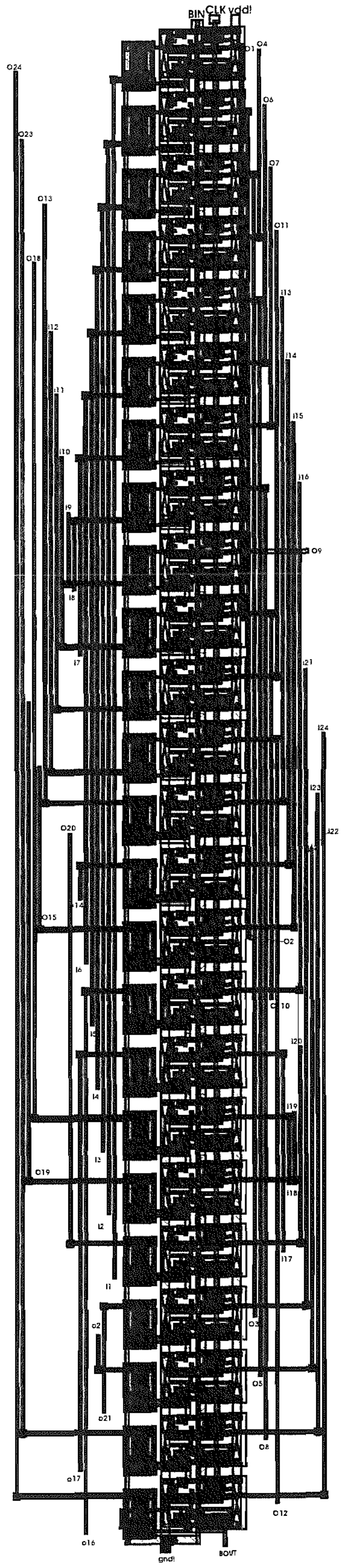


FIGURE C.5: The S2 Block

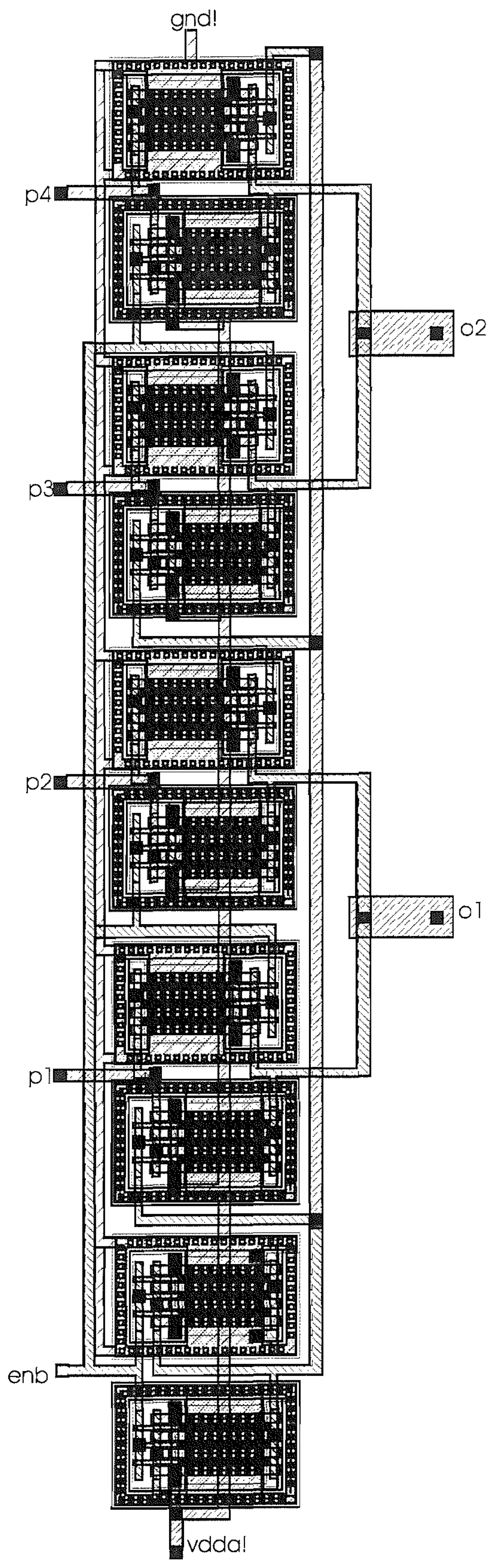


FIGURE C.6: The DPDTSW Block

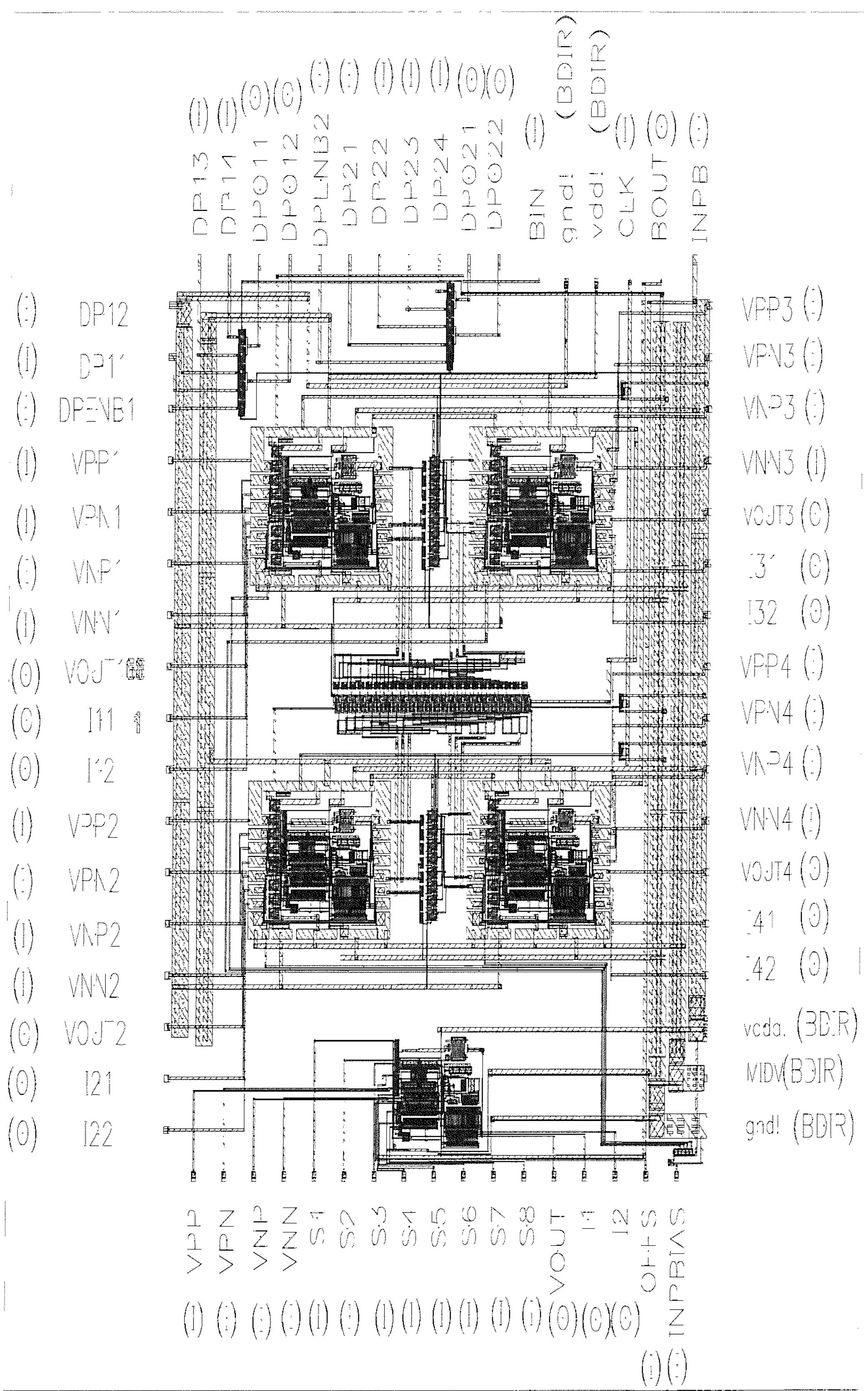


FIGURE C.7: Layout of the FPAA core

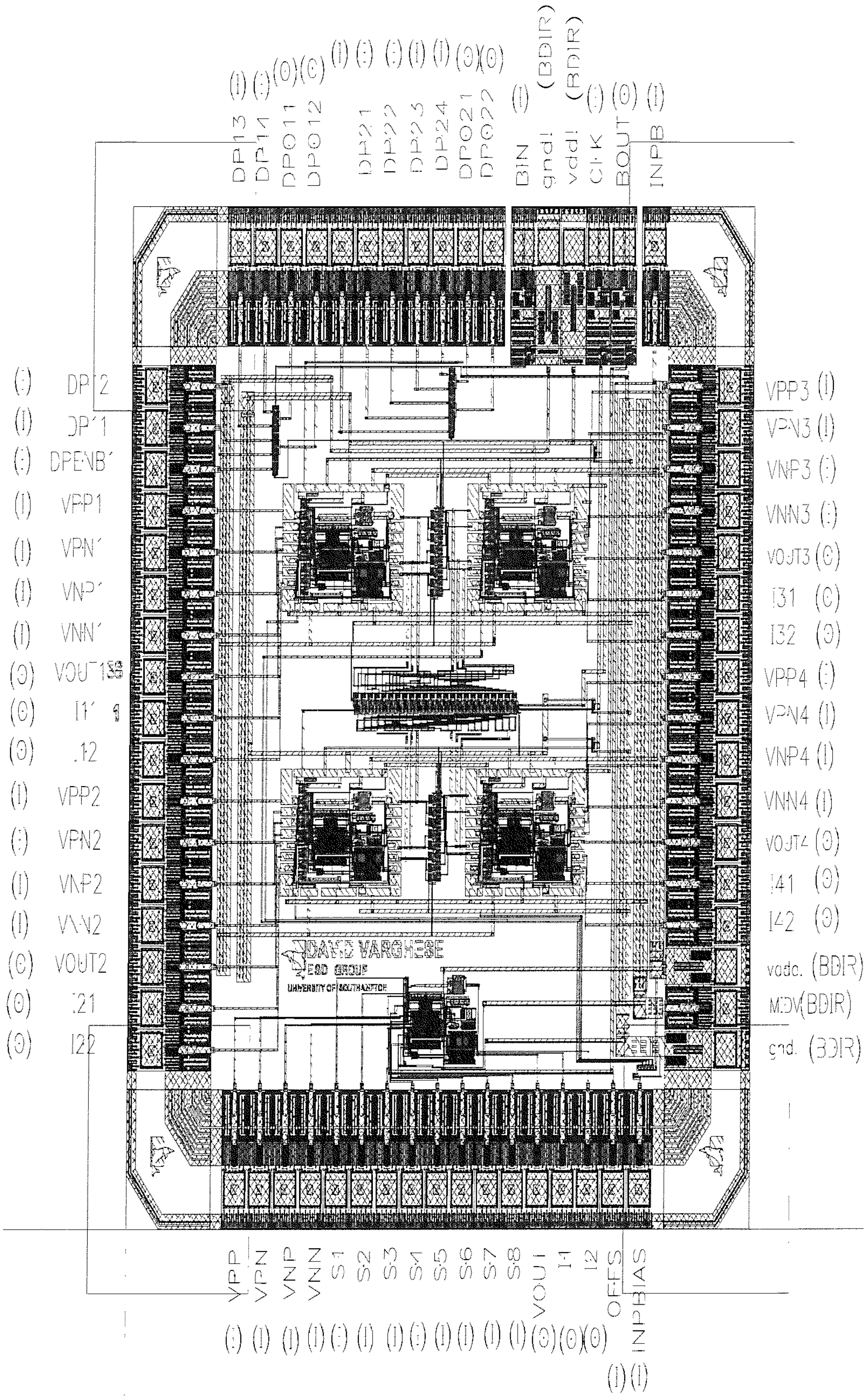


FIGURE C.8: Layout of the chip

Appendix D

Data Sheets

This appendix contains the following data sheets:

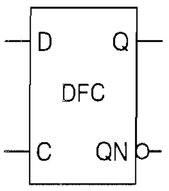
- DFC flip-flop in Figure D.1.
- Clock buffer CLKBUF is same as BU4 in Figure D.2, except for slight difference in the layout.
- PP01 (VSS) pad Figure D.3.
- PP02 (VDD) pad Figure D.4.
- PPA1P (VSSA) pad Figure D.5.
- PPA2P (VDDA) pad Figure D.6.
- IOA2P pad used mainly for all the analogue inputs/outputs Figure D.7.
- IOA5P pad used only for the MIDV input Figure D.8.
- IBD5 pad used for BIN, CLK and BOUT Figure D.9.
- Pin specifications of the chip Table D.1.
- Chip schematic Figure D.10.
- Bonding diagram of the chip Figure D.11.

AMS Austria Mikro Systeme International **DFC** CUB 0.6 μm CMOS

DFC is a fast, static, master-slave D flip-flop with 1x drive strength.

Truth Table

D	C	Q	QN
H	↑	H	L
L	↑	L	H
X	↓	no change	



Capacitance

	Ci (pF)
D	0.019
C	0.019

Area

1.35 mils^2

Power

11.59 μW/MHz

Delay [ns] = tpd.. = f(SL, L) with SL = Input Slope [ns] ; L = Output Load [pF]
Output Slope [ns] = op_sl.. = f(L) with L = Output Load [pF]

AC Characteristics : Tj = 25℃ VDD = 5V Typical Process

AC Characteristics

Characteristics	Symbol	SL = 0.1			SL = 2.0		
		L = 0.1	L = 0.7	L = 1.0	L = 0.1	L = 0.7	L = 1.0
Delay C to Q	tpdcqr	0.55	1.26	1.60	0.63	1.32	1.66
	tpdcqf	0.52	1.29	1.68	0.57	1.34	1.72
Delay C to QN	tpdcqnr	0.62	1.32	1.68	0.67	1.36	1.70
	tpdcqnf	0.70	1.45	1.84	0.78	1.54	1.91
Output Slope C to Q	op_slcqr	0.53	2.57	3.85	0.52	2.63	3.72
	op_slcqf	0.50	2.56	3.57	0.50	2.53	3.57
