Algorithm for Switching Context Temporal Partitioning Based in CDFG-Petri Net Model

Paulo Sérgio B. Nascimento, Manoel E. Lima and Paulo R. M. Maciel
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@cin.ufpe.br

Keywords. *Virtual Hardware, FPGA, Switching Context, Temporal Partitioning, Petri Nets, Pipeline.*

Abstract

The reconfigurable computing has seen many promising advances in recent years. The development of platforms that match reconfigurable hardware with software elements, such as microprocessors, promise great applicability in diverse areas in the future. This article describes a algorithm for switching context temporal partitioning, based on a control data flow representation of hardware components, called CDFG-Petri Net model. The proposed partitioning algorithm allows to implement attached co-processors to speed up slow procedures in a hardware/software co-design approach, using a context switching model.

1 INTRODUCTION

With the increasing size of the digital systems, more powerful system architectures are needed to increase the speed of application. Generally, hardware implementations into a FPGA allow concurrency and increasing data processing speed. Software implementations present some advantages such as flexibility and low implementation cost of complex functions; however it presents limitations as difficulties to explore parallelism and meet constraints for high speed applications [2] [3] [4]. In a hw/sw codesign methodology [1], the advantages of both hw and sw are taking into account. This methodology break the system in software and hardware components with objective of accelerating the application. In general, a complex hardware component specification is represented by a single and large file, that can be too large to be fitted into a single FPGA. Thus, the circuit mapping requires splitting the original file into small ones in order to fit the logic into the FPGAs. Multi-FPGAs platforms [3] [5] are possible solutions since it is possible, from one given specification, to split the system into smaller processes adjustable to the platform. However, this approach demands hardware area proportional to the system complexity, increasing the cost of solution. In our model, the partitions on contexts could be switched at time in a FPGA, according to a scheduling algorithm [6]. The methodology suggests a virtual hardware mechanism like a virtual memory [4].

In this article we present a temporal partitioning algorithm based on CDFG-Petri Net model [6] for switching context co-processor implementation. Petri net analyses and metrics [7] [8] are used, having as target the Xilinx XC4000 family FPGAs. This algorithm allows the division of the process in time multiplexing contexts for execution in low cost hardware, based on static reconfigurable devices (FPGAs). Two stages pipeline structure can be used to decrease the effects of the FPGA reconfiguration time in the system performance. Section 2 presents a *CDFG-Petri Net* model [6] used for control-data flow representations. Section 3 presents the hardware context circuits model and switching context mechanism. Section 4 discusses temporal partitioning algorithm. Section 5 presents a complete example and some results. Finally, section 6 presents conclusions and future works.

2 CDFG-TIMED PETRI NET MODEL

In the proposed model, the hw is represented for a Control Data Flow Graph, showed in Figure 1, called *CDFG-Timed Petri Net Model* [6] [8].

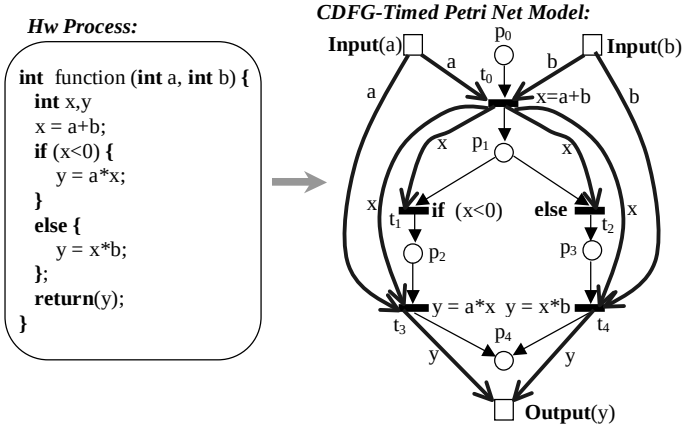


Figure 1: CDFG-Timed Petri Net Model

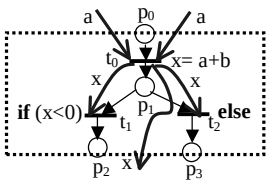
Timed Petri nets are a formal family of specification techniques that allow for a graphical, mathematical representation and have powerful methods for performing qualitative and quantitative analysis [7]. Time information is expressed by duration and is associated to the transitions:

