Architecture Level Power-Performance Tradeoffs for Pipelined Designs

Haider Ali and Bashir M. Al-Hashimi
ESD, School of ECS
University of Southampton
Southampton, UK
{ha02r, bmah}@ecs.soton.ac.uk

***Abstract*—This paper presents a method to investigate power-performance tradeoffs in digital pipelined designs. The method is applied at the architectural level of the design. It will be shown that addressing the tradeoffs at this level will result in significant savings in power consumption without impacting the performance. The reduction in power is obtained through reducing the number of registers used in implementing the pipeline stages. The method has been validated by synthesizing a floating-point unit with different pipeline stages and power consumption of the designs were obtained using industry standard tools. It is shown that it is possible to obtain up to 18% reduction in power without affecting the clock period and with less area.**

I. INTRODUCTION

Power in recent years has become one of the most important parameters facing design engineers. Designers on the other hand are being asked to integrate more circuitry that operates faster to meet the functional specification which in turn drives up the overall power consumption. Excessive power consumption can lead to thermal issues, reliability and complex power supply design [1]. Power saving can be tackled at different levels of the design cycle and research has shown the earlier in the cycle the more gain in terms of power reduction [2].

Pipelining a design involves the addition of registers to create stages with the aim of improving throughput when the pipeline is fully utilized. Crucially the location of the pipeline stage will dictate the number of registers required to implement the stage. Since registers are power hungry elements, reduction in the total number of registers will yield reduction in power consumption. In recent research [3]-[5], pipeline power-performance tradeoffs have either been tackled at the gate level or with no systematic approach to where stages should be inserted. For example, [3] presented a gate level analytical approach to determine the number of gates in each pipeline stages that optimizes power and performance. Limited benefit was seen in terms of power reduction because of the low level of abstraction. While in [4] and [5] attempts were made to solve the problem at the architecture level, there was no systematic approach to where stages should be inserted. Since analyzing a design at a higher level of abstraction produce better results in terms of power reduction, in this paper we present a new method which operates at the architecture level. The method explores efficiently the power-performance tradeoffs and takes into account the required clock period while evaluating the merits of each stage insertion based on the number of registers needed to implement the stage. The final result is sets of solutions with different power-performance characteristics.

II. MOTIVATIONAL EXAMPLE

In this section we show through an example of a data-dominated architecture (floating-point adder) how different pipeline stage insertion could lead to reduced power consumption through reduction in stage registers without impacting performance. The floating-point adder architecture in [6] is the benchmark for this example and is shown in Fig. 1. Each box in Fig. 1 is a functional element (FE) that performs a task which contributes to the overall floating-point adder. These elements have been the subject of extensive research and for this example an appropriate implementation has been selected. To highlight the impact of different stage insertion on power, [7] proposed a 5-stages pipeline of this architecture (stages are shown in Fig. 1) and we use it as a baseline to compare against our ad hoc 3-stages pipeline shown in Fig. 2. We implemented the ad hoc stages by trying to find stage insertion points that results in fewer number of registers compared to the 5-stages. The designs were implemented in VHDL and synthesized with a target system frequency of 200MHz. A state of the art 1.3V, 90nm technology library was used during the synthesis and the netlist was analyzed for dynamic power consumption in Synopsys PrimePower tool using toggle data from gate-level simulation. Table I shows the final result which indicates both 3-stages and 5-stages architectures satisfy the target frequency of 200MHz. However, the ad hoc stages consumes 17.6% less power and 25% less area compared to the

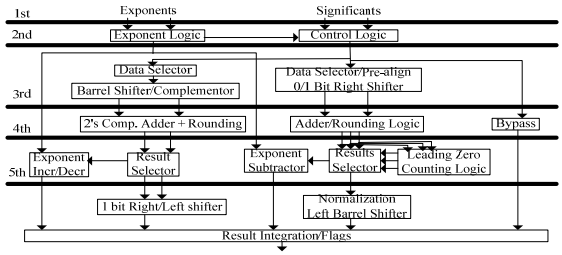


Figure 1. Floating-point adder [6] with 5-stages implementation [7]