Output Slope C to QN	op_slcqnr	0.52	2.78	3.86	0.52	2.68	3.81
	op_slcqnf	0.47	2.35	3.56	0.45	2.50	3.52

Characteristics		Symbol	[ns]	Characteristics		Symbol	[ns]
Min D Setup Time to C	High	tsudch	0.06	Min D Hold Time to C	High	thdch	0.00
	Low	tsudcl	0.14		Low	thdcl	0.00
Min C Width	High	twch	0.22				
	Low	twcl	0.39				

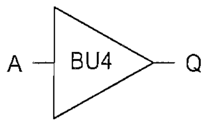
FIGURE D.1: DFC Flip-flop

AMS Austria Mikro Systeme International **BU4** CUB 0.6 µm CMOS

BU4 is a noninverting buffer with 4x drive strength.

Truth Table

A	Q
L	L
H	H



Capacitance

	Ci (pF)
A	0.025

Area

0.41 mils^2

Power

14.43 µW/MHz

Delay [ns] = tpd.. = f(SL, L) with SL = Input Slope [ns] ; L = Output Load [pF]
Output Slope [ns] = op_sl.. = f(L) with L = Output Load [pF]

AC Characteristics : Tj = 25°C VDD = 5V Typical Process

AC Characteristics

Characteristics	Symbol	SL = 0.1			SL = 2.0		
		L = 0.4	L = 2.8	L = 4.0	L = 0.4	L = 2.8	L = 4.0
Delay A to Q	tpdar	0.37	1.34	1.87	0.54	1.48	1.98
	tpdaf	0.40	1.32	1.78	0.66	1.56	2.03
Output Slope A to Q	op_slar	0.63	3.85	5.41	0.68	3.76	5.30
	op_slaf	0.52	3.01	4.25	0.60	3.02	4.25

FIGURE D.2: CLKBUF

Austria Mikro Systeme International **PP01** CUB 0.6 µm CMOS

PP01 is a pad cell for the VSS power supply.

Area

71.61 mils^2

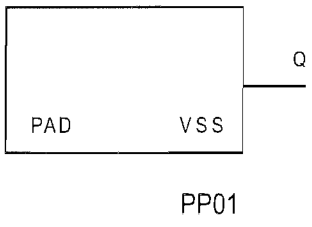


FIGURE D.3: PP01P

Austria Mikro Systeme International **PP02** CUB 0.6 µm CMOS

PP02 is a pad cell for the VDD power supply.

Area

71.61 mils^2

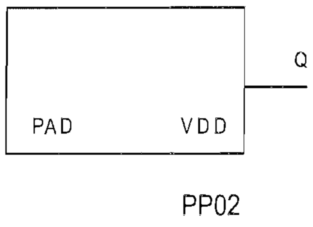


FIGURE D.4: PP02P

austriamicrosystems

0.6 µm Analog Input/Output Pad (CUB CUE CUQ)

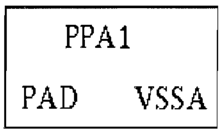
PPA1C/PPA1P

Analog Power Supply Pad
VSSA Pad

Key Features

- Pad Limited:
  Small Area 0.041mm^2
  Size x=110µm y=370µm
- Core Limited:
  Small Area 0.035mm^2
  Size x=150µm y=230µm

