CDFG -Petri Net Temporal Partitioning for Switching Context Applications

Paulo Sérgio B. Nascimento, Manoel E. Lima, Paulo Maciel
Abel G. S. Filho, Edna Barros and Sérgio Cavalcante
Centro de Informática - CIn / UFPE - P.O. Box 7851 - Cidade Universitária
Phone: +55 081 32718430, Fax: +55 081 32718438
Recife - PE - Brazil - psbn, mel, prmm, agsf, ensb, svc@cin.ufpe.br

Abstract

The reconfigurable computing has presented a promising progress. The development of platforms that match the reconfigurable hardware with programmable elements, such as DSPs or microprocessors promise great applicability in diverse areas in the future. In this article we describe a co-design methodology for single-context and virtual hardware applications based on Petri net for PISH design environment. A platform description is presented as well as the reconfiguration methodology. A temporal partitioning is described exploring different solutions for different applications with time and communication constraint.

1 Introduction

With the increasing in the complexity and size of the digital systems, more powerful CAD tools are needed to speed up the validation processes and the prototyping platforms in digital system designs. Many rapid prototyping platforms have been launched aiming at giving digital system designers the option to implement their projects in a fast and reliable way, aiming at cost reduction, better performance and minor time -to-market. Prototyping platforms are in general based on single or multiple reconfigurable hardware [7],[8]. Generally, hardware implementations allow concurrency explotation and increasing data processing speed. These hardware components will also continue to grow and so the problem of fitting them into a single reconfigurable component such as a FPGA or an ASIC would not be always possible.

Software implementations present some advantages such as flexibility and low implementation cost of complex functions. However, it presents limitations as difficulties to exploiter parallelism and meet constraints for high-speed applications. Both worlds, hardware and software, however, have to work together aiming to reduce time -to-market, costs and improving performance.

In a hardware/software codesign methodology, the advantages of both approaches are taking into account. This methodology offers flexible alternatives based on criteria and constrains for hardware and software components (cost function), such as silicon area and speed. Software components are represented by microcontrollers or microprocessors and hardware ones by ASICs or FPGAs [7],[8].

In this article, we present a codesign methodology for switching context application based on a CDFG-Petri net model [13],[16]. This approach is integrated with PISH environment (Integrated Design of Sw/Hw) [2] and Chameleon prototyping platform [12]. The prototyping platform allows run time switching context in a XC400XX FPGA Xilinx series. The platform contains a microcontroller that manages each access to the FPGA and its configurations. In Section 2, the methodology and the prototyping environment are described. Section 3 presents the Petri net model. Section 4 presents CDFG - Petri net model used for control and data flow representations. Section 5 discusses hardware area estimation. Section 6 introduces temporal partitioning tools. Section 7 presents a complete example and some results. Finally, Section 8 presents conclusions and future works.

2 PISH codesign system overview

The PISH codesign methodology (see Figure 1) aims to provide a complete top-dow environment for hw/sw codesign, from the system specification to its prototyping. Its current specification mechanism uses Occam [10] as specification language.

After the specification, the system is partitioned [3].

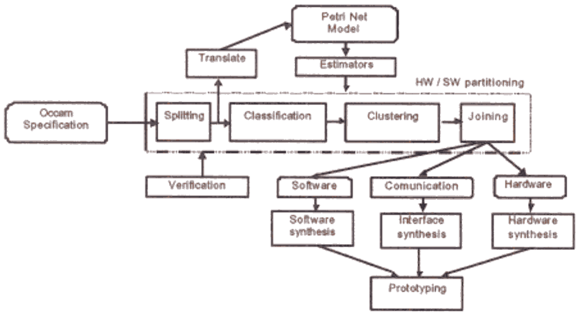


Figure 1 – PISH methodology

The partitioning, through a formal mechanism, guarantees the semantics preservation of the initial specification. The analysis phase is carried out considering a Petri net representation [6],[15],[16] of the description as a model to perform quantitative analysis and metrics computation (time and area). An automatic interfaces generator [1] implements the hw/sw communication after partitioning. As results, the software and hardware components can be implemented in the Chameleon prototyping platform [12].