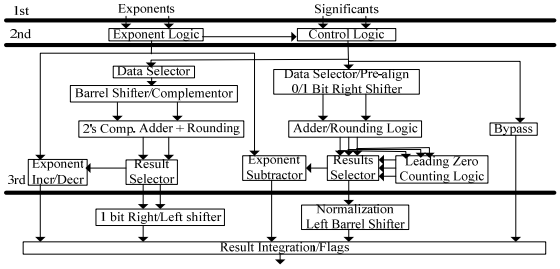


Figure 2. 3-stages ad hoc

5-stages pipeline. As expected, the 5-stages pipeline adder has higher power consumption compared to the 3-stages due to the extra number of registers needed to implement the stages. As will be shown in the experimental results different system clock period results in different number of stages and power consumption.

TABLE I. RESULTS OF 3-STAGES AND 5-STAGES PIPELINE

	3-stages ad hoc	**5-stages [7]**
Power (mW)	3.7	4.49
Area (mm2)	2.1e-3	2.28e-3
Freq. (MHz)	200	200
Latency (ns)	15	25

III. PROPOSED METHOD

Three main steps compose the method: **1)** Start with non-pipelined design architecture, e.g. floating-point adder and generate a detail data flow graph (DFG), see Fig. 3; **2)** synthesize each element of the architecture to determine its minimum delay and estimate the element power consumption using Synopsys PrimePower tool. Finally, generate naive pipeline stages (naive stages in this context represents the simplest way of inserting stages achieved by registering FE outputs without considering the number of outputs), see Fig. 3; **3)** the naive stages are simplified based on two observations; a) across a stage one element could dominate in terms of delay masking all other elements and becomes the critical path of that stage; b) when an interconnect crosses a stage boundary that interconnect needs to be registered along with outputs of all elements at that stage (critical path elements or otherwise). The information in the simplified naive stages (see Fig. 4) are extracted and saved in a data array. The array is examined using a pipeline stage algorithm (PSA) to determine if stages could be removed without impacting performance.

The PSA has been developed to take in to account system clock period C, stage outputs O and stage critical path delay T and can be summarized in Eqs. (1), (2) and (3) were i is stage number. P: stage does not exist after outputs O, W: the ratio of the number of next stage outputs over previous stage outputs and I: a stage after outputs O can be inserted are all single bit flags. In order for the equations to work correctly the value for I and W is restricted to 1 or 0 with the rules shown in Eq. (4). The variable Φ is used as way of delaying the insertion of a stage and the value used depends on the distribution of outputs in the architecture as will be shown in the experimental results.

$$I_i = \frac{\left\lfloor \sum_{k=1}^{k=i-1} T_k \left\lceil \prod_{m=k}^{m=i-1} P_m \right\rceil \right\rfloor + T_i + T_{i+1}}{C} \quad (1)$$

$$P_i = | I_i - 1 | * | W_i - 1 | \quad (2)$$

$$W_i = \frac{O_{i+1}}{O_i} \quad (3)$$

$$\begin{aligned} &\text{If } I_i > 1, \text{ set } I_i \text{ to 1 otherwise 0} \\ &\text{If } \{W_i > \Phi\} \text{ is true, set } W_i \text{ to 1 otherwise 0, where } \Phi \geq 0 \end{aligned} \quad (4)$$

Each naive stage will have a different value for I, P and W and it is the value of P that determines whether a stage could be removed or not. For example, when P=0 it means a stage must exist otherwise timing would be violated. When P=1, it implies a stage could be removed without affecting timing. Power consumption of the final pipelined design is estimated with Synopsys PrimePower tool.