Symbol



Description

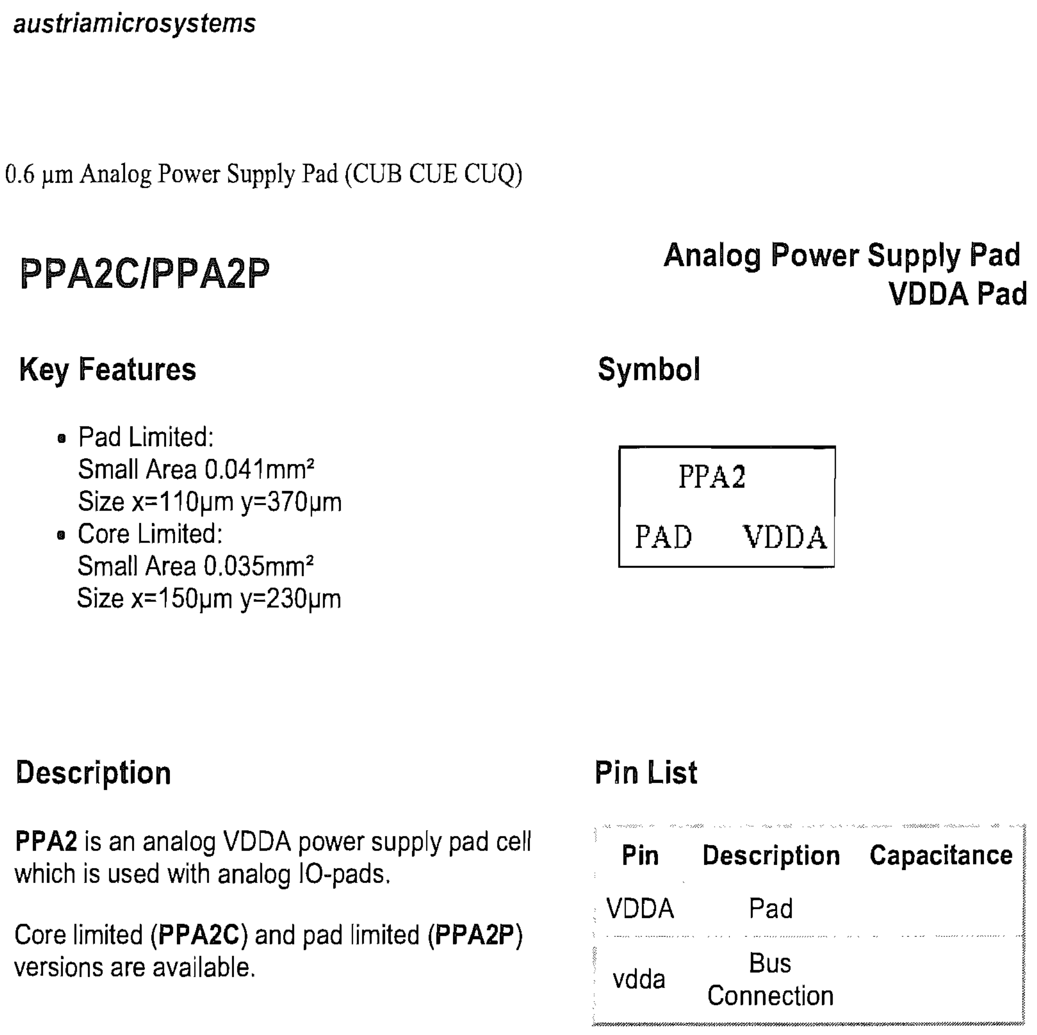
PPA1 is an analog VSSA power supply pad cell which is used with analog IO-pads.

Core limited (**PPA1C**) and pad limited (**PPA1P**) versions are available.

Pin List

Pin	Description	Capacitance
VSSA	Pad	
vssa	Bus Connection	

FIGURE D.5: PPA1P



austriamicrosystems

0.6 µm Analog Power Supply Pad (CUB CUE CUQ)

PPA2C/PPA2P

Analog Power Supply Pad
VDDA Pad

Key Features

- Pad Limited:
  Small Area 0.041mm^2
  Size x=110µm y=370µm
- Core Limited:
  Small Area 0.035mm^2
  Size x=150µm y=230µm

Symbol

PPA2
PAD VDDA

Description

PPA2 is an analog VDDA power supply pad cell which is used with analog IO-pads.

Core limited (**PPA2C**) and pad limited (**PPA2P**) versions are available.

Pin List

Pin	Description	Capacitance
VDDA	Pad	
vdda	Bus Connection	

FIGURE D.6: PPA2P

austriamicrosystems

0.6 µm Analog Input/Output Pad (CUB CUE CUQ)

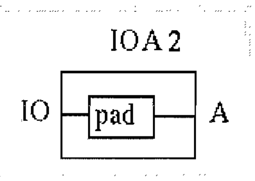
IOA2C/IOA2P

Analog Input/Output Pad
200 Ohm minimum

Key Features

- Pad Limited:
  Small Area 0.041mm^2
  Size x=110µm y=370µm
- Core Limited:
  Small Area 0.043mm^2
  Size x=185.5µm y=230µm
- Protection Resistance Included
  (p+ diff-resistor) 200 Ohm minimum

Symbol



Description

IOA2 is a general purpose analog input/output pad cell with 200 ohms series resistance.

Core limited (**IOA2C**) and pad limited (**IOA2P**) versions are available.

Pin List

Pin	Description	Capacitance
IO	Pad	4 pF
A	Input/Output	

Series Resistance

Minimum 200 Ohm
Typical 300 Ohm
Maximum400 Ohm

Note:
For this cell the CLAMP diodes to the internal power busses have to be added as specified in rule *P06G* of the **LV ESD Design Rules Document** ENG-41 by the designer. It is the responsibility of the designer to complete his design according to the ESD design rules.

FIGURE D.7: IOA2P

austriamicrosystems

0.6 μm Analog Input/Output Pad (CUB CUE CUQ)

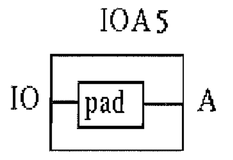
IOA5C/IOA5P

Analog Input/Output Pad
No Serial Resistor

Key Features

- Pad Limited:
  Small Area 0.041mm^2
  Size x=110μm y=370μm
- Core Limited:
  Small Area 0.043mm^2
  Size x=185.5μm y=230μm

Symbol



Description

IOA5 is an analog input/output pad cell with zero ohms series resistor.

Core limited (**IOA5C**) and pad limited (**IOA5P**) versions are available.

Pin List

Pin	Description	Capacitance
IO	Pad	4 pF
A	Input/Output	

Note:
This cell is not complete in terms of **ESD protection** due to the missing RBLOCK resistor and CLAMP diodes as defined in the **LV ESD Design Rules Document** ENG-41. It is the responsibility of the designer to complete his design according to the ESD design rules.

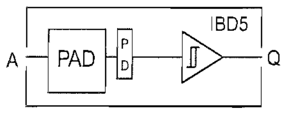
FIGURE D.8: IOA5P

AMS Austria Mikro Systeme International **IBD5** CUB 0.6 µm CMOS

IBD5 is a Schmitt-Trigger input buffer pad with input voltage hysteresis.

Truth Table

A	Q
L	L
H	H



Capacitance

	Ci (pF)
A	0.766

Area

71.61 mils^2

Power

29.20 µW/MHz

Delay [ns] = tpd.. = f(SL, L) with SL = Input Slope [ns] ; L = Output Load [pF]
Output Slope [ns] = op_sl.. = f(L) with L = Output Load [pF]

AC Characteristics : Tj = 25°C VDD = 5V Typical Process

AC Characteristics

Characteristics	Symbol	SL = 0.1			SL = 2.0		
		L = 0.2	L = 1.4	L = 2.0	L = 0.2	L = 1.4	L = 2.0
Delay A to Q	tpdar	0.92	1.40	1.60	1.48	1.98	2.19
	tpdaf	1.21	1.65	1.85	1.73	2.19	2.37
Output Slope A to Q	op_slar	0.61	1.87	2.55	0.62	1.90	2.50
	op_slaf	0.68	1.57	2.01	0.68	1.58	1.97

FIGURE D.9: IBD5

Pin No:	Pin Name	Pin Type	Pin No:	Pin Name	Pin Type
1	VPN4	Input	35	I11	Output
2	VPP4	Input	36	I12	Output
3	I32	Output	37	VPP2	Input
4	I31	Output	38	VPN2	Input
5	VOUT3	Output	39	VNP2	Input
6	VNN3	Input	40	VNN2	Input
7	VNP3	Input	41	VOUT2	Output
8	VPN3	Input	42	I21	Output
9	VPP3	Input	43	I22	Output
10	INP	Input	44	VPP	Input
11	BOUT	Output	45	VPN	Input
12	CLK	Input	46	VNP	Input
13	vdd!	BDIR	47	VNN	Input
14	gnd!	BDIR	48	S1	Input
15	BIN	Input	49	S2	Input
16	DPO22	Output	50	S3	Input
17	DPO21	Output	51	S4	Input
18	DP24	Input	52	S5	Input
19	DP23	Input	53	S6	Input
20	DP22	Input	54	S7	Input
21	DP21	Input	55	S8	Input
22	DPENB2	Input	56	VOUT	Output
23	DPO12	Output	57	I1	Output
24	DPO11	Output	58	I2	Output
25	DP14	Input	59	OFFS	Input
26	DP13	Input	60	INPBIAS	Input
27	DP12	Input	61	gnd!	BDIR
28	DP11	Input	62	MIDV	BDIR
29	DPENB1	Input	63	Vdda!	BDIR
30	VPP1	Input	64	I42	Output
31	VPN1	Input	65	I41	Output
32	VNP1	Input	66	VOUT4	Output
33	VNN1	Input	67	VNN4	Input
34	VOUT1	Output	68	VNP4	Input