2.1 Chameleon prototyping platform

The Chameleon platform is showed in Figure 2. The platform is composed of software component, a microcontroller 8051 type, a hardware component, a Xilinx FPGA, and two banks of memory to keep program and data.

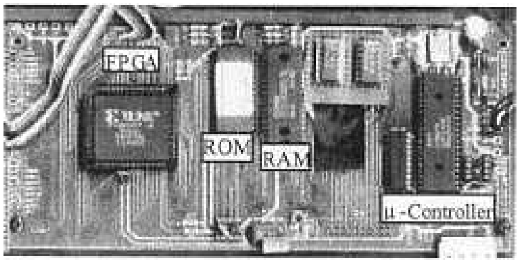


Figure 2 – Chameleon platform

An overview of the prototyping environment is depicted in Figure 3.

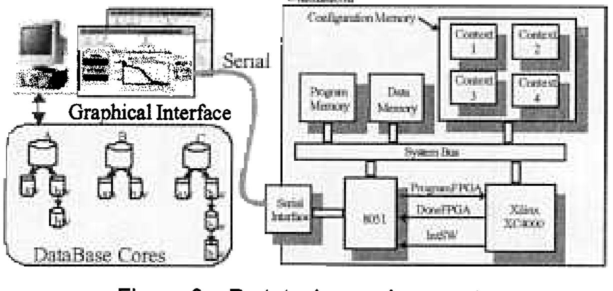


Figure 3 – Prototyping environment

A graphical user interface has been developed in order to provide a friendly design environment. A database storages all hw and sw application cores, and a supervisory system, installed in the PC host, manages the download of all applications, through a serial link with the platform.

2.2 Run time reconfiguration

In general, a complex hardware specification is represented by a single file that can be too large to be fitted into a single FPGA. Thus, the circuit requires splitting the original file into small ones in order to fit the logic into the FPGAs. Multi-FPGAs platforms [7], [12] are possible solutions since it is possible to split the system into smaller processes adjustable to the platform. However, this approach demands hardware area proportional to the system complexity. In our model, the context partitions could be switched at a time in a single FPGA, according to a scheduling algorithm. The methodology suggests a virtual hardware mechanism like a virtual memory [9]. The switching context is based on Petri net analysis. Figure 4 shows the design flow for hardware and software components considering a single FPGA platform.

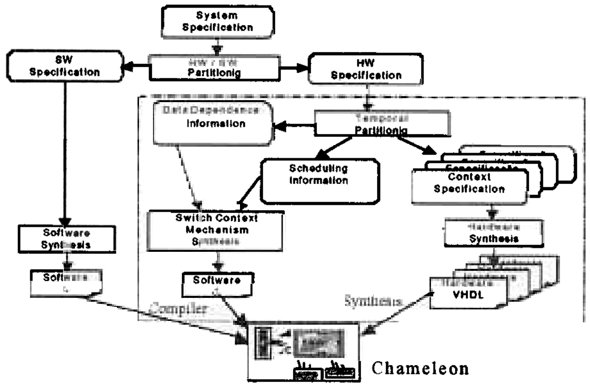


Figura 4 – Design flow

3 Timed Petri Net

Petri nets are formal specification techniques that allow a graphical and mathematical representation and have powerful methods for qualitative and quantitative analysis [14]. Petri nets are used to systems modeling [17], [18]. Petri net temporal approach was proposed by Ramchandani [19].

Timed Petri nets are Petri net extensions in which the time information is expressed by duration (deterministic timed net with three phase policy firing semantics) and is associated to the transitions.

Definition1– Timed Petri Nets: *Let $Nt = (P,T,I,O,M_0,N,D,C)$ be a timed Petri net, , $D:T \rightarrow A$, $A=\{ a \mid a > 0, a \in R \}$ is a function which associates with each transition t_i the duration of the firing d_i, $C: T \rightarrow B$, $B=\{ b \mid b \in [0,1], b \in IR \}$ is a choice function which assigns a free-choice* *probability to each transition of the net, where $\sum_{t \in Tc} C(t) = 1$. $Tc \subseteq T$ is a set of structural conflicting transitions.*

4 Intermediate model

After the hw/sw partitioning, the Occam code is translated into a Petri net intermediate representation (see Figures 5 and 6). In this section, the control flow and data models are defined [13],[15]. The formal definitions are described below.

