Dual Transitions Petri Net based Modelling Technique for Embedded Systems Specification

Mauricio Varea and Bashir Al-Hashimi

Department of Electronics and Computer Science
University of Southampton, SO17 1BJ, Southampton, UK

{mv99r,bmah}@ecs.soton.ac.uk

Abstract

This paper presents a new modelling technique capable of modelling both control and data information using a single unified approach. This is achieved by modifying the classical Petri Net structure, allowing it to have two types of transitions and arcs. As a consequence, loops and conditional operations within complex specifications are easily identified. The system dynamic behaviour is modelled using a new marking scheme of the net consisting of a new element called value *for data representation in addition to classical tokens used for control purpose. Structural definitions, behavioural rules and graphical representation of the new modelling technique are given. One potential application of the proposed modelling technique is the internal representation of embedded systems specification. Two examples are included illustrating the applicability and efficiency of the proposed modelling technique.*

1. Introduction

Classical Petri Net (PN) have been applied to the modelling of various systems [8]. Due to their inability to deal with data flow, a number of extensions to the classical PN have been proposed, including[1] Predicate / Transitions Nets (PrT-Nets) [3] and Coloured Petri Nets (CPN) [4–6]. Extended Timed Petri Nets (ETPN) were introduced in CAMAD high-level synthesis system [9]. It consists of two separate but related parts: The control part, captured as a Timed Petri Net (TPN), and the data path, represented as a directed graph where nodes are used to capture data manipulation and storage.

Some progress has been achieved in applying PNs oriented techniques to the modelling process in a hardware/software embedded system co-design framework. In PURE methodology [12], ETPN has been extended to allow modelling of hardware/software interface and software structure. By making data dependencies explicit, PURE is capable of easily identifying communication needs. Recently another PN based representation called PRES has been introduced [1] as a modelling technique for verification of heterogeneous systems. It can be used to model a system at different levels of of abstraction using hierarchy. The model includes an explicit notation of time, and tokens are redefined in order to let its dynamic behaviour to deal with both data and control.

ETPN model [9] is a good internal representation for hardware synthesis, since it exploits one of the features of PN, which is concurrency identification. However, *data* and *control* flow are not very well linked in this model. To overcome the lack of union between these two domains, PRES [1] and our proposed modelling technique employ a single graph to model the embedded system specification. Embedded Systems specification usually consists of both, control functions and data operations. Its functionality can be thought as a transformation of data controlled by logical functions [11]. Modelling the internal design representation (IDR) of such specification plays an important role in the quality of the final solution for embedded system co-design framework. We believe that a good embedded hardware/software co-design framework is not only obtained by speeding up the co-design algorithms affecting macroscopic system implementation, i.e. partitioning and scheduling [7], but also by efficient IDR of embedded system where data and control flow should be tightly linked. The aim of this paper is to present a new efficient modelling technique suitable for the internal representation of embedded systems specification.

2. Proposed Modelling Technique

In this section we present the new modelling technique and describe its structural definition, behavioural rules and graphical representation. Although notation is based on PN, some extensions have been made in order to allow the representation of two mutually exclusive domains: *control* and *data*. This has resulted in modifying the classical four-tuple [10], extending the definition of a PN structure (i.e. $PN = \langle P, T, F, W \rangle$) into a seven-tuple (Definition 1, Section 2.1).

2.1. Structural model

The structure of the proposed modelling technique is composed of passive elements, active elements, arcs and some mapping functions. Passive elements - or places (P), are associated with memory storage in the embedded system (e.g. registers, memory cells, latches, variables, etc.). Active elements - or transitions (T, Q), refer to any logical or arithmetic operation (e.g. data assignments, conditional jumps, control signals, etc.). There are two types of transition in the new modelling technique: control transitions (T) and data transitions (Q). Also two types of arcs support this structure, control flow arcs (F_C) and data flow arcs (F_D).