Definition of Timed Petri Nets: *Let a pair PN = (N,D) be a timed Petri net, where N = (P,T,I,O,M_0) is a Petri net [7], D:T $\rightarrow$A, A={ a | a > 0, a $\in$ R } is a function which associates with each transition t_i the duration of the firing d_i,.*

The transitions $t_i \in T$ represent operations and the duration d_i represent the execution time of transition t_i. operation. Places $p_j \in P$ interconnect transitions to establish an operation control sequence. Timed Petri net form allows the application of the associated mathematical formalism for verification of the system, properties extraction and estimates [7] [8]:

Definition of CDFG - Petri Net*: is a pair (PN^C,G^D) where: PN^C is safe [7] and Timed Petri net and represents the control flow. G^D = (V, E, FD), V = T $\cup$ In $\cup$ Out, is the data dependence graph. T is Petri net transitions set; In and Out are respectively data input and data output points; E $\subset$ VxV is the edge set that represents data dependence among the vertex of G^D in V set; FD(e) is the variables subset associated to the data flow represented for e $\in$ E.*

When the hardware represented by the CDFG is not possible to be fit into the FPGA, it is partitioned in multiple contexts. Each context is a transition subset. The contexts are then implemented in a time-multiplexed approach as a Virtual Hardware Circuit [4] [6]. Figure 2, shows C_i context model defined by the transition subset $C_i.T$.



$C_i.T = \{t_0,t_1,t_2\}$
$C_i.P_{IN} = \{p_0\}$
$C_i.P_{OUT} = \{p_2,p_3\}$
$C_i.V_{IN} = \{a,b\}$
$C_i.V_{OUT} = \{x\}$

Figure 2: Context Model (C_i).

For each context, we can define a limit formed for an input places set $C_i.P_{IN}$ and for an output places set $C_i.P_{OUT}$. The context limits of C_i cuts the data dependence graph G^D in a set of entrance data edges $C_i.V_{IN}$, that represents data that will be processed in the context, and a set of exit data edges $C_i.V_{OUT}$, that represents data produced after context execution.

3 MULTI-CONTEXT CIRCUIT MODELS

The Multi-Context Circuit model is based on the Chameleon prototyping platform [5], composes by a software component, microcontroller Intel 8051, hardware component, FPGA XC4010 from Xilinx, and memory banks.

3.1 Context Circuits

Each context C_i is converted in a control-datapath circuit.

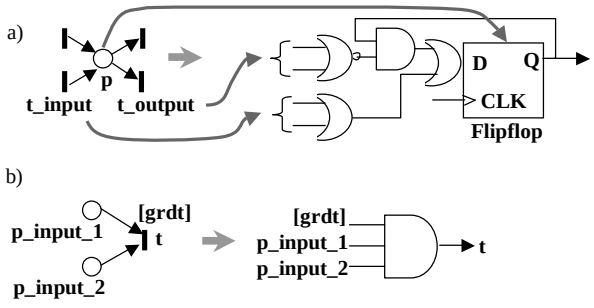


Figure 3: Hardware model for controller: a) PN^C place circuit; b) PN^C transition circuit.

In Figure 3, the place in control flow PN^C is mapping in D flipflop in a control unit. Transitions are mapping in AND gates, were inputs are place token value (0,1). Making the mapping of each transition $t \in C_i.T$ and each places of the control flow graph, a control unit for context C_i is generated. The datapath circuit is generates by mapping the datafllow graph G^D in RTL (Register Transfer Level) structure. Variables are mapping in registers, operations are mapping in functional units and the data dependency, represented by edges in G^D, is mapping in nets and multiplexers[6] [8]. The sets $C_i.V_{IN}$, $C_i.V_{OUT}$, $C_i.P_{IN}$, $C_i.P_{OUT}$ (Figure 2) represent the data value and control state communications between the contexts. The communication circuit model for control state and data values interchange between two contexts is depicted in Figure 4. In this circuit, the place p is common to both contexts C_i and C_j. and is implemented in the both context controllers. The transfer is performed by the microcontroller in the prototyping platform. The data value and control states read from C_i are stored in memory buffer in order to be sent to the next context C_j, after FPGA reconfiguration.