Definition2 – Control Flow Model: *PN^c is safe and timed Petri net $Nt=(P^c,T^c,f,O^c,M_0^c,D,C)$ as defined in* ***Definition1****, Section 3, and represents the control flow.*

Definition3 – Data Flow Model: *$G^d=(V=T \cup In \cup Out, E, FD)$ is the data dependence graph. T is Petri net transitions set; In and Out are data input and data output points respectively; $E \subset V x V$ is the edge set that represents data dependence between vertex of G^d in V set; FD(e) is the variables sub-set associated to the data flow represented for $e \in E$.*

Figure 5 shows a simple example, in order to explain the principles of our approach. The example written in Occam is translated into control (PN^c) and dataflow (G^d) models. These models are described in Figure 6 as control data flow graph.

The construction rules of net PN^C are described in [15]. Although the model has been originally conceived for Occam description, other languages such as SystemC [20] can also be considered.

```
INT a,b,c,d,e
CHAN OF INT ch
PAR
    SEQ
        a := a + 1
        b := a + 2
    SEQ
        IF c < 0
            c: = c + 1
        IF c >= 0
            c: = c + 2
        ch ! c
    SEQ
        ch ? d
        d : = d + e
```

Figura 5 – A Occam code example

5 Hardware area estimation

Before the hardware synthesis, it is necessary to estimate its total area. For a given high-level behavioural description, the hardware design is divided into two classes of functional blocks: data-path and controller.

This estimate is accomplished considering the CDFG $=(PN^c,G^d)$ and hardware models. The method for estimating hardware area is described in [15], [16].

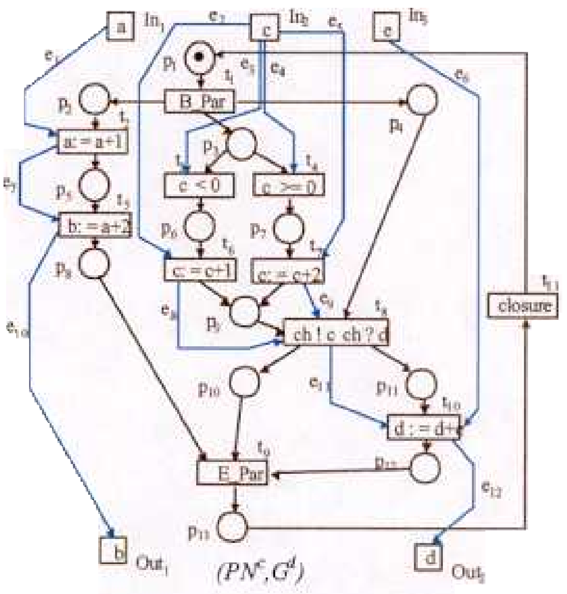
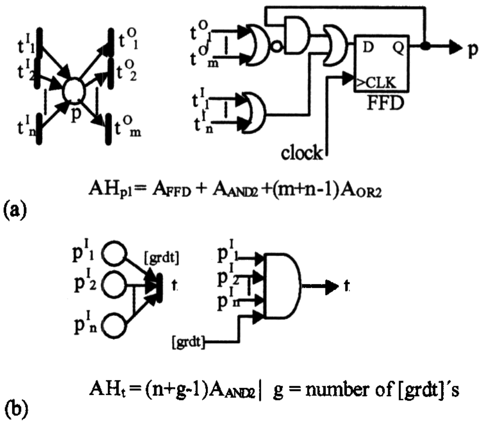


Figure 6 – CDFG – Petri net model

5.1 Hardware model

The controller is synthesized directly from PN^c [15], where each place is associated to a flip-flop type D and each transition associated to an AND gate as it is depicted in Figure 7.



t^I_1
t^I_2
t^I_n
t^O_1
t^O_2
t^O_m
p
D Q
>CLK
FFD
clock
$AH_{pl} = A_{FFD} + A_{AND2} + (m+n-1)A_{OR2}$
(a)
p^I_1
p^I_2
p^I_n
[grdt]
t
$AH_t = (n+g-1)A_{AND2}$ | g = number of [grdt]'s
(b)

Figure 7 – Hardware model for controller: place (a) and transittion (b).