IV. EXPERIMENTAL RESULTS

In this section we demonstrate the three steps of the proposed method outlined in section III on the floating-point adder shown in Fig. 1.Step 1 is satisfied by taking Fig. 1 and generating a detail DFG shown in Fig 3. Note, also shown in Fig. 2 is the naive stages which is required by step 2 and they have been implemented by simply adding stages after FEs and having where possible more than one FE in a stage. Each element in Fig. 3 was coded in VHDL, synthesized with state of the art 1.3V, 90nm technology library and a layout floorplan was generated using Magma BlastFusion tool. Table II summaries the minimum delay for each element. The power consumption of each element was estimated using Synopsys PrimePower tool with a verilog netlist from layout data and switching activity from real netlist simulations. Step 3 is achieved by simplifying Fig. 3 to that shown in Fig. 4 based on the observations outlined in section III. From Fig. 4, Table III is extracted and forms the input data array to PSA (stage0 is assumed to always exist). The PSA was executed with different clock period constraints with Φ set to 1 and power was estimated at each clock period. The power-performance result is shown in Fig. 5 for the proposed approach and the naive stages. There are three interesting observations. First, PSA is able to perform as well as the naive approach and a solution was generated for each clock period. Second, when the clock period is > 1700ps, our algorithm always gives better results in terms of power consumption compared to the naive stages. For

example, when clock period is 2500ps, PSA stages results in dynamic power consumption of 21mW compared with 26mW for naive stages which represents an 18% power reduction. At higher clock frequencies with clock periods of < 1700ps both approaches produce the same result. This can be explained by the fact that when the clock period is very short there is less opportunity to remove stages without compromising performance. Third, given a power consumption budget, PSA generates results that have higher performance than naive stages (assuming clock period is > 1700ps). For example, with a power budget of 30mW, PSA solution can operate at 2100ps compared with 2300ps for naive stages. If we consider the breakdown of power consumed by combinational logic versus that of registers we get the result shown in Fig. 6. This graph indicates that register power is the dominant part at high frequencies but when clock frequency is reduced a crossover point occurs where combinational logic starts to consume more power than registers. Note, the crossover point happens at a fairly high