TABLE D.1: Pins of the FPAA chip

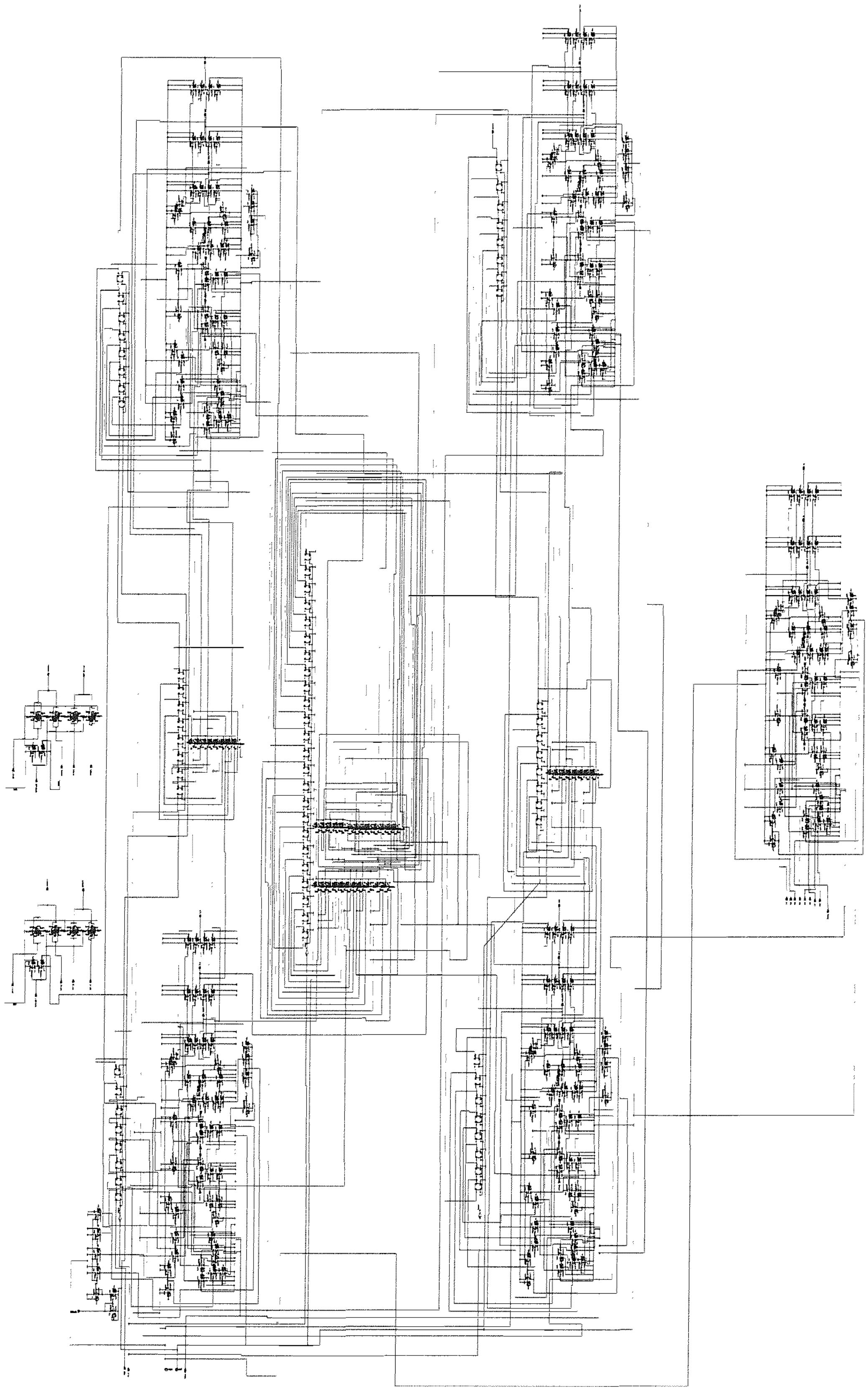
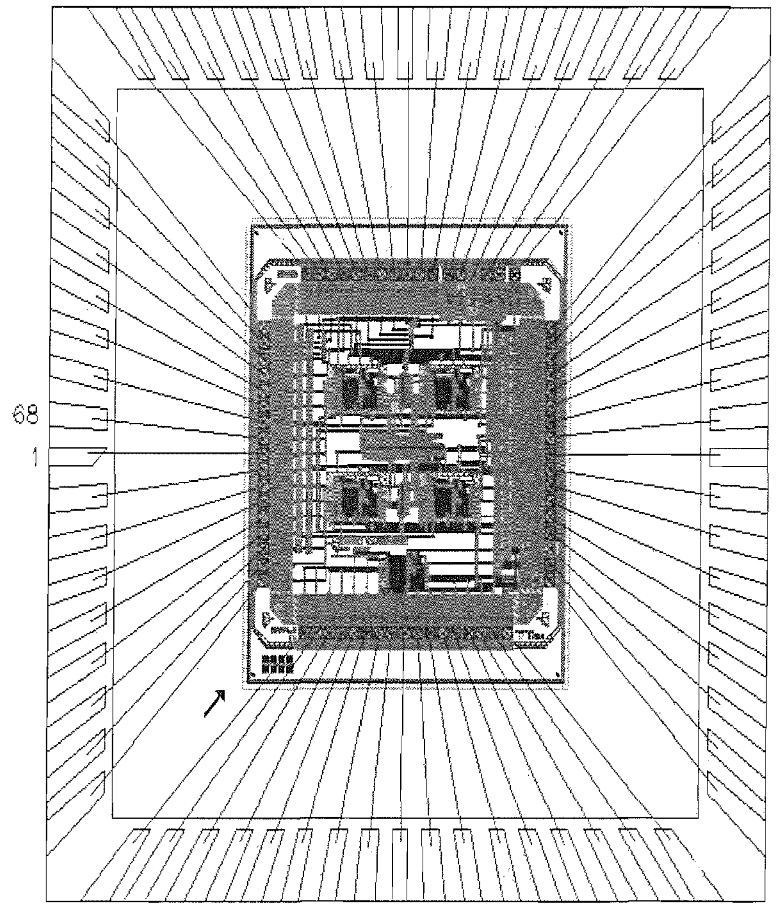


FIGURE D.10: Schematic of the target FPAA chip

Remarks ↗ Fleche vers le logo "A6ØC4_2 FØ"	JLCC68



A6ØC4_2 FØ

Run A6ØC4_2	Date 25 mai 2ØØ4	Scale 1Ø
Die FØ (DAVIDESDSOTON)	Macrodie AMS1Ø541	
Sample Qty 1Ø	Wire	
Size	Lid removable ☒ Sealed ☐	
Die Attach		