It should be noted that a classical PN can be constructed by means of using a subset of the proposed technique[2]. Con-

[1] Both PrT-Nets and CPN are High Level Petri Nets (HLPN)

[2] The proposed structure is equivalent to the classical PN when it does not contain any data information. This is:

$$N \equiv PN = \langle P, T, F_C, W_C \rangle \iff \exists(Q = \varnothing) \wedge \exists(F_D = \varnothing) \wedge \exists(G = \varnothing)$$

trol transitions concept remains from the classical PN transitions [10], whereas data transitions are incorporated into the proposed technique to provide information from the data domain. This differs from other proposed modelling approaches based on PNs. Neither ETPN nor PRES modifies the structural information of the four-tuple in order to deal with the data domain. In ETPN, the data flow modelling is carried out by including another graph. In PRES, token's definition has been modified in order to allow data representation. HLPNs have one set of passive elements and one set of active elements, while the proposed modelling technique has two sets of active elements (T and Q) and allows each set to focus on a specific domain, i.e. control and data.

Definition 1 *The structure is a seven-tuple* $N = \langle P, T, Q, F_C, F_D, W, G \rangle$

Where: $P = \{p_1, p_2, \cdots, p_n\}$ *is a finite set of places.*
$T = \{t_1, t_2, \cdots, t_m\}$ *is a finite set of control transitions.*
$Q = \{q_1, q_2, \cdots, q_h\}$ *is a finite set of data transitions.*
$F_C \subseteq (P \times T) \cup (T \times P) \cup (P \times Q)$ *is a set of arcs describing the control flow relation.*
$F_D \subseteq (P \times Q) \cup (Q \times P) \cup (P \times T)$ *is a set of arcs describing the data flow relation.*
$W : \{F_C \cup F_D\} \to \mathbb{Z}$ *is a weight function.*
$G : 2^P \to \{0,1\}$ *is the guard function.*

P, T and Q are disjointed, i.e. $P \cap T = \varnothing$, $P \cap Q = \varnothing$ and $T \cap Q = \varnothing$. In order to have a valid model, the existence of at least one passive and one active element is required, thus both P and $T \cup Q$ are not empty sets. Note that arcs describing the control/data flow connect only passive with active elements or vice versa. The connection of passive to passive elements or active to active elements are not allowed in the proposed modelling technique. The order of each set P, T and Q is $n \in \mathbb{N}$, $m \in \mathbb{N}$ and $l \in \mathbb{N}$ respectively; where $n > 0$, $m \geqslant 0$ and $h \geqslant 0$. Also $m + h > 0$ applies.

Let x be any passive or active element (i.e. p_i, t_j or q_l). Then, pre- and post-sets definitions are composed of: ${}^\bullet x$ and $x^\bullet$, which are kept from the classical PN [8] definitions for modelling the control domain; and ${}^\circ x$ and x°, which are introduced in the proposed modelling technique to support the data domain. A summary of symbols is shown in Table 1.

Domain	Pre-set	Post-set
Control	${}^\bullet x$	$x^\bullet$
Data	${}^\circ x$	x°

Table 1. Symbols for pre- and post-sets

Definition 2 *The weight function W is composed (i.e. union) of:*

$W_C \subseteq (P \times T) \cup (T \times P) \to \mathbb{Z}^+$ *is a mapping from a subset of* F_C *into a finite set of positive integers.*

$W_D \subseteq (P \times Q) \cup (Q \times P) \to \{\mathbb{Z}^+ \cup \mathbb{Z}^-\}$ *is a mapping from a subset of* F_D *into a finite set of integers (excluding 0).*

The two subsets introduced in Definition 2 are disjointed, i.e. $W_C \cap W_D = \varnothing$.

To link control and data information in the proposed modelling technique a new mapping function called guard function (G) has been introduced[3], in addition to the weight function (W) which is normally used to combine several arcs. The guard function is used to represent situations where the control flow is influenced by some data. In order to give a formal definition of G, let L be a finite set of predicates, then:

Definition 3 *The guard function G is a mapping from sets of places to the boolean space, i.e.* $G : 2^P \to \{0,1\}$*, such that each set of places in the guard function domain is the pre-set* ${}^\circ t$ *associated with a transition* t_j *(i.e.* ${}^\circ t_j$*). Formally:*

$$2^P = \left\{ {}^\circ t_j \,\middle|\, j < m, \exists \ell_j \in L \right\}$$

Therefore, the function G is composed (i.e. union) of a number of sub-functions G_j*:*

$$\forall j < m \,\middle|\, \exists \ell_j \in L \implies G_j : {}^\circ t_j \to \{0,1\}$$

Where the evaluation of G_j *is performed*[4] *through the assessment of the predicate* $\ell_j \in L$*.*