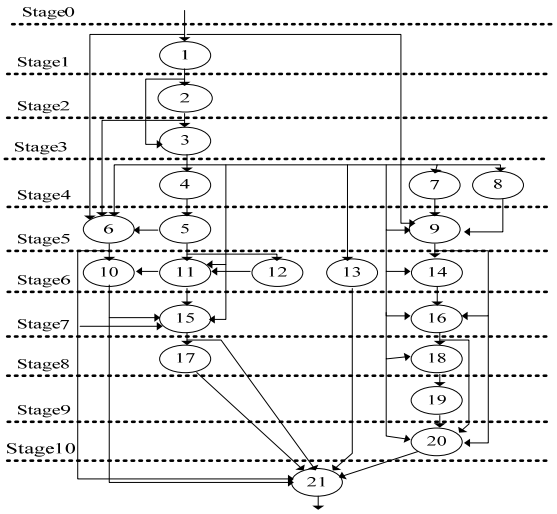


Figure 3. Detail DFG of [6] with naive stages

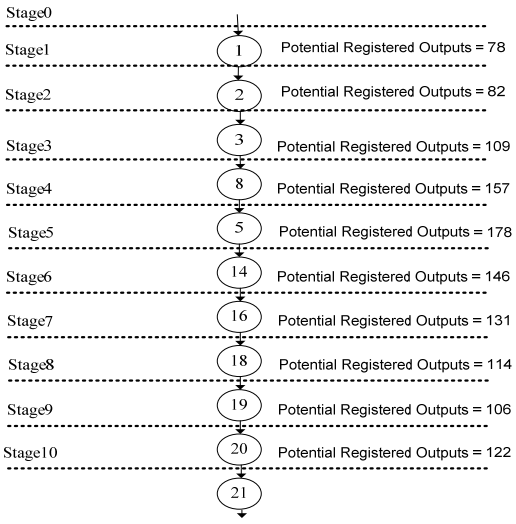


Figure 4. Simplified naive stages

TABLE II. ADDER FUNCTIONAL ELEMENTS DELAYS

No.	Element Name	Delay (ps)
1	Denormal Check	900
2	Exponent Subtractor	1300
3	Control Logic	1200
4	Data Select	600
5	Significant Adder	1400
6	Final Sign	700
7	Barrel Right	1000
8	StickyBit	1500
9	Pre-Significant Adder	600
10	Exponent Update	1300
11	Result Select	900
12	LZCounter	1100
13	Bypass logic	600
14	Significant Adder	1400
15	Pre-Barrel Left	700
16	M0Minus1 Generator	1100
17	Barrel Left	1000
18	Exponent Subtractor	1100
19	Underflow/Overflow	700
20	LZB Final Shift	1000
21	Result Integrator	900

TABLE III. SIMPLIFIED NAIVE STAGES DATA ARRAY

Stage No.	Delay (ps)	Outputs
Stage1	900	78
Stage2	1300	82
Stage3	1200	109
Stage4	1500	157
Stage5	1400	178
Stage6	1400	146
Stage7	1100	131
Stage8	1100	114
Stage9	700	106
Stage10	1000	112

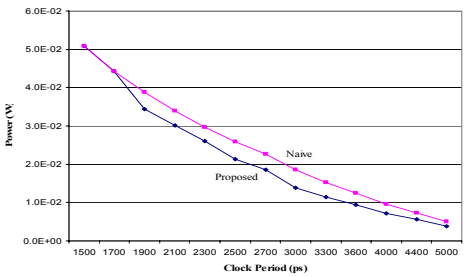


Figure 5. Power-performance trend for PSA

frequency corresponding to clock period range of 2700-3000ps. The significance of this means that if the user selects a system clock period > 3000ps, adding or removing small number of registers will not impact power consumption significantly. Therefore, the PSA result provides the user with the flexibility to alter the number of registers (albeit a small addition or subtraction) without compromising performance or power. Such a crossover point does exist in the naive stages but at much increased clock period > 7000ps. In terms of actual number of stages and register count verses clock period, Fig. 7 a) and b) respectively show how for PSA results these two parameters decrease with increasing clock period. This result is as expected since increasing clock period allows more stages to be removed and hence less register count. Although for periods > 3000ps the actual register count for PSA is almost flat we still see power reduction due to decreased operating frequency. It also can be observed that the value of Φ can affect the final solution in terms of power. For example, at 5000ps the

solution with Φ=1.4 consumes less dynamic power than when Φ=1. It may seem that we have to search through a large number of Φ values to find the least power consuming solution, however, in practice this is not the case. The maximum value of Φ is found from the result of dividing the largest number of stage outputs by the smallest. The range of Φ values is simply from 1 to maximum Φ increasing at a user defined interval rate. This significantly reduces the exploration space and makes the algorithm have low time complexity.

Further experiment was done by comparing PSA to a general-purpose exhaustive search algorithm [8]. The exhaustive algorithm was executed for a range of clock periods and *best_solution* is noted for each clock period when least power consumption is achieved and timing is met. Fig. 8 shows power-performance comparison between PSA and exhaustive search. From this it is clear that for the floating-point adder benchmark the PSA results are closely correlated to that of exhaustive search. In terms of register count and number of stages, Fig. 9 a) and b) respectively show practically identical results between the two algorithms. The downside of the exhaustive search is the polynomial time complexity $O(n^2)$ compared with the linear time complexity for PSA $O(n)$ where n is number of stages (for example, it took PSA 9 seconds to generate the results in Fig. 9 compared with 13s for exhaustive search). This is significant when considering complex architectures with large number of stages.

V. CONCLUSION

In this paper, we have demonstrated that it is important to consider clock period, functional elements outputs and delays when inserting pipeline stages. This is best achieved at the architecture level where element boundaries are clearly defined. We also detailed a new method which facilitates the investigation of the best possible stage location that yields the least number of registers. As part of the method an algorithm was developed which has the benefit of low time complexity and the ability to efficiently search through design space to explore power-performance tradeoffs. Our results indicate that the algorithm can generate solutions that are more power efficient compared with the naive stages and comparable in power to that generated with an exhaustive search algorithm.