FIGURE D.11: Bonding Diagram, of the FPAA chip

Appendix E

Codes written in C++

E.1 Program 1

```
/******************************************
 DDA Connecting Strategies & Visualisation

Coded and Compiled by
David Varghese
ESD Group
University of Southampton
********************************************/
#include <iostream>
#include <vector>
#include <map>
#include <fstream>
#include <algorithm>
#include <string>

using namespace std;

typedef vector<char> t_vector_char;
typedef vector<int> t_vector_int;
typedef map<int, t_vector_int> t_map_intvect;
typedef vector<t_vector_int> tconnections;
struct sw_connect{
  sw_connect(int id, int p, int a): ddaid(id), pos(p), active(a) {};
  int ddaid;
  int pos;
  int active;
};
struct subcluster{
  subcluster(int id1, int id2, int id3): ddaid1(id1),
  ddaid2(id2), switchid_bw_id1id2(id3) {};
  int ddaid1;
  int ddaid2;
  int switchid_bw_id1id2;
};
struct cluster{
  cluster(int iden1,int iden2,int iden3): subcluster_id1(iden1),
  subcluster_id2(iden2), swid_bw_id1id2(iden3){};
  int subcluster_id1;
  int subcluster_id2;
  int swid_bw_id1id2;
};

typedef vector<sw_connect> t_vector_switch;
typedef map<int, t_vector_switch> t_map_intvect_switch;
typedef vector<t_map_intvect_switch> vector_map_switches;
typedef vector<subcluster> t_vector_subcluster;
typedef map<int, t_vector_subcluster> t_map_intvect_subcluster;
typedef vector<cluster> t_vector_cluster;
typedef map<int, t_vector_cluster> t_map_intvect_cluster;
t_vector_int function1(int ddanum);
t_map_intvect_switch function2(t_vector_int dda,
int source_ddaid,int target_ddaid);
int function3(vector_map_switches SV,t_vector_int dda,
t_map_intvect_subcluster SBCLUSMAP,vector_map_switches SV2,
t_map_intvect_cluster CLUSMAP);
int function4(t_map_intvect_subcluster SBCLUSMAP);
t_map_intvect_switch function5(t_vector_int dda,int first_subcluster,
int second_subcluster,t_map_intvect_subcluster SBCLUSMAP);
int function6(vector_map_switches SV2,t_vector_int dda);

t_vector_int function1(int ddanum) {
int ddanumber = ddanum;
cout << ddanumber << endl;
t_vector_int dda(9*ddanumber);
t_vector_int::iterator ddaiter = dda.begin();
t_vector_int::iterator ddaiter2 = dda.begin();
ddaiter2++;
int q = 1;
for(;ddaiter != dda.end(); ddaiter=ddaiter+9) {
if (q <= ddanumber)  *ddaiter = q;
for(int i=1; i<=8; i++){
*ddaiter2 = i;
ddaiter2++;
}
ddaiter2++;
q++;
}
ddaiter = dda.begin();
for(;ddaiter != dda.end(); ddaiter++) {
cout << "dda values = " << *ddaiter << "\n\n";
}
cout << endl;
return dda;
}

t_map_intvect_switch function2(t_vector_int dda,int source_ddaid,
int target_ddaid){
t_map_intvect_switch S1; // creating switching map
int src_id,tar_id;
src_id = (source_ddaid - 1) * 9;
tar_id =  (target_ddaid - 1) * 9;
for (int j=0;j<8;j++) {
if (j==1) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+1],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+5],0));
```

```
}
if (j==2) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+1],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+6],0));
}
if (j==3) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+5],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+1],0));
}
if (j==4) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+5],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+6],0));
}
if (j==5) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+6],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+1],0));
}
if (j==6) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+6],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+2],0));
}
if (j==7) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+6],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+6],0));
}
if (j==0) {
S1[j].push_back(sw_connect(dda[src_id],dda[src_id+1],0));
S1[j].push_back(sw_connect(dda[tar_id],dda[tar_id+1],0));
}
}
t_map_intvect_switch::iterator switer = S1.begin();
for(; switer!= S1.end(); switer++) {
cout << "Switch" << switer->first << "-> dda pos ";
t_vector_switch &vector_of_connectors = switer->second;
for (int i=0; i< vector_of_connectors.size(); i++)
{  cout << "(" << vector_of_connectors[i].ddaid << ","
<< vector_of_connectors[i].pos << ","
<< vector_of_connectors[i].active << ")";}
cout << endl;
}
return S1;
}
int function3(vector_map_switches SV,t_vector_int dda,
t_map_intvect_subcluster SBCLUSMAP,vector_map_switches SV2,
t_map_intvect_cluster CLUSMAP){
ofstream fout("test5.dot");
fout << "digraph test { \n"; fout << "ranksep = 5;\n";
fout << "size=" <<'"' << "50,50" <<'"'<<"\n";
fout << "node [shape = box] ;\n";
t_map_intvect_switch S1;
vector_map_switches SV1;
t_map_intvect_subcluster SBMAP1;
t_map_intvect_cluster CBMAP1;
t_vector_int switchclustervalu2;
CBMAP1=CLUSMAP;
t_map_intvect_cluster::iterator clusmapiter1 = CBMAP1.begin();
for (; clusmapiter1 != CBMAP1.end(); clusmapiter1++){
fout << "subgraph cluster" << "00000" << clusmapiter1->first << "{\n\n";
fout << "label = " << '"'<< "cluster00000" <<
```

```
clusmapiter1->first << '"'<<";"<<"\n\n";
t_vector_cluster &vector_of_cluster_contents1 = clusmapiter1->second;
for (int s=0; s< vector_of_cluster_contents1.size(); s++){
fout << "subgraph cluster" <<"0000"<<
vector_of_cluster_contents1[s].subcluster_id1 <<";" << endl;
fout << "subgraph cluster" <<"0000"<<
vector_of_cluster_contents1[s].subcluster_id2 <<";" << endl;
fout << "subgraph cluster" <<"sw0000"<<
vector_of_cluster_contents1[s].swid_bw_id1id2 << ";" << endl;
switchclustervalu2.push_back(vector_of_cluster_contents1[s].swid_bw_id1id2);
}
fout << " }\n\n"<<endl;
}
/***************************************************************************/
t_map_intvect_switch S3;
vector_map_switches SV3;
int switchclusterval = 0;
SV3 = SV2;
vector_map_switches ::iterator  switchiterC = SV3.begin();
for (; switchiterC != SV3.end(); switchiterC++){
if (switchclusterval < switchclustervalu2.size()){
S3 = *switchiterC;
t_map_intvect_switch::iterator switer3 = S3.begin();
fout << "subgraph cluster"<<"sw0000"<< switchclusterval<<"{ \n";
fout << "{rank=same;";
for(; switer3!= S3.end(); switer3++)
fout << "switch"<<switchclusterval<<switer3->first << ";";
fout << "}\n"; fout << "}\n";
switer3 = S3.begin();
for(; switer3!= S3.end(); switer3++) {
t_vector_switch &vector_of_connectors = switer3->second;
for (int q=0; q< vector_of_connectors.size(); q++)
{
if (q!=1){
~~if (vector_of_connectors[q].pos == 1)~~
fout << "vpp" << vector_of_connectors[q].ddaid << "->"
<< "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 2)
fout << "vpn" << vector_of_connectors[q].ddaid << "->"
<< "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 3)
fout << "vnp" << vector_of_connectors[q].ddaid << "->"
<< "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 4)
fout << "vnn" << vector_of_connectors[q].ddaid << "->"
<< "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 5)
fout << "vo" << vector_of_connectors[q].ddaid << "->"
<< "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 6)
fout << "i" << vector_of_connectors[q].ddaid <<1<<
"->" << "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 7)
fout << "i" << vector_of_connectors[q].ddaid <<2<<
"->" << "switch"<<switchclusterval<< switer3->first << ";\n";
if (vector_of_connectors[q].pos == 8)
fout << "off" << vector_of_connectors[q].ddaid <<
"->" << "switch"<<switchclusterval<< switer3->first << ";\n";
}
```

```
else{
if (vector_of_connectors[q].pos == 1)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "vpp"
<< vector_of_connectors[q].ddaid << ";\n";
if (vector_of_connectors[q].pos == 2)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "vpn"
<< vector_of_connectors[q].ddaid << ";\n";
if (vector_of_connectors[q].pos == 3)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "vnp"
<< vector_of_connectors[q].ddaid << ";\n";
if (vector_of_connectors[q].pos == 4)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "vnn"
<< vector_of_connectors[q].ddaid << ";\n";
if (vector_of_connectors[q].pos == 5)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "vo"