In Figure 7, AH_{pl} and AH_t are the areas for implementation of a *p* place and *t* transition respectively.

These metrics are given by the expressions:

$AH_{pl} = A_{FFD} + A_{AND2} + (m+n-1) A_{OR2}$ (1)

$AH_t = (n+g-1) A_{AND2}$, g = number of [grdt] (2)

A_{FFD} is the area of the Flip-Flop D, A_{AND2} and A_{OR2} are the areas of the AND and OR gates of 2 input. The data-path circuit consists of registers, multiplexers and functional units defined in the Data-path Components Library: $DCL = \{R1$(1bit register), $MUX2$ (multiplexer2:1 bits) $\} \cup FUL$. $FUL = \{FUi\ ,\ i = 1,2,..\ ,n\}$ is a Functional Unit Library. Figure 8 illustrates the data-path models. The DCL component areas are designated by: AH_{R1}, AH_{MUX2}, AH_{FUi}.

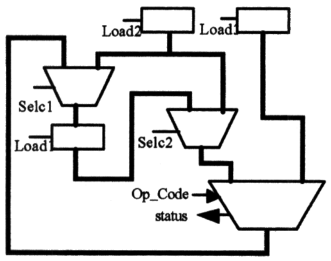


Figure 8 – Data-path hardware model

The area estimation is carried out based on equivalent gates number for the XC400XX FPGAs [13], considering control and data-path areas:

$$AH = AH_{CTRL} + AH_{DP} \qquad (3)$$

5.2 Area estimation techniques

The control area estimation are obtained directly from expressions (1) and (2):

$$AH_{CTRL} = \Sigma_{\forall p \in P^c} AH_{pl}(p) + \Sigma_{\forall t \in T^c} AH_t(t) \qquad (4)$$

For data path area (AH_{DP}), it is important to find out the number of registers, multiplexers and functional units.

The registers area (AH_{reg}) is calculated as a function of the number of variables used in the process (VP). If VP is the set of variables in the process and $/v/$ is the length in bits of $v \in VP$, Then the registers area used is:

$$AH_{reg} = \Sigma_{\forall v \in VP} /v/ AH_{R1} \qquad (5)$$

In the same way, multiplexers area (AH_{mux}) is estimated as a function of the number of incidences for each operation in hardware process. If OP is a set of incident operations, $N(op)$ is the number of times that $op \in OP$ appears in the process and $/op/$ is the length in bits of the op inputs. Then, the area estimated of mu ltiplexers is given by:

$$AH_{mux} = \Sigma_{\forall op \in OP} N(op) /op/ AH_{MUX2} \qquad (6)$$

The expressions (5) and (6) do not take into account register and functional units reutilization. Hence, the estimates obtained may not be very accurate. For functional units area estimates, (AH_{fu}), the results are closer to real metrics. If $\forall op \in OP$, $\exists\ FU_{op} \in FUL$ that implements op, and $N(FU_{op})$ is the number of functional units of the type FU_{op}, then the estimate area for functional units is given by:

$$AH_{fu} = \Sigma_{\forall op \in OP} N(FU_{op})\ AH_{FUop} \qquad (7)$$

Data-path area is the summation of (5), (6) and (7):

$$AH_{DP} = AH_{reg} + AH_{mux} + AH_{fu} \qquad (8)$$

5. 3 Functional unit estimation

The calculation of the number of functional units $N(FU_{op})$, in the expression (7), is accomplished in two parts [15],[16]. First, the set $P_{MI}(PN^c)$ of p-minimus invariants supports [14] for PN^c is calculated and the universal set of transition-paths $UTPS(P_{MI}(PN^c))$ is generated. Let T_{op} be the PN^c transitions set that represent operations op. The upper limit of $N(FU_{op})$ is given by the smallest number of sets TS$_i$ ($TS_i \in UTPS(P_{MI}(PN^c))$), whose union covers T_{op}. It is because the paths of transition TS may represent transitions that has exclusion relation in the PN^c

The second stage of the $N(FU_{op})$ calculation consists of a refinement of the bound obtained in the first stage. PN^c is extended for PN^{ce} by introduction of a set of places P^{FU}. P^{FU} represents functional units and arcs among these places and transitions represent the operation allocation. The amount of tokens in these places is represent by $N(FU_{op})$. The execution time of the net PN^{ce} is computed, by using the INA tool [18], for combinations of initial marking of P^{FU} starting from the upper minimal bound, until the minimum amount of functional units that do not increase the execution time. This method has provided good results for the area of functional units.