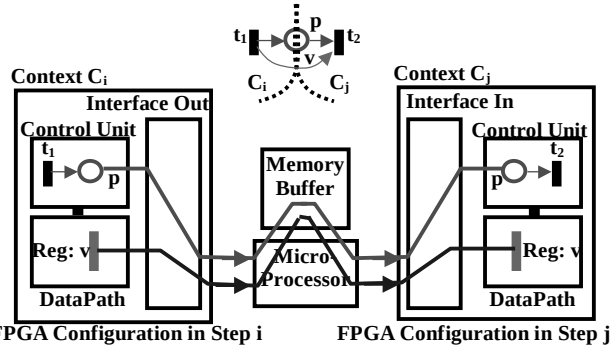


Figure 4: Output and Input context circuit interface.

Finally, the Figure 5 shows the circuit to detect the end of the context execution. For each initial marking of the input places in $C_i.P_{IN}$, a set of possible final markings of output places $C_i.P_{OUT}$ exists. These markings are only

reached in the end of the context execution and when a context finishes a marking of this set is always reached.

In each context, the initial input places marking are storage in special flipflops called *IMHF* (*Initial Marking Hold FlipFlop*s in Figure 5) that holds this marking during all context execution time. Therefore, it is always possible to detect the end of the context execution by observing if the initial $C_i.P_{IN}$ marking, in *IMHF* combined with the marking in $C_i.P_{OUT}$ reaches the end context marking.

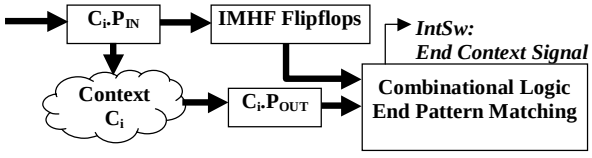


Figure 5: End Context Detect Circuit.

These patterns marking are generated during the partitioning of the application in contexts.

3.2 Switching Context Mechanism

The FPGA is used to implement the contexts according to a previous context scheduling and switching task managed by the microcontroller. The scheduling information is generated during the temporal partitioning. The context switching routine in software is activated by an *IntSw* interrupt signal generated by the FPGA (Figure 5). The data and state generated for the old context are transferred to a memory buffer and, after the FPGA reconfiguration, the microcontroller transfers the data information from the buffer to the new context. The context switching time T_{SW} is given by the FPGA reconfiguration time (T_{Rec}) and the data transference time between contexts (T_{Data}), $T_{SW} = T_{Rec} + T_{Data}$.

In order to improve the performance of the switching contexts on a static reconfigurable FPGA, a two stages pipeline structure is proposed in Figure 6. While a FPGA executes a context C_i another FPGA is reconfigured with next context C_{i+1}.

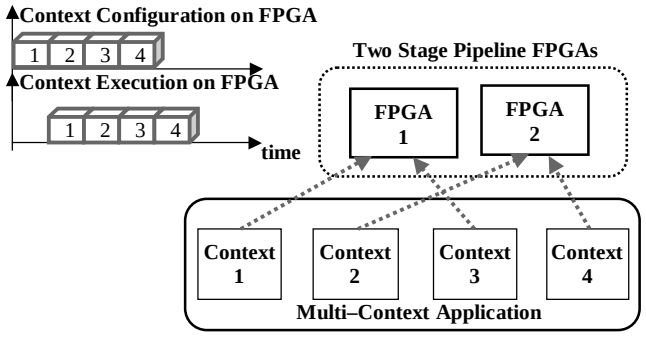


Figure 6: Two stages pipeline reconfigurable architecture.

If C_1, C_2, ..., C_N are the contexts and if T_{Ci} is the execution time of context C_i, T_{Di} is the time for data transfer in switching context C_i to next C_{i+1} and T_{Rec} is the FPGA reconfiguration time. Thus, the execution time T_{MC} of the multi-context application can be given by:

$$T_{MC}=T_{Rec}+T_{D0}+\{\sum_{i=1}^{N-1}[Max(T_{Rec},T_{Ci})+T_{Di}]\}+T_{CN}+T_{DN} \quad (1),$$

Where *Max(a,b)* function is the major value of a and b. T_{D0} is the time for input data in the first context C_1 and T_{DN} is the time for output data in last context C_N. For massive computation applications, it is possible that $T_{Ci} \geq T_{Rec}$ for all the contexts. The formula *(1)* is reduced to:

$$T_{MC}=T_{Rec}+T_{D0}+\{\sum_{i=1}^{N-1}(T_{Ci}+T_{Di})\}+T_{CN}+T_{DN} \quad (2)$$

In this in case, execution time depends only on the sub-process time in each context and the speed of data transfer, eliminating the reconfiguration latency.