<< vector_of_connectors[q].ddaid << ";\n";
if (vector_of_connectors[q].pos == 6)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "i"
<< vector_of_connectors[q].ddaid <<1<< ";\n";
if (vector_of_connectors[q].pos == 7)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "i"
<< vector_of_connectors[q].ddaid <<2<< ";\n";
if (vector_of_connectors[q].pos == 8)
fout << "switch"<<switchclusterval<< switer3->first << "->"<< "off"
<< vector_of_connectors[q].ddaid<< ";\n";
}
fout << endl;
}
}
switchclusterval++;
}
}
/**************************************************************************/
SBMAP1=SBCLUSMAP;
t_map_intvect_subcluster::iterator submapiter1 = SBMAP1.begin();
t_vector_int::iterator ddaiter = dda.begin();
ddaiter = dda.begin();
for (; submapiter1 != SBMAP1.end(); submapiter1++){
fout << "subgraph cluster" << "0000" << submapiter1->first << "{\n\n";
fout << "label = " << '"'<<"cluster0000" <<
submapiter1->first<<'"'<<";"<< "\n\n";
t_vector_subcluster &vector_of_subcluster_contents1=submapiter1->second;
for (int i=0; i< vector_of_subcluster_contents1.size(); i++)
{  fout << "dda" << vector_of_subcluster_contents1[i].ddaid1 << ";"
<< "dda" << vector_of_subcluster_contents1[i].ddaid2 << ";";
int ddaidval1 = vector_of_subcluster_contents1[i].ddaid1;
int ddaidval2 = vector_of_subcluster_contents1[i].ddaid2;
fout << "subgraph cluster" <<"000" << ddaidval1 << "{" << "\n\n";
int l = 1;
for (;l<=8; l++) {
if (l==1)
fout << "dda" << ddaidval1 << "->" << "vpp" <<  ddaidval1 << ";" <<"\n";
if (l==2)
fout << "dda" <<  ddaidval1 << "->" << "vpn" << ddaidval1 << ";" <<"\n";
if (l==3)
fout << "dda" <<  ddaidval1 << "->" << "vnp" <<  ddaidval1 << ";" <<"\n";
if (l==4)
fout << "dda" << ddaidval1 << "->" << "vnn" << ddaidval1 << ";" <<"\n";
if (l==5)
fout << "dda" <<  ddaidval1 << "->" << "vo" <<  ddaidval1 << ";" <<"\n";
if (l==6)
fout << "dda" <<  ddaidval1 << "->" << "i" <<  ddaidval1 <<1 << ";" <<"\n";
if (l==7)
fout << "dda" <<  ddaidval1 << "->" << "i" <<  ddaidval1 <<2 <<";" <<"\n";
if (l==8)
fout << "dda" <<  ddaidval1 << "->" << "off" <<  ddaidval1 << ";" <<"\n";
}
fout << "}\n\n";
fout << "subgraph cluster" <<"000" <<  ddaidval2 << "{" << "\n\n";
l = 1;
for (;l<=8; l++) {
if (l==1)
fout << "dda" <<  ddaidval2 << "->" << "vpp" <<  ddaidval2 << ";" <<"\n";
if (l==2)
fout << "dda" << ddaidval2 << "->" << "vpn" <<ddaidval2 << ";" <<"\n";
if (l==3)
fout << "dda" << ddaidval2 << "->" << "vnp" << ddaidval2 << ";" <<"\n";
if (l==4)
fout << "dda" << ddaidval2 << "->" << "vnn" <<ddaidval2 << ";" <<"\n";
if (l==5)
fout << "dda" << ddaidval2 << "->" << "vo" << ddaidval2 << ";" <<"\n";
if (l==6)
fout << "dda" << ddaidval2 << "->" << "i" << ddaidval2 <<1 << ";" <<"\n";
if (l==7)
fout << "dda" << ddaidval2 << "->" << "i" << ddaidval2 <<2 <<";" <<"\n";
if (l==8)
fout << "dda" << ddaidval2 << "->" << "off" << ddaidval2 << ";" <<"\n";
}
fout << "}\n\n";
}
fout << "}\n\n";
}
//*********************************************************************
submapiter1 = SBMAP1.begin();
t_vector_int compint;
t_vector_int switchclustervalu;
t_vector_int::iterator compintiter = compint.begin();
t_vector_int::iterator switchclustervaliter = switchclustervalu.begin();
for (; submapiter1 != SBMAP1.end(); submapiter1++){
t_vector_subcluster &vector_of_subcluster_contents1=submapiter1->second;
for (int i=0; i< vector_of_subcluster_contents1.size(); i++)
{  cout << "dda" << vector_of_subcluster_contents1[i].ddaid1 << ";" ;
cout<< "dda" << vector_of_subcluster_contents1[i].ddaid2 << ";";
cout<< "Switchclus" <<vector_of_subcluster_contents1[i].switchid_bw_id1id2
 << ";";

int ddaidvalu1 = vector_of_subcluster_contents1[i].ddaid1;
int ddaidvalu2 = vector_of_subcluster_contents1[i].ddaid2;
int switchidvalbw = vector_of_subcluster_contents1[i].switchid_bw_id1id2;
cout << ddaidvalu1 << " " << ddaidvalu2 <<"\n";
compint.push_back(ddaidvalu1);
compint.push_back(ddaidvalu2);
switchclustervalu.push_back(switchidvalbw);
}
}
ddaiter = dda.begin();
compintiter = compint.begin();
sort(compint.begin(),compint.end());
```

```
cout << "done" << endl;
for( ; ddaiter != dda.end() ; ddaiter=ddaiter+9){
 //writing out the ddas not included in the cluster into the file
bool found = binary_search(compint.begin(),compint.end(),*ddaiter);
if (found){
cout << *ddaiter << " was found in compint " << endl;
}
else
{
cout << *ddaiter << " was not found in compint " << endl;
fout << "subgraph cluster" <<"000" << *ddaiter << "{" << "\n";
int l =1;
for (;l<=8; l++) {
if (l==1)
fout << "dda" << *ddaiter << "->" << "vpp" << *ddaiter << ";" <<"\n";
if (l==2)
fout << "dda" << *ddaiter << "->" << "vpn" <<*ddaiter << ";" <<"\n";
if (l==3)
fout << "dda" << *ddaiter << "->" << "vnp" << *ddaiter << ";" <<"\n";
if (l==4)
fout << "dda" << *ddaiter << "->" << "vnn" <<*ddaiter << ";" <<"\n";
if (l==5)
fout << "dda" << *ddaiter << "->" << "vo" << *ddaiter << ";" <<"\n";
if (l==6)
fout << "dda" << *ddaiter << "->" << "i" << *ddaiter <<1 << ";" <<"\n";
if (l==7)
fout << "dda" << *ddaiter << "->" << "i" << *ddaiter <<2 <<";" <<"\n";
if (l==8)
fout << "dda" << *ddaiter << "->" << "off" << *ddaiter << ";" <<"\n";
}
fout << "}" << "\n";
}
}
//************************************************************************
SV1 = SV;
vector_map_switches ::iterator  switchiterA = SV1.begin();
switchclusterval = 0;
for (; switchiterA != SV1.end(); switchiterA++){
if (switchclusterval < switchclustervalu.size()){
S1 = *switchiterA;
t_map_intvect_switch::iterator switer = S1.begin();
fout << "subgraph cluster0000"<<switchclusterval<< "{ \n\n";
fout << "subgraph cluster00"<<switchclusterval<<"{\n";
fout << "label = " << '"'<<"switchbox00"<<switchclusterval<<'"'<<";"<<"\n\n";
fout << "{rank=same;";
for(; switer!= S1.end(); switer++)
fout << "Switch"<<switchclusterval<< switer->first << ";";
fout << "}\n"; fout << "}\n";
fout << "}\n";
switer = S1.begin();
for(; switer!= S1.end(); switer++) {
t_vector_switch &vector_of_connectors = switer->second;
for (int i=0; i< vector_of_connectors.size(); i++)
{
if (i!=1){
if (vector_of_connectors[i].pos == 1)
fout << "vpp" << vector_of_connectors[i].ddaid << "->"
<< "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 2)
fout << "vpn" << vector_of_connectors[i].ddaid << "->"
<< "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 3)
fout << "vnp" << vector_of_connectors[i].ddaid << "->"
<< "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 4)
fout << "vnn" << vector_of_connectors[i].ddaid << "->"
<< "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 5)
fout << "vo" << vector_of_connectors[i].ddaid << "->"
<< "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 6)
fout << "i" << vector_of_connectors[i].ddaid <<1<<
"->" << "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 7)
fout << "i" << vector_of_connectors[i].ddaid <<2<<
"->" << "Switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 8)
fout << "off" << vector_of_connectors[i].ddaid <<
"->" << "Switch"<<switchclusterval<< switer->first << ";\n";
}
else{
if (vector_of_connectors[i].pos == 1)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "vpp"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 2)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "vpn"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 3)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "vnp"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 4)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "vnn"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 5)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "vo"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 6)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "i"
<< vector_of_connectors[i].ddaid <<1<< ";\n";
if (vector_of_connectors[i].pos == 7)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "i"
<< vector_of_connectors[i].ddaid <<2<< ";\n";
if (vector_of_connectors[i].pos == 8)
fout << "Switch"<<switchclusterval<< switer->first << "->"<< "off"