6 Temporal partition and virtual hardware

Figure 9 illustrates de temporal partition for virtual hardware methodology. The hw process may be represented by a CDFG-Petri net model $P^{HW} = (PN^c, G^d)$. In case the estimated area is larger than the available FPGA area, the hw process is partitioned into multiple contexts that are performed in a time multiplexing approach. In [11] is presented temporal partitioning from a data flow graphs hardware description. In [4], [5] is presented a hardware gate level partitioning. In this work, The PN^c.and G^d is partitioned at the operation level. Each context has a controller that is derived from a partition $PN^c_p \subseteq PN^c$, and a data-path obtained from a partition $G^d_p \subseteq G^d$.

The foresee methods described in Section 5 are used to guide the partitioning of the (PN^c, G^d) in subprocesses placed, as contexts, into FPGA.

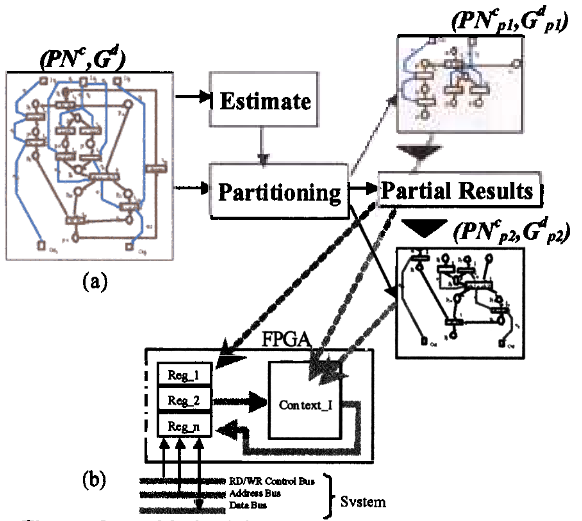


Figure 9 – Methodology to virtual hardware: a) temporal partitioning, b) Context implementation in FPGA

6.1 Multi-contexts implementation

Contexts are implemented, in the FPGA, a structure as showed in the Figure 9(b). The context switching is performed by microcontroller. FPGA reconfiguration spends several milliseconds in XC400XX chips. For industrial low speed applications, this overhead can be supported. *Reg_1* to *Reg_n*, used as partial result I/O ports, are accessible through the system bus. In the end of each context, the registers with partial results are saved in the RAM memory and, after the FPGA reconfiguration, sent to the registers of the next context.

6.2 Temporal partitioning algorithm

The partitioning algorithm groups transitions into contexts:

Definition4 – P^{HW} partitioning: *Let a pair P^{HW} = (PN^C, G^d) be a CDFG hardware process implemented in an FPGA area AH_{FPGA}, we defined a P^{HW} partitioning as being a contexts set $PartT^{HW} = \{C_1, C_2,\dots, C_K\}$ and $C_i.T \cap C_j.T = \phi\ \forall\ C_i,C_j \in PartT^{HW}$, $i \neq j$. $C_i.T$ is the transitions set contained in each context; $C_i.A_{HW}$ is the FPGA area used in each context and context execution order is $C_1 \rightarrow C_2 \rightarrow \dots C_k$. $PartT^{HW}$ Constraints is:*

Area: $C_i.A_{HW} \leq AH_{FPGA}\ \forall C_i \in PartT^{HW}$.

Precedence: $t \in C_i.T \Rightarrow (\forall\, t' \in I(p) \wedge p \in I(t): \exists\, C_j \,/\, t' \in C_j.T \wedge j \leq i)$.

When a context cannot group more transitions without violating the *Area* constraints, a new context is created. The order which transitions are grouped must respect the *Precedence* constraint. Figure 10 shows the group transitions method.