2.2. Graphical model

Formal analysis of a IDR can be made in terms of graph theory. The structure of a classical PN [8] is analogue to a directed, weighted, bipartite graph. In order to include two domains (control and data) rather than one (control), the proposed modelling technique uses a directed, weighted, *tripartite* graph [2]. The proposed modelling technique has three type of vertices and two type of arcs, unlike classical PNs which has two type of vertices and one type of arc. Each *vertex* of the graph is represented by either a circle, a bar or a rectangle (i.e. either ◯, | or ▯) according to if it is a place, control transition or a data transition respectively. *Arcs* are employed to represent either control flow relation (i.e. F_C) or data flow relation (i.e. F_D), so the former is graphically represented with light directed arcs and the latter with bold directed arcs. In other words, light arcs are used for the flow of *tokens* ($\in$ control domain) and bold arcs for the flow of *values* ($\in$ data domain).

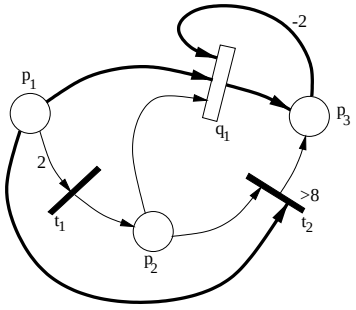


Figure 1. A structure with $n = 3$, $m = 2$ **and** $h = 1$

A simple structure is shown in Figure 1 where $n = 3$, $m = 2$ and $h = 1$. This is $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2\}$ and $Q = \{q_1\}$ respectively. Also several light and bold arcs can be easily identified as well as weights belonging to both W_C and W_D.

[3] Despite being widely used in the literature [1, 4–6], guard functions definition varies slightly from one HLPN to another

[4] *See Definition 6, Section 2.3*

2.3. Behavioural model

In addition to tokens, the system's dynamic behaviour is modelled by means of a new element called *value*. Therefore, a *marking* is an assignment of not only tokens but values to the places of the net. A token is a primitive concept derived from the classical PNs and kept for control flow representation, while values are introduced into the technique in order to allow data flow representation. Unlike classical tokens, values have a natural number attached which is an abstract representation of data domain. The marking function (μ) provides a state-oriented analysis to the model. For a certain marking μ given at a time step k, i.e. μ_k, changes in the embedded system state are represented as changes in its marking.

Definition 4 *A marking function (μ) is a mapping from the set of places P to the complex numbers $\mathbb{C}$. This is $\mu : P \to \mathbb{C}$*

Since Definition 4 assigns to each place $p_i \in P$ a complex number $\varsigma \in \mathbb{C}$, we say hereafter that p_i is marked with $\varsigma = \mu(p_i)$. The marking function has been defined in the complex domain. This implies the coexistence of two disjoint domains, which is a desirable feature for representing control and data flow in one model. A complex number can be represented as a composition of two real numbers: *modulus* and *phase*, and the relationship between them is given in Definition 5. It should be noted that both α and γ are natural numbers rather than real, since the discrete nature of the state space. This is possible because: $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C} \implies \mathbb{N} \subseteq \mathbb{C}$. We will use the modulus of a marked place for data domain representation and the phase for control domain representation.

Definition 5 *Each marked place $\mu(p_i)$ has two parts, namely value (α_i) and the number of tokens (γ_i), corresponding to the modulus and phase (respectively) of its complex number*[5].

$$\forall(\alpha_i,\gamma_i) \in \mathbb{N} \implies \exists \mu(p_i) \in \mathbb{C} \,\Big|\, \mu(p_i) = \alpha_i e^{\mathbf{i}\pi/2 \cdot \gamma_i}$$

Definition 5 puts forward a phase step of $\pi/2$ in order to reduce the complexity of mathematical analysis, since: $e^{\mathbf{i}\pi/2} = \mathbf{i}$. To extract modulus and phase information from a marked place (i.e. a complex number ς), the *mod* operator (i.e. $|\varsigma|$) and the *phase* operator (i.e. $\angle\varsigma$) has been defined and its syntax is:

$$\alpha_i = |\mu(p_i)| \quad , \quad \gamma_i = \angle\mu(p_i)$$

Figure 2 illustrates the use of Definition 4 and 5 with a simple marking on the proposed modelling technique. Place p_1 is marked with a value of $\alpha_1 = 10$ and two tokens (i.e. $\gamma_1 = 2$). As a consequence: $\mu(p_1) = 10e^{\mathbf{i}\pi/2 \cdot 2} = 10e^{\mathbf{i}\pi}$. On the other hand place p_3 is only marked with a value of $\alpha_3 = 2$, then $\mu(p_3) = 2e^{\mathbf{i}\pi/2 \cdot 0} = 2$. The marking μ of the set P at a time $k = 0$ is represented by a vector (i.e. $\mu_k(P)$) which has $\mu_k(p_1), \mu_k(p_2), \cdots, \mu_k(p_n)$ as components. This is $\mu_0 = (10e^{\mathbf{i}\pi}, 0, 2)^T$ for Figure 2's marking. It should be noted that values are represented by natural numbers stamped inside marked places and tokens are represented in the classical way, by black dots drawn in marked places. Definitions 6-10 are applied to this example, in order to provide a better understanding of the modelling technique's behaviour.

[5] *Note the difference between index i and the imaginary unit* **i**. *The unit* **i** *is the base of imaginary numbers (i.e. $\Im$), which is:* $\mathbf{i} = \sqrt{-1}$

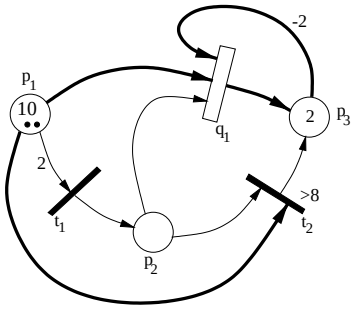


Figure 2. Initial marking $\mu_0 = (10e^{\mathbf{i}\pi}, 0, 2)^T$

Definition 6 *The guard function G is:*

$$G_j(p_i \in {}^\circ t_j) = \begin{cases} 1 & if\,(\forall p_i)\,[\ell_j(|\mu(p_i)|)] = \top \\ 0 & if\,(\exists p_i)\,[\ell_j(|\mu(p_i)|)] = \bot \end{cases}$$

Where: $\top$ stands for TRUE and $\bot$ stands for FALSE.

In Figure 2, a guard[6] function G_2 assess the value placed at p_1 (i.e. $G_2(p_1) = (|\mu(p_1)| > 8)$). The assessment for the marking μ_0 results TRUE, since $|\mu(p_1)| = 10$. Therefore, $G_2(p_1) = 1$.

For a certain marking μ_k, whether or not an active element in the model is enabled depends on the marking of passive elements. This is stated by Definitions 7 and 8.

Definition 7 *A transition t is said to be enabled if all the places of its pre-set ${}^\bullet t$ hold at least $W_C(p,t)$ tokens and the guard function G is valid for the values placed in pre-set ${}^\circ t$. This is:*

$$\Big[\angle\mu(p_i \in {}^\bullet t) \geqslant W_C(p_i,t)\Big] \wedge \Big[G(|\mu(p_j) \in {}^\circ t|) = 1\Big]$$

It should be noted that if no place exist in ${}^\circ t$, neither does the function $G(.)$. Therefore, the transition is then enabled only subject to the number of tokens placed in its pre-set ${}^\bullet t$, as in classical PNs.

Definition 8 *A transition q is said to be enabled when at least one place in its pre-set ${}^\bullet q$ holds $W_C(p,q)$ or more tokens. If ${}^\bullet q = \varnothing$, then transition q is always enabled. This is:*

$$\Big[\exists p_i \in {}^\bullet q \,\Big|\, \angle\mu(p_i) \geqslant W_C(p_i,q)\Big] \vee \Big[{}^\bullet q = \varnothing\Big]$$

Figure 2 shows a marking μ_0 where transition t_1 is enabled, since $\angle\mu(p_1) \geqslant 2$. Although $G_2(p_1) = 1$, Definition 7 requires a token placed in p_2 as well, in order to let transition t_2 to be enabled. Therefore, t_2 is not enabled. Also, transition q_1 is not enabled because there are no tokens in its pre-set ${}^\bullet q$.

Represented by a PN based modelling technique, the behaviour of an embedded system can be described in terms of firing a sequence of transitions. Definitions 9 and 10 states the firing rules for control and data transitions respectively.