<< vector_of_connectors[i].ddaid<< ";\n";
}
fout << endl;
} }
switchclusterval++;}
}
fout << "} \n";
fout.close();
return 0;
}
int function4(t_map_intvect_subcluster SBCLUSMAP){
t_map_intvect_subcluster SBMAP;
SBMAP = SBCLUSMAP;
t_map_intvect_subcluster::iterator submapiter = SBMAP.begin();
```

```
for(; submapiter!= SBMAP.end(); submapiter++) {
cout << "Subcluster" << submapiter->first << "-> contains ";
t_vector_subcluster &vector_of_subcluster_contents = submapiter->second;
for (int i=0; i< vector_of_subcluster_contents.size(); i++)
{  cout << "(" << vector_of_subcluster_contents[i].ddaid1 << ","
<< vector_of_subcluster_contents[i].ddaid2 << ","
<< vector_of_subcluster_contents[i].switchid_bw_id1id2 << ")";}
cout << endl;
}
return 0;
}
/***********************************************************/
 t_map_intvect_switch function5(t_vector_int dda,int first_subcluster,
int second_subcluster,t_map_intvect_subcluster SBCLUSMAP){

t_map_intvect_switch S1; // creating switching map
t_map_intvect_subcluster SBMAP;
int _noelements;
int noconnections;
int srcid,srcport,tarid,tarport,src_ddaid,tar_ddaid;
int cab1, cab2, cab3, cab4;
tconnections _vconnections;
SBMAP = SBCLUSMAP;
t_map_intvect_subcluster::iterator submapiter = SBMAP.begin();

for(; submapiter!= SBMAP.end(); submapiter++) {
cout << submapiter->first << " " << first_subcluster << endl;
if (submapiter->first == first_subcluster){
t_vector_subcluster &vector_of_subcluster_contents = submapiter->second;
for (int i=0; i< vector_of_subcluster_contents.size(); i++)
{  cab1 = vector_of_subcluster_contents[i].ddaid1;
cab2 =   vector_of_subcluster_contents[i].ddaid2;}
}
 cout << submapiter->first << " " << second_subcluster<< endl;
if (submapiter->first == second_subcluster){
t_vector_subcluster &vector_of_subcluster_contents = submapiter->second;
for (int i=0; i< vector_of_subcluster_contents.size(); i++)
{  cab3 = vector_of_subcluster_contents[i].ddaid1;
cab4 =   vector_of_subcluster_contents[i].ddaid2;}
}
}
cout << cab1<< " " <<cab2 << " "<<cab3 << " "<<cab4 <<endl;
ifstream inStream;
inStream.open("rules.txt");
inStream >> _noelements;
_vconnections.resize(_noelements);
for (int i=0; i< _noelements; i++){
inStream >> noconnections;
_vconnections[i].resize(noconnections);
for (int j=0; j < noconnections; j++)
{
inStream >> _vconnections[i][j];
}
}
inStream.close();
for (int i=0; i< _noelements ; i++){
for (int j=0; j < 4;j++){
if (j==0) {src_ddaid = _vconnections[i][j];}
if (j==1) {srcport = _vconnections[i][j];}
if (j==2) {tar_ddaid = _vconnections[i][j];}
if (j==3) {tarport = _vconnections[i][j];}
}
if (src_ddaid == 1)    srcid = (cab1 - 1) * 9;
if (tar_ddaid == 1)    tarid = (cab1 - 1) * 9;

if (src_ddaid == 2)    srcid = (cab2 - 1) * 9;
if (tar_ddaid == 2)    tarid = (cab2 - 1) * 9;

if (src_ddaid == 3)    srcid = (cab3 - 1) * 9;
if (tar_ddaid == 3)    tarid = (cab3 - 1) * 9;

if (src_ddaid == 4)    srcid = (cab4 - 1) * 9;
if (tar_ddaid == 4)    tarid = (cab4 - 1) * 9;

S1[i].push_back(sw_connect(dda[srcid],dda[srcid+srcport],0));
S1[i].push_back(sw_connect(dda[tarid],dda[tarid+tarport],0));

}
return S1;
}

int function6(vector_map_switches SV2,t_vector_int dda){
t_map_intvect_switch S1;
vector_map_switches SV1;
int switchclusterval = 101;
ofstream fout("test5.dot",ios::app);
SV1 = SV2;
vector_map_switches ::iterator  switchiterA = SV1.begin();
for (; switchiterA != SV1.end(); switchiterA++){
S1 = *switchiterA;
t_map_intvect_switch::iterator switer = S1.begin();
fout << "subgraph clusterA"<< "{ \n";
fout << "{rank=same;";
for(; switer!= S1.end(); switer++)
fout << "switch"<<switchclusterval<<switer->first << ";";
fout << "}\n"; fout << "}\n";
switer = S1.begin();
for(; switer!= S1.end(); switer++) {
t_vector_switch &vector_of_connectors = switer->second;
for (int i=0; i< vector_of_connectors.size(); i++)
{
if (i!=1){
if (vector_of_connectors[i].pos == 1)
fout << "vpp" << vector_of_connectors[i].ddaid << "->"
<< "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 2)
fout << "vpn" << vector_of_connectors[i].ddaid << "->"
<< "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 3)
fout << "vnp" << vector_of_connectors[i].ddaid << "->"
<< "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 4)
fout << "vnn" << vector_of_connectors[i].ddaid << "->"
<< "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 5)
fout << "vo" << vector_of_connectors[i].ddaid << "->"
<< "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 6)
fout << "i" << vector_of_connectors[i].ddaid <<1<<
"->" << "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 7)
fout << "i" << vector_of_connectors[i].ddaid <<2<<
"->" << "switch"<<switchclusterval<< switer->first << ";\n";
if (vector_of_connectors[i].pos == 8)
fout << "off" << vector_of_connectors[i].ddaid <<
"->" << "switch"<<switchclusterval<< switer->first << ";\n";
}
else{
if (vector_of_connectors[i].pos == 1)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "vpp"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 2)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "vpn"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 3)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "vnp"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 4)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "vnn"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 5)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "vo"
<< vector_of_connectors[i].ddaid << ";\n";
if (vector_of_connectors[i].pos == 6)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "i"
<< vector_of_connectors[i].ddaid <<1<< ";\n";
if (vector_of_connectors[i].pos == 7)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "i"
<< vector_of_connectors[i].ddaid <<2<< ";\n";
if (vector_of_connectors[i].pos == 8)
fout << "switch"<<switchclusterval<< switer->first << "->"<< "off"
<< vector_of_connectors[i].ddaid<< ";\n";
}
fout << endl;
}
}
fout << " } \n";
}
fout.close();
}

int main(){
int ddanum;
int source_ddaid,target_ddaid;
t_vector_int dda;
t_map_intvect_switch SW, SW2, S6;
vector_map_switches SV, SV2;
t_map_intvect_subcluster SBCLUSMAP;
t_map_intvect_cluster CLUSMAP;
int val,val2,val3; int countnum,countnum2;
int first_subcluster, second_subcluster;
cout << "enter the number of ddas:" ;
cin >> ddanum; cout << "ddanum =" <<ddanum<< endl;
dda = function1(ddanum);
int numofclusters = ddanum/4;

t_map_intvect_switch::iterator switchiterator = SW.begin();
```

```
vector_map_switches ::iterator  switchiter = SV.begin();
t_map_intvect_switch::iterator switchiterator2 = SW2.begin();
vector_map_switches ::iterator  switchiter2 = SV2.begin();
t_map_intvect_subcluster::iterator subclusmapiter = SBCLUSMAP.begin();
t_map_intvect_cluster::iterator clusmapiter = CLUSMAP.begin();

for (int j=1;j<=ddanum/2;j++){
cout << "enter the source dda id:";
cin >> source_ddaid; cout << endl;
cout << " enter the target dda id:";
cin >> target_ddaid; cout << endl;
SW=function2(dda,source_ddaid,target_ddaid);
SV.push_back(SW);
countnum =  SV.size();
SBCLUSMAP[j-1].push_back(subcluster(source_ddaid,target_ddaid,countnum-1));
}
cout << "possible number of clusters= total dda blocks / 4="
<< numofclusters<<endl;
cout << "listing available subclusters"<< endl;
val2 = function4(SBCLUSMAP);
cout << "the size of ths subclusters:=" << SBCLUSMAP.size()<< endl;
for(int h=1;h<=numofclusters;h++){
cout <<"1st subcluster:" ;
cin >> first_subcluster;cout<<endl;
cout << "2nd subcluster:";
cin >> second_subcluster;cout<<endl;
SW2=function5(dda,first_subcluster,second_subcluster,SBCLUSMAP);
SV2.push_back(SW2);
countnum2 = SV2.size();
CLUSMAP[h-1].push_back(cluster(first_subcluster,second_subcluster
,countnum2-1));
}
 /**************************************************/
 cout << "the size of ths clusters:=" << CLUSMAP.size()<< endl;
 cout << "the size of the SV2 :=" << SV2.size()<< endl;
 /***************************************************/
val = function3(SV,dda,SBCLUSMAP,SV2,CLUSMAP);
return 0;
}
```