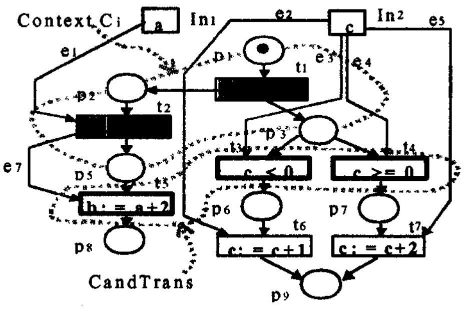


Figure 10 –Group transitions methods

Transitions t_1 and t_2 form context C_i. The CandTrans set $\{t_3,t_4,t_5\}$ represents the candidates to be grouped in C_i without violate *Area* and *Precedence* constraints. The grouping method adopted uses local criteria (myopic heuristics) for transition choice:

Concurrency Criterion: *Group the transition that exploits **less concurrency** or Group the transition that exploits **more concurrency**.*

Communication Criterion: *Group the transition that **minimizes the communication** between contexts.*

In Figure 10, the criterion ***more concurrency*** implies the choice of transitions t_3 and t_4. During the partitioning, transitions are searched in width, considering parallelism exploration and execution time minimization. The criterion ***less concurrency*** implies the choice of transition t_5: b:=a+2. Hence, transitions are searched in depth take into account parallelism minimization. The criterion ***minimizes the communication*** implies the choice of the transition that results in less data edges, in graph G^d, crossing the context. Transition t_5 is chosen, implying parallelism reduction and execution time increasing.

7 Case study

The methodology presented was applied to Occam representation that describes a differential equation:

$$x(t + i) = x(t) + y(t).i$$
$$y(t + i) = y(t) - f(x).y(t).t.i - 3.x(t).i$$
$$t' = t + i \text{, new t value}$$

The CDFG-Petri net resulted in 40 places and 39 transitions. Table 1 shows the temporal partitioning results according to two criteria:

Table 1 – Temporal partitioning results

Partitioning 1				Partitioning 2			
Criterion : More Concurrency				**Criterion** : Minimizes Communic.			
Ctx	**A**	**T**	**C**	**Ctx**	**A**	**T**	**C**
1	4987	4	13	1	4768	8	11
2	4377	5	15	2	4512	5	17
3	4995	6	19	3	2589	5	17
4	1053	2	9				
Total	15412	17	56	**Total**	11869	18	45

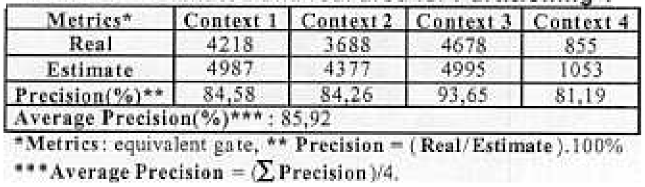
Ctx = Context, **A** = Area(equivalent gates), **T** = Time(cycles) and **C** = Communication (bytes). FPGA area is AH_{FPGA} = 5000 equivalent gates.

The context execution time (T) is estimated on the basis of the critical transition path extracted using techniques for search in graph.

For the first contexts (Table 1), we can observe that the *concurrency* criterion takes less execution time than the *communication* criterion. It is due to the parallelism exploitation (**Partitioning 1**) and reduction of communication (**Partitioning 2**). Next contexts group sequential transitions. This causes low sensitivity to the choice criteria. Considering as reference a partitioning that places all processes in a single context, **Partitioning 1** takes 40,64 % and **Partitionig 2** takes 22,92% more area than the circuit in single context, respectively.

The CDFG-Petri net $P^{HW} = (PN^C, G^d)$ and the partitioning result $PartT^{HW}$ had been translated manually into a RTL-level hardware description for each context. The Xilinx Foundation 3.1i synthesis results are showed in the Table 2.

Table 2 – Estimated and real area for **Partitioning 1**



Metrics*	Context 1	Context 2	Context 3	Context 4
Real	4218	3688	4678	855
Estimate	4987	4377	4995	1053
Precision(%)**	84,58	84,26	93,65	81,19
Average Precision(%)*** : 85,92				

*Metrics: equivalent gate, ** Precision = (Real/Estimate).100%
***Average Precision = ($\sum$ Precision)/4.

Good results have been obtained for area estimates with precisions around 86% in relation to Xilinx Syntheses Tool. In this architecture, the time for FPGA reconfiguration takes 16 ms. Depending on the number of contexts and application constrains the FPGA reconfiguration time may be suitable.