Definition 9 *The firing of an enabled transition t changes a marking μ_k into μ_{k+1} by the use of the following rules:*

[6] Represented as a predicate attached to transition t_2, i.e. "> 8"

i. A finite number of tokens are removed from ${}^\bullet t$:

$$\forall p_i \in {}^\bullet t \implies \angle\mu_{k+1}(p_i) = \angle\mu_k(p_i) - W_C(p_i,t)$$

ii. A finite number of tokens are added to $t^\bullet$:

$$\forall p_i \in t^\bullet \implies \angle\mu_{k+1}(p_i) = \angle\mu_k(p_i) + W_C(t,p_i)$$

This is shown in Figure 3, where transition t_1 of Figure 2 has been fired according to Definition 9. Note that the new marking μ_1 presents two enabled transitions, i.e. t_2 and q_1.

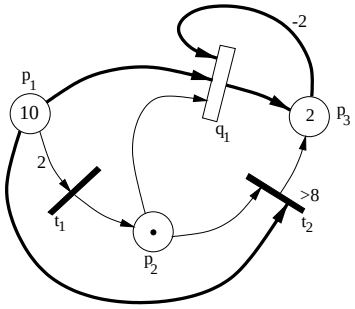


Figure 3. Marking $\mu_1 = (10, 0e^{\mathbf{i}\frac{\pi}{2}}, 2)^T$

Definition 10 introduces the functionality of a data transition, which is to perform a linear combination of all values stored at input places and put the result in its output place. It should be noted that a data transition has only one output, since the synthesis of a embedded systemspecification would consider arithmetic operations with only one result.

Definition 10 *The firing of an enabled transition q changes a marking μ_k into μ_{k+1} by:*

$$\forall p_i \in {}^\circ q, p_j \in q^\circ \implies |\mu_{k+1}(p_j)| = \sum_i |\mu_k(p_i)| \cdot W_D(p_i,q)$$

Although conditions for enabling either a transition t or q are mapped from both control and data domains[7] the firing of any transition does not combine information from the two domains, since Definition 9 takes place in control domain, i.e. $\{{}^\bullet t \cup t^\bullet\}$, and Definition 10 in data domain, i.e. $\{{}^\circ q \cup q^\circ\}$).

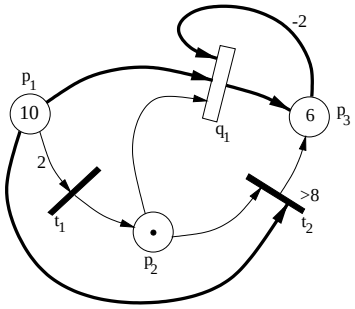


Figure 4. Marking $\mu_2 = (10, 0e^{\mathbf{i}\frac{\pi}{2}}, 6)^T$

[7]Definition 7 mapping's domain is $\{{}^\bullet t \cup {}^\circ t\}$, which is a combination of both control and data domains. Definition 8 mapping's domain is $\{{}^\bullet q\}$, which is purely within the control domain

Figure 4 shows the marking of the system (i.e. μ_2) after the firing of transition q_1, according to Definition 10. The value placed in p_3 is the result[8] of:

$$|p_1| \cdot W_D(p_1,q_1) + |p_3| \cdot W_D(p_3,q_1) = 10 + 2 \cdot (-2) = 6$$

3. Application Examples

Two examples are given in this section. The first example shows how the modelling principles are applied to the description of a simple multiplier. This example is also used to show how the modelling technique compares with ETPN [9]. The second example involves the modelling of a video processor's specification. This example is chosen to demonstrate the efficiency of the proposed modelling technique in dealing with complex embedded systems specification.

3.1. Multiplier example

Figure 5 shows the specification of a multiplier proposed in [1]. The algorithm calculates the product of two integers by means of iterative sums.

```
 1:  int mult(int a,int b) {
 2:    int x,y,z;
 3:    x=a;
 4:    y=b;
 5:    z=0;
 6:    while(y>0) {
 7:      z=z+x;
 8:      y=y-1;
 9:    }
10:    return z;
11:  }
12:
13:  c=mult(a,b);
```

Figure 5. Multiplier algorithm [1]