E.2 Program 2

```
/*********************************************************
Creating combinations without repeatitions or permutations
of an input set. The grouping size is 4. Change the value
of "num" depending upon the input set.

Coded and Compiled by
David Varghese
ESD Group
University of Southampton
*********************************************************/

#include <iostream>
#include <vector>
#include <map>
```

```
#include <fstream>
#include <algorithm>
#include <string>

using namespace std;

typedef vector<char> t_vector_char;
typedef map<char,t_vector_char> t_map_char;
t_vector_char fill_in(t_vector_char charact,char n[]);
void print_out(t_vector_char charact);
t_map_char fillmap(t_vector_char charact, t_map_char map_char);
void printmap(t_map_char map_char);
int char_combination(t_vector_char charact, t_map_char map_char,int num);
t_vector_char repeatfunction(t_map_char mapchar,t_vector_char charact,
t_vector_char comb);

int main()
{
int num = 33;

 char n[]= "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg";

 t_vector_char charact;
 t_map_char map_char;

 charact = fill_in(charact,n);

 map_char = fillmap(charact,map_char);
 printmap(map_char);

 for (int f=1; num!=0; f++){
 char_combination(charact,map_char,num);
 map_char.erase(map_char.begin());
 cout << " map after an erase" << endl;
 printmap(map_char);
 cout << " charac after an erase" << endl;
 charact.erase(charact.begin());
 print_out(charact);
 cout << "**************************************************"<<endl;
 cout << f << endl;
 num--;
 }
}

int char_combination(t_vector_char charact, t_map_char map_char,int num){
 t_vector_char comb;
 t_vector_char r;
 string test;
 t_vector_char::iterator comb_iter = comb.begin();
 t_vector_char::iterator comb_iter2 = comb.begin();
 t_vector_char::iterator comb_iter3 = comb.begin();
 t_vector_char::iterator comb_iter4 = comb.begin();
 t_vector_char::iterator r_iter = r.begin();
 t_vector_char::iterator charact_iter = charact.begin();
 t_vector_char::iterator charact_iter2 = charact.begin();
 t_vector_char::iterator charact_iter3 = charact.begin();

 t_map_char::iterator map_iter = map_char.begin();
 int val = 0;
 int val2 =0;
 int val3=0;

 int blocks1,blocks2,blocks3,blocks4,blocks5,blocks6,blocks7;
 int blocks8,blocks9,blocks10,blocks11,blocks12,blocks13,
blocks14,blocks15,blocks16,blocks17,blocks18;
int blocks19,blocks20,blocks21,blocks22,blocks23,
blocks24,blocks25,blocks26,blocks27,blocks28;
int blocks29,blocks30,blocks31,blocks32,blocks33;
int totalnum;

int  noi=0; int nop=0; int noex=0; int nogn=0;
int  no_of_inps=0;
int no_of_ops =0;
int no_of_exts =0;
int no_of_gnd =0;

for(;map_iter!=map_char.end();map_iter++){
for (;charact_iter!=charact.end();charact_iter++){
for (;charact_iter2!=charact.end() and val2 < charact.size();charact_iter2++){
for (;charact_iter3!=charact.end() and val3 < charact.size();charact_iter3++){
comb.push_back(map_iter->first);
comb.push_back(*charact_iter);
comb.push_back(*charact_iter2);
comb.push_back(*charact_iter3);
}
val3++;
charact_iter3 = charact.begin()+ val3;
}
val2++;
charact_iter2 = charact.begin()+ val2;
}
val++;
charact_iter = charact.begin()+val;
}
cout << "inside again" << endl;
/****************************************************************/

for (int i=1; i<=num;i++){
cout << "inside again 2 " << endl;
charact.erase(charact.begin());
comb=repeatfunction(map_char,charact,comb);
}

/****************************************************************/

ofstream fout("results_now_greater4.txt",ios::app);
comb_iter = comb.begin();
int sum=0;
totalnum=0;

for (; comb_iter != comb.end();comb_iter=comb_iter+4){
r.push_back(*comb_iter);
comb_iter2 = comb_iter+1;
r.push_back(*comb_iter2);
comb_iter3 = comb_iter+2;
r.push_back(*comb_iter3);
comb_iter4 = comb_iter+3;
r.push_back(*comb_iter4);
```

```
blocks1 = 0; blocks2=0; blocks3=0; blocks4=0;
blocks5 = 0; blocks6=0; blocks7=0; blocks8=0;
blocks9 = 0; blocks10=0; blocks11=0; blocks12=0;
blocks13 = 0; blocks14=0; blocks15=0; blocks16=0;
blocks17 = 0; blocks18=0; blocks19=0;blocks20=0;
blocks21=0;blocks22=0;blocks23=0;blocks24=0;blocks25=0;
blocks26=0;blocks27=0;blocks28=0;
blocks29=0;blocks30=0;blocks31=0;blocks32=0;blocks33=0;
sum=0; noi=0; nop=0; noex=0; nogn=0;
no_of_inps =0;
no_of_ops =0;
no_of_exts =0;
no_of_gnd =0;

for (int i=0; i<=3;i++){
if (r[i] == 'A'){ blocks1 = 1; noi = 1; nop=1; noex = 0; nogn=2;}
if (r[i] == 'B'){ blocks2 = 1; noi = 1; nop=1; noex = 0; nogn=2;}
if (r[i] == 'C'){ blocks3 = 1;noi = 3; nop=1; noex = 0; nogn=0;}
if (r[i] == 'D'){ blocks4 = 1;noi = 2; nop=1; noex = 0; nogn=1;}
if (r[i] == 'E'){ blocks5 = 1;noi = 2; nop=1; noex = 0; nogn=2;}
if (r[i] == 'F'){ blocks6 = 1;noi = 2; nop=1; noex = 2; nogn=1;}
if (r[i] == 'G'){ blocks7 = 1;noi = 3; nop=1; noex = 0; nogn=1;}
if (r[i] == 'H'){ blocks8 = 1;noi = 1; nop=1; noex = 1; nogn=0;}
if (r[i] == 'I'){ blocks9 = 1;noi = 2; nop=1; noex = 2; nogn=1;}
if (r[i] == 'J'){ blocks10 = 1;noi = 2; nop=1; noex = 3; nogn=0;}
if (r[i] == 'K'){ blocks11 = 1;noi = 1; nop=1; noex = 0; nogn=1;}
if (r[i] == 'L'){ blocks12 = 1;noi = 1; nop=1; noex = 1; nogn=0;}
if (r[i] == 'M'){ blocks13 = 1;noi = 1; nop=1; noex = 2; nogn=0;}
if (r[i] == 'N'){ blocks14 = 1;noi = 1; nop=1; noex = 0; nogn=1;}
if (r[i] == 'O'){ blocks15 = 1;noi = 1; nop=1; noex = 2; nogn=0;}
if (r[i] == 'P'){ blocks16 = 1;noi = 1; nop=1; noex = 2; nogn=0;}
if (r[i] == 'Q'){ blocks17 = 2;noi = 1; nop=3; noex = 4; nogn=1;}
if (r[i] == 'R'){ blocks18 = 4;noi = 1; nop=1; noex = 6; nogn=1;}
if (r[i] == 'S'){ blocks19 = 8;noi = 1; nop=1; noex = 10; nogn=0;}
if (r[i] == 'T'){ blocks20 = 1;noi = 2; nop=2; noex = 0; nogn=0;}
if (r[i] == 'U'){ blocks21 = 3;noi = 2; nop=1; noex = 3; nogn=1;}
if (r[i] == 'V'){ blocks22 = 2;noi = 4; nop=1; noex = 2; nogn=1;}
if (r[i] == 'W'){ blocks23 = 1;noi = 1; nop=1; noex =1; nogn=2;}
if (r[i] == 'X'){ blocks24 = 1;noi = 2; nop=1; noex = 2; nogn=0;}
if (r[i] == 'Y'){ blocks25 = 4;noi = 1; nop=1; noex = 9; nogn=0;}
if (r[i] == 'Z'){ blocks26 = 1;noi = 2; nop=1; noex = 0; nogn=0;}
if (r[i] == 'a'){ blocks27 = 3;noi = 1; nop=3; noex = 4; nogn=2;}
if (r[i] == 'b'){ blocks28 = 1;noi = 2; nop=1; noex = 1; nogn=0;}
if (r[i] == 'c'){ blocks29 = 1;noi = 2; nop=1; noex = 2; nogn=2;}
if (r[i] == 'd'){ blocks30 = 3;noi = 3; nop=1; noex = 4; nogn=4;}
if (r[i] == 'e'){ blocks31 = 2;noi = 1; nop=3; noex = 5; nogn=1;}
if (r[i] == 'f'){ blocks32 = 2;noi = 1; nop=1; noex = 2; nogn=1;}
if (r[i] == 'g'){ blocks33 = 2;noi = 1; nop=2; noex = 2; nogn=1;}

sum =sum+blocks1+blocks2+blocks3+blocks4+blocks5+blocks6+blocks7+
blocks8+blocks9+blocks10+blocks11+blocks12+blocks13+blocks14+
blocks15+blocks16+blocks17+blocks18+blocks19+blocks20+blocks21+
blocks22+blocks23+blocks24+blocks25+blocks26+blocks27+blocks28+
blocks29+blocks30+blocks31+blocks32+blocks33;

no_of_inps = no_of_inps+noi;
no_of_ops = no_of_ops+nop;
no_of_exts = no_of_exts + noex;
```

```
no_of_gnd = no_of_gnd + nogn;

noi=0;nop=0;noex=0;nogn=0;

blocks1 = 0; blocks2=0; blocks3=0; blocks4=0;
blocks5 = 0; blocks6=0; blocks7=0;
blocks8=0; blocks9 = 0; blocks10=0; blocks11=0; blocks12=0;
blocks13 = 0; blocks14=0; blocks15=0; blocks16=0;
blocks17 = 0; blocks18=0;blocks19=0;blocks20=0;blocks21=0;
blocks22=0;blocks23=0;blocks24=0;blocks25=0;blocks26=0;blocks27=0;blocks28=0;
blocks29=0;blocks30=0;blocks31=0;blocks32=0;blocks33=0;
}
totalnum = sum;
r_iter=r.begin();
if (totalnum >4){
for (;r_iter!=r.end();r_iter++){
fout<<*r_iter;}
fout << "="<<totalnum<<"="<<no_of_inps<<"="<<no_of_ops<<"="<<
no_of_exts<<"="<<no_of_gnd<<endl;
}

totalnum = 0;
sum=0;
no_of_inps =0;
no_of_ops =0;
no_of_exts =0;
no_of_gnd =0;

r.clear();
r_iter= r.begin();
}

fout.close();
cout << comb.size() << endl;
comb.clear();
return 0;


t_vector_char repeatfunction(t_map_char map_char,t_vector_char
charact,t_vector_char comb){

t_vector_char::iterator charact_iter = charact.begin();
t_vector_char::iterator charact_iter2 = charact.begin();
t_vector_char::iterator charact_iter3 = charact.begin();
t_map_char::iterator map_iter = map_char.begin();


charact_iter = charact.begin();
charact_iter2 = charact.begin();
charact_iter3 = charact.begin();
map_iter = map_char.begin();

int val =0;
int val2=0;
int val3=0;

for(;map_iter!=map_char.end();map_iter++){
for (;charact_iter!=charact.end() and val < charact.size();charact_iter++){
for (;charact_iter2!=charact.end() and val2 < charact.size();charact_iter2++){
```

```
for (;charact_iter3!=charact.end() and val3 < charact.size();charact_iter3++){
comb.push_back(map_iter->first);
comb.push_back(*charact_iter);
comb.push_back(*charact_iter2);
comb.push_back(*charact_iter3);
}
val3++;
charact_iter3 = charact.begin()+ val3;
}
val2++;
charact_iter2 = charact.begin()+ val2;
}
val++;
charact_iter = charact.begin()+val;
}
return comb;
}

void printmap(t_map_char map_char){
t_map_char::iterator map_iter = map_char.begin();

map_iter = map_char.begin();
cout << "size of the map="<< map_char.size() << endl;

for(;map_iter != map_char.end(); map_iter++){

cout << "map" << map_iter->first << " ->";
t_vector_char &vector_of_char = map_iter->second;

for(int i=0; i< vector_of_char.size();i++){
cout << vector_of_char[i];

}
cout << endl;
}
}

t_map_char fillmap(t_vector_char charact,t_map_char map_char){
t_vector_char::iterator charact_iter = charact.begin();
t_vector_char::iterator charact_iter2 = charact.begin();
t_map_char::iterator map_iter = map_char.begin();
int val =0;

for (int j=0; j< charact.size();j++){
for (;charact_iter != charact.end();charact_iter++){
for(;charact_iter2 !=charact.end() and val<charact.size();charact_iter2++){
map_char[*charact_iter].push_back(*charact_iter2);
}
val++;
charact_iter2 = charact.begin()+ val;
}
charact_iter = charact.begin();
}
return map_char;
}

t_vector_char fill_in(t_vector_char charact,char n[])
{
t_vector_char::iterator charact_iter = charact.begin();
```

```
for (int i=0; n[i] !='\0'; i++)
{
charact.push_back(n[i]);
}
return charact;
}
void print_out(t_vector_char charact)
{  t_vector_char::iterator charact_iter = charact.begin();
cout << endl;
for(;charact_iter != charact.end();charact_iter++)
{
cout << *charact_iter;
}
cout << endl;
}
```