8 Conclusion and future works

A switching context methodology based on Petri nets was presented. Large algorithms may be split into suitable hw and sw processes and run in switching context platform. Estimates from Petri net guarantees that hw processes areas are suitable for the area available in the prototyping partform. A virtual hardware mechanism allows correct hw execution and its FPGA reconfigurations. An example has been presented in order to validate the method for low-end speed applications. Hardware reconfiguration speed may be improved with partial reconfiguration devices.

The partitioning algorithm groups transitions using local criteria. To prevent sub-optimal solutions, we intend to introduce strategies for exploiting solution space taking into account real time and communication constraints.

9 References

[1] Araújo, C.C., E. Barros, "A approach for Interface Generation in the PISH Codesign System ", Proceedings of SBCCI99, Brazil.

[2] Barros, E. et al., "Hardware/Software Codesign in the PISH project", procedings of the II Brazilian Workshop on Hardware/Software Codesign, 1997.

[3] Barros, E.; and Sampaio, A. "Towards provably correct hardware/software partitioning using occam", 1994, pp.210-217.

[4] Douglas Chang, Malgorzata Marek-Sadowska, "Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs", preceedings of IEEE Transaction on Computers, vol 48, No 6, June 1999, pp 565-578.

[5] E. Cantó, J. M. Moreno, Joan Cabestany, I. Lacadena, J. M. Insenser., "A Temporal Bipartitioning Algorithm for Dynamically Reconfigurable FPGAs", preceedings of IEEE Transaction on Very Large Scale Integration (VLSI) Systems, vol 9, No 1, February 2001, pp 210-218.

[6] F. Cruz Filho., "Particionamento em hardware software usando redes de petri", MSc Thesis, Centro de Informática – UFPE, Brazil,2000.

[7] Hauck, S., "The future of reconfigurable systems", Keynote Address,5th Canadian conference on field programmable devices, Montreal, 1998.

[8] Hauck, S., "The holes of FPGAs in reprogramable systems", preceedings of IEEE, vol. 86, No 4, 1998, pp 615-638.

[9] Hauck, S.; Compton K.,"Configurable Computing: A Survey of Systems and Software". submitted *to* ACM Computing Surveys, 2000.

[10] INMOS Limited, "occam 2 Reference Manual" , Prentice Hall International Series in Computer Science, 1988.

[11] Karthikeya M. Gajjala Purna, Dinesh Bhatia, "Temporal Partition and Scheduling Data Flow Graphs for Reconfigurable Computers", preceedings of IEEE Transaction on Computers, vol 48, No 6, June 1999, pp 579-590.

[12] Lima, M. E., D. S. Silva, D. G. Ramalho, A. V. Burgos, "Chameleon-I: A Rapid Prorotyping Multi-FPGA Platform for PISH Codesign System", SBMicro2000 - XV International Conference on Microelectronics and Packaging", pp. 86-91.

[13] Lima, Manoel E., Filho, Abel G.S., Nascimento, Paulo Sérgio B., Maciel, Paulo., "A reconfigurable codesign methodology for switching context applications based on Petri net". High Performance Computing Symposium 2002, v1, pp. 306-313.

[14] Murata T., Petri Nets: Properties, Analysis and Applications, Proceeding of The IEEE, 1989.

[15] Maciel, P.M.; "Petri Net Based Estimators for Hardware/Software Codesign". Doctor Degree Thesis. Centro de Informática – UFPE, Brasil, 1999.

[16] Maciel P. R., E. Barros, M. de Lima, D. S. da Silva, "Resource Sharing Estimation by Petri Nets in PISH Codesign System", HPC 2000 Special Track on Petri Nets and Performance Evaluation in HPC2000 , Washington.

[17] N. G. Leveson, J. L. Stolzy. Safety Analysis using Petri Nets. IEEE Transaction Software Eng., vol. SE-13, no. 3 , March 1987.

[18] P. H. Starke and S. Roch. INA – Integrated Net Analyzer – Version 2.2. Humbolt Universität zu Berlin – Institut für Informatik, 1999.

[19] Ramchandani Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Technical Report n 120 , laboratory for Computer Science, MIT, Cambridge, MA, 1974.

[20] www.systemc.org.