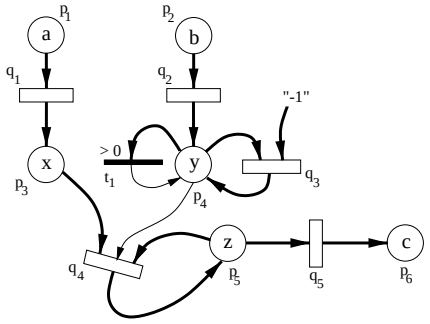


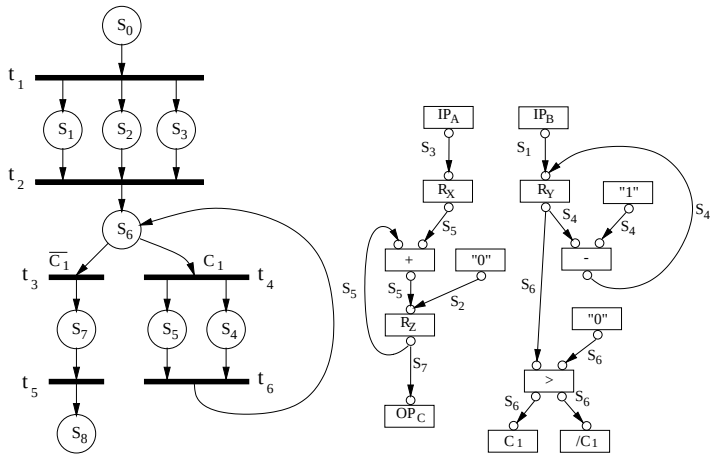
Figure 6. Proposed representation of the multiplier

The graphical representation of the multiplier using the proposed modelling technique is shown in Figure 6. To perform a multiplication by means of this model, we place a value in p_1 and other value in p_2. We assume neither p_1 nor p_2 receives another value within the process. According to Definition 8, both q_1 and q_2 are enabled. Then we can fire either q_1, q_2, or both, following Definition 10. This will lead to the movement of the two values into places p_3 and p_4. Some of the operations mapped into data transitions require a control signal to be executed. This is useful for defining several scopes for the operation. For example transition q_4 is enabled when p_4 holds a token and only transition t_1 can place a token there, since it is

[8]Note that ${}^\circ q = \{p_1, p_3\}$

the only transition that exists in ${}^\bullet p_4$. Furthermore, t_1 will generate a token each time it is fired, if the guard function ($G_1(p_4)$) is valid (i.e. $y > 0$) according to Definition 9. The value of p_4 for a certain marking (i.e. $|p_4| = y$) is changed only by transition q_3, since ${}^\circ p_4$ is composed of q_2 and q_3, but only q_3 will be active again according to our initial assumption. All these actions together represent the behaviour of the complete *while* loop in the specification given at the beginning of the section.

It is important to note that in the proposed modelling technique operations and conditions are modelled in a different way, when compared to ETPN [9] and PRES [1]. Operations are carried out by *data transitions* in such a way that a linear combination of the values contained in their input places is performed (See Definition 10). Conditions are carried out by *control transitions* and its guard function, which validates the firing of the transition based in the value placed at ${}^\circ t$. For example, transition t_1 in Figure 6 represents the condition "$y > 0$" in the *while* loop at Figure 5. The guard function $G_1(p_4)$ evaluates the content of p_4, resulting 1 if $|p_4| > 0$ or 0 otherwise.



(a) Control Part (b) Data Path

Figure 7. ETPN representation of the multiplier

In order to show the advantages of having one unified graph for modelling both data and control, we have constructed the ETPN model shown in Figure 7 for the multiplier example. Clearly the proposed modelling technique presents a more compact and easier to follow model.

3.2. Video processor example

In [13] the development of a spectrum analyser is presented. One of its main embedded system is a video processor. This processor has three processes (*memory*, *raster* and *render*). Figure 8 shows the Behavioural VHDL specification of the *raster* process, which has been modelled using the proposed modelling technique in Figure 9.