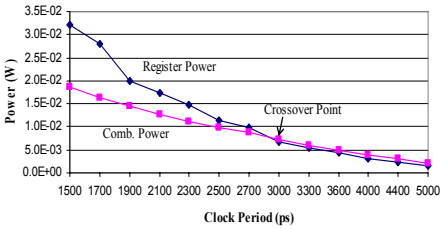
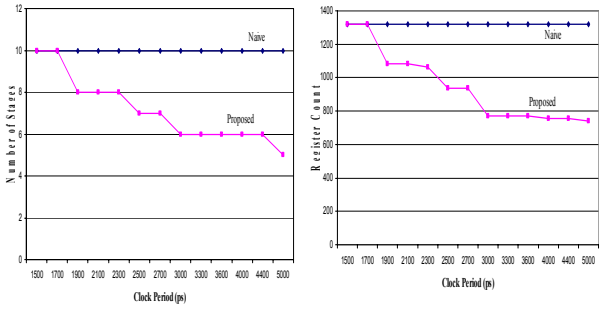


Figure 6. Combinational Logic and register power trend



a) **b)**

Figure 7. Number of stages and register count for naive stages and PSA with Φ=1

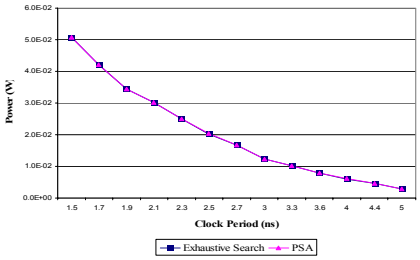
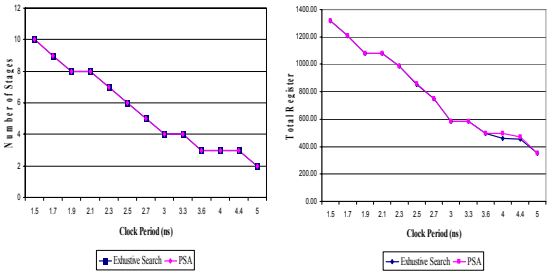


Figure 8. Power-performance comparison between exhaustive search and PSA



a) **b)**

Figure 9. Number of stages and register count for exhaustive search and PSA

REFERENCES

[1] B. M. Al-Hashimi, "System-on-chip: next generation electronics", The Institution of Electrical Engineers, IEE Circuits, Devices and Systems Series 18, 2006, isbn 0-86341-552-0.

[2] D. I. Lazorenko, A. A. Chemeris, "Low-power issues for soc", Proc. of the 2006 IEEE 10th Int. Symp. on Consumer Electronics, 28-01 June/July 2006, pp. 1-3.

[3] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, P. Bose, P. N. Strenski, P. G. Emma "Integrated analysis of power and performance for pipelined microprocessors", IEEE Trans. on Computers, 53(8), Aug. 2004, pp. 1004-1016.

[4] A. Garcia, W. Burleson, J. L. Danger, "Low power digital design in fpga's", Proc. of the 2000 IEEE Int. Symp. on Circuits and Systems, vol. 5, 28-31 May 2002, pp. 561-564.

[5] A. Hartstein and T. R. Puzak, "Optimum power/performance pipeline depth", Proc. of the 36th Annual IEEE/ACM International Symposium on Microarchiecture, 2003, pp. 117-125.

[6] R. V. K. Pillai, D. Al-Khalili, A. J. Al-Khalili, S. Y.A. Shah, "A low power approach to floating point adder design for dsp applications", Journal of VLSI Signal Processing, 27(3), March 2001, pp. 195-213.

[7] S. Y. A. Shah, "On synthesis and optimisation of floating point unit", MSc Thesis, Concordia University, Montreal, Quebec, Canada, Oct. 2000.

[8] S. H. Gerez, "Algorithms for vlsi design automation", 1999, Wiley.