Transitions q_1, q_2 and t_4 represent some operations performed in the inner loop of the VHDL code, given by lines 19-27. All these transitions are enabled when p_2 holds a token. Transition t_3 can be interpreted as the *unconditional* way into the loop, since it is the only transition capable of introducing a token into p_2. Transition t_4 acts as a *conditional exit* to the loop, since it would take out the token based on the condition stated in the guard function $G(t_4)$ (This is equivalent to the execution of line 25 in Figure 8). The VHDL code has four nested

```
 1:  Ports{  vclk       : buffer bit;                          -- p1
 2:          hsync      : out bit;                             -- p7
 3:          vsync      : out bit;                             -- p9
 4:  }
 5:  Interprocess signals {
 6:       signal raster_sem : bit := '1';
 7:       signal memory_val : bit_vector(7 downto 0);         -- p3
 8:  }
 9:  raster: process
10:    variable pixel  : bit_vector(7 downto 0);             -- p4
11:    variable pix_no : integer range 0 to 7;               -- p2
12:    variable read   : bit_vector(8 downto 0);             -- p6
13:    variable line   : bit_vector(8 downto 0);             -- p8
14:  begin
15:    vsync <= '1';
16:    refresh_loop: loop
17:      vert_loop: loop                                     -- t1
18:        horiz_loop: loop                                  -- t2
19:          block_loop: loop                                -- t3
20:            hsync <= '1';                                 -- q9
21:            vclk <= '0';                                  -- q1
22:            wait for 100 ns;
23:            vclk <= '1';
24:            wait for 100 ns;
25:            exit when pix_no = 7;                         -- t4
26:            pix_no := pix_no + 1;                         -- q2
27:          end loop block_loop;
28:          pix_no := 0;                                    -- q3
29:          if line(8) = '0' then       -- (line < 256)?    -- t5
30:            pixel := memory_val;                          -- q4
31:            raster_sem <= not raster_sem;                 -- q5
32:          end if;
33:          exit when read = convert_int2bv(39,9);          -- t6
34:          read := read + convert_int2bv(1,9);             -- q6
35:        end loop horiz_loop;
36:        vclk <= '0';                                      -- q1
37:        hsync <= '0';                                     -- q8
38:        vsync <= '0';                                     -- q10
39:        wait for 100 ns;
40:        read := convert_int2bv(0,9);                      -- q7
41:        read := convert_int2bv(0,9);                      -- q7
42:        exit when line = convert_int2bv(259,9);           -- t7
43:        line := line + convert_int2bv(1,9);              -- q11
44:      end loop vert_loop;
45:      line := convert_int2bv(0,9);
46:      vsync <= '1';
47:      wait for 0 ns;
48:    end loop refresh_loop;
49:  end process;
```

Figure 8. Raster process [13]

loops (i.e. *refresh*, *vert*, *horiz*, *block*). Places p_9, p_8, p_6 and p_2 are related to each loop in the following way: Holding a token in p_2 is associated to the execution of the inner loop whereas holding a token in p_9 is related to the outer loop. Pair of transitions $\{t_1, t_7\}$, $\{t_2, t_6\}$ and $\{t_3, t_4\}$ describe a bidirectional path which represents moving towards the inner or the outer loop. A subgraph of Figure 9 which concentrates more in the main control flow between loops, is shown in Figure 10.

4. Conclusions

This paper has proposed a new modelling technique for embedded systems specification where control and data flow are tightly linked. The technique is based on a modified Petri Net structure consisting of dual transition and arc definition, leading to share control domain and data domain information in a single unified graph. The union between control and data domain is provided by the definition of a guard function (Definitions 3 and 6). One of the main features of the proposed modelling technique, is the ability of dealing easily with nested loops and conditional operations, which may lead to better results in the co-design of embedded systems specification. This area of research is currently being investigated by the authors.

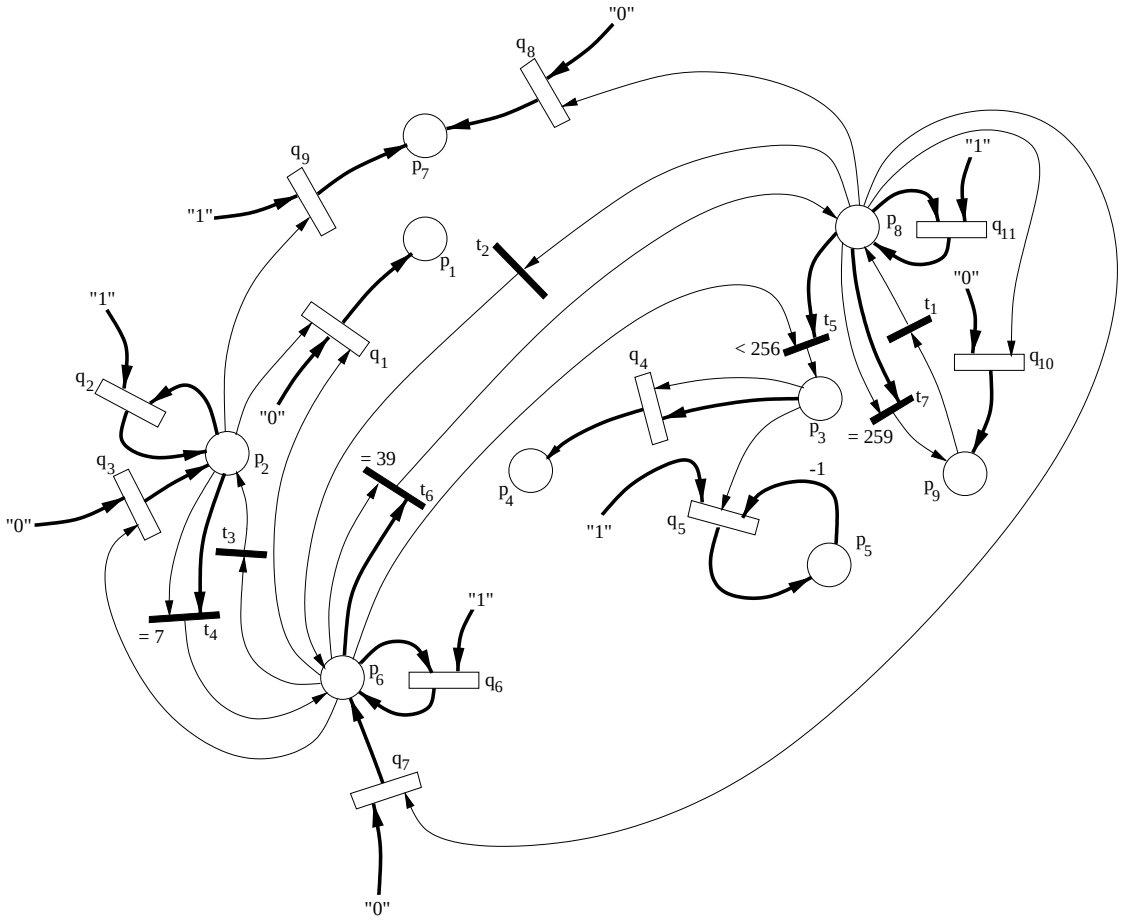


Figure 9. Proposed model of the raster process specification

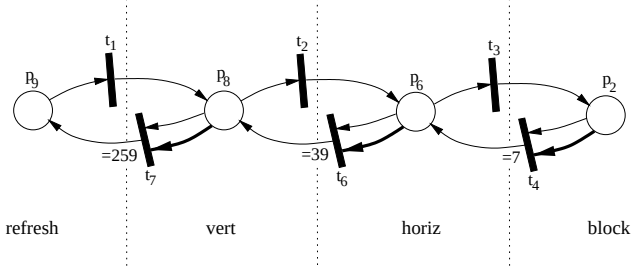


Figure 10. Control Flow between loops

References

[1] L. A. Cortés, P. Eles, and Z. Peng. A Petri Net Based Model for Heterogeneous Embedded Systems. In *IEEE NORCHIP Conference*, pages 248–255, Oslo, Norway, Nov. 8-9 1999.

[2] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2nd edition, 2000.

[3] H. Genrich. Predicate/Transition-Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, 1987.

[4] K. Jensen. Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts. *EATCS Monographs in Theoretical Computer Science*, pages 1–234, 1992.

[5] K. Jensen. Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 2: Analysis Methods. *EATCS Monographs in Theoretical Computer Science*, 1994.

[6] K. Jensen. Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 3: Practical Use. *EATCS Monographs in Theoretical Computer Science*, 1997.

[7] G. D. Micheli and R. K. Gupta. Hardware/Software Co-Design. *Proceedings of the IEEE*, 85(3):349–365, Mar. 1997.

[8] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, Apr. 1989.

[9] Z. Peng and K. Kuchcinski. Automated Transformation of Algorithms into Register-Transfer Level Implementations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(2):150–166, Feb. 1994.

[10] J. Peterson. *Petri net theory and the modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

[11] M. Sgroi, L. Lavagno, and A. Sangiovanni-Vincentelli. Formal models for embedded system design. *IEEE Design & Test of Computers*, 17(2):14–27, Apr.-June 2000.

[12] E. Stoy. A Petri Net Based Unified Representation for Hardware/Software Co-Design. Licentiate thesis: LiU-Tek-Lic 1995:21, Linköping University, S-581 83, Sweden, 1995.

[13] A. Williams. *A Behavioural VHDL Synthesis System using Data Path Optimisation*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK, 1997.