4 TEMPORAL PARTITIONING ALGORITHM

The temporal partitioning split the *CDFG-Petri net* of hardware process P^{HW}, generating a contexts set $PartT^{HW}$. The total area of each context can not exceed the FPGA area and transitions must be grouped in the contexts obeying the order in the control flow PN^C:

Definition Temporal Partitioning: *Let a pair* $P^{HW} = (PN^C, G^D)$ *be a CDFG process implemented in an FPGA area* AH_{FPGA}*, we defined a* P^{HW} *partitioning as being a contexts set* $PartT^{HW} = \{C_1, C_2,\ldots, C_K\}$ *and* $C_i.T \cap C_j.T = \{\}\ \forall C_i,C_j \in PartT^{HW}$, $i \neq j$. $C_i.T$ *is the context transitions set;* $C_i.A_{HW}$ *is the FPGA area for* C_i *and context execution order is* $C_1 \rightarrow C_2 \rightarrow \ldots C_k$. ***Constraints*** *is* ***Area:*** $C_i.A_{HW} \leq AH_{FPGA}$ $\forall C_i \in PartT^{HW}$ *and* ***Precedence:*** $t \in C_i.T \Rightarrow (\forall t' \in I(p) \wedge p \in I(t): \exists C_j \mid t' \in C_j.T \wedge j \leq i)$.

The partitioning uses a constructive algorithm, showed in Figure 7, based on the transitions grouping into the contexts.

```
TempPart{
    Input (P^HW=(PN^C,G^D), AH_FPGA )
    PartT^HW := {} ; CandT :={t0}
    IndxCtx := 0;
    While CandT ≠ {} do {
        Possible := True; IndxCtx :=IndxCtx+1;
        C_IndxCtx := Criate_Ctx(IndxCtx);
        While Possible do {
            (Possible,t) = Choice_Trans(CandT,C_IndxCtx);
            If Possible then {
                UpDate_Ctx(C_IndxCTX,t);
                UpDate_CandT(CandT,t);
            };
        };
        PartT^HW := PartT^HW ∪ {C_IndxCtx};
    };
    Output(PartT^HW);
}
```

Figure 7: Temporal partitioning algorithm.

The algorithm inputs are *CDFG-Petri net* and area constraint AH_{FPGA}. The output algorithm is a contexts set $PartT^{HW}$. The *CandT* set represents the transitions candidates to be grouped in $C_{IndxCtx}$ Context, without violate the precedence constraint. The choice of the transition, that must be grouped in a context, is made by *Choice_Trans* function that returns a transition $t \in CandT$ that can be grouped in $C_{IndxCtx}$ without violate the area constraint. When the context $C_{IndxCtx}$ cannot group more transitions without violating the *area* constraints, the *Choice_Trans* function attributes the False value to *Possible* variable and a new context is created by *Criate_Ctx* function. It is necessary to apply an criterion to choose the most appropriate transition to be placed in context $C_{IndxCtx}$. The criterion More *concurrency* implies the choice where the transitions are searched in width, considering parallelism exploration and execution time minimization. The criterion *minimizes the communication* implies the choice of the transition that results in less data edges, in data dependence graph G^D, crossing the contexts.

4.1 Area Estimation

In Figure 3, 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:

$AH_{pl} = AH_{FFD} + AH_{AND2}+(m+n-1)AH_{OR2}$ *(3)*

$AH_t = (n+g-1)AH_{AND2}$, *g = number of [grdt]* *(4)*

AH_{FFD} is the area of the Flip-Flop D, AH_{AND2} and AH_{OR2} are the areas of the AND and OR gates of 2 input. The datapath circuit consists of registers, multiplexers and functional unit components with areas designated by: AH_{R1}, AH_{MUX2}, AH_{FUi}. The area estimate is done based on equivalent gates number for the XC400XX FPGAs [8], considering control and datapath areas:

$AH_P = AH_{CTRL} + AH_{DP}$ *(5)*

The control area estimation are obtained directly from expressions *(3)* and *(4)*:

$AH_{CTRL} = \sum_{\forall p \in P^c} AH_{pl}(p) + \sum_{\forall t \in T^c} AH_t(t)$ *(6)*

For datapath 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 variables set (*VP*) used in the process. If $|v|$ is the length in bits of $v \in VP$, the registers area used is:

$AH_{reg} = \sum_{\forall v \in VP} |v| AH_{R1}$ *(7)*

Multiplexers area AH_{mux} is estimated as a function of the number of incidences for each operation in the 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, the area estimated of multiplexers is given by:

$AH_{mux} = \sum_{\forall op \in OP} N(op)|op| AH_{MUX2}$ *(8)*

The expressions *(7)* and *(8)* do not take into account registers and functional units reutilization. In this way, 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:

$AH_{fu} = \sum_{\forall op \in OP} N(FU_{op})\, AH_{FUop}$ *(9)*

Datapath area is the addition of *(7)*, *(8)* and *(9)*:

$AH_{DP} = AH_{reg} + AH_{mux} + AH_{fu}$ *(10)*

The number of functional units $N(FU_{op})$, in the expression *(9)*, is based on p-minimus invariants supports and universal transitions path set calculation for PN^C [6] [7] [8]. Path transition may represent transitions that has exclusion relation in the PN^C. Transition path set covering techniques can be used to determine the transitions concurrency degree inside of a contexts. This metrics is necessary for functional unit reuse estimation and concurrency criteria during the temporal partitioning.

The additional area is necessary for implementation of circuits *Interface_In* and *Interface_Out*, depicted in Figure 4, and *End Context Detect Circuit*, showed in Figure 5:

$AH_{CSW} = AH_{InterfaceIn/Out} + AH_{EndCtx}$ *(11)*

If Nb_{IN} and Nb_{OUT} are numbers of input and output bits in context C_i, givens by:

$Nb_{IN} = \#\, C_i.P_{IN} + \sum_{\forall v \in Ci.V_{IN}} |v|$ *(12)*

$Nb_{OUT} = \#\, C_i.P_{OUT} + \sum_{\forall v \in Ci.V_{OUT}} |v|$ *(13)*

If *SizeB* is the size of data bus interface between microprocessor and FPGAs, in bits, then the communication cost of context C_i is given by:

$C_i.CC = (Nb_{IN} + Nb_{OUT})/SizeB$ *(14)*

The area for implementation the circuits interfaces in context C_i is given by:

$AH_{InterfaceIn/Out} = log_2(SizeB^{-1} * max\,(Nb_{IN}, Nb_{OUT}) * [c_1 + c_2 * max\,(Nb_{IN}, Nb_{OUT})] + c_3 * Nb_{OUT} + c_4 * \#C_i.P_{IN} + c_5 * (\sum_{\forall v \in Ci.V_{IN}} |v|)$ *(15)*,

where c_1, c_2, c_3, c_4 and c_5 are constants dependents of the details of interface implementation into the FPGAs.

Finally, AH_{EndCtx}, the *End Context Detect Circuit* area, is given by:

$AH_{EndCtx} = c_6 * M * N_{Marking} + c_7 * M + c_8 * N_{Marking}$ *(16)*

where c_6, c_7 and c_8 are constants, $M = \#C_i.P_{IN} + \#C_i.P_{OUT}$ and $N_{Marking}$ is the number of patterns marking of the initial $C_i.P_{IN}$ marking combined with the marking in $C_i.P_{OUT}$ that indicates the end of context execution.

The total area for context C_i is obtained by application of formulas *(3)* to *(16)* and is given by:

$C_i.A_{HW} = AH_P + AH_{CSW}$ *(17)*

4.2 Time Execution Estimation

The time execution T_{Ci}, for context C_i, is determined in recursive way. For each transition *t*, we associate two time values called *MST*, maximum start time, and *MET*, maximum end time, for execution operation:

$$MST(t) = \begin{cases} 0, \text{ if } t \text{ is start transition in context } C_i \\ Max\{MET(t') | (t' \in I(p) \wedge p \in I(t))\} \end{cases} \quad (18)$$

$$MET(t)= \begin{cases} D(OP_t), \text{ if } t \text{ is start transition in } C_i \\ D(OP_t) + MST(t), \text{ otherwise} \end{cases} \quad (19),$$

Where, $D(OP_t)$ is the execution delay of operation OP_t associated the *t*.

For a context C_i, we can define the time context execution T_{Ci} as being the maximum *MET* value for the last transitions set in context C_i:

$$T_{Ci} = Max\{MET(t) \mid t \text{ is last transition in } C_i\} \quad (20)$$

5 CASE STUDY

In This section a differential equation resolution is performed by the multi-context model presented in this work. The equations are as following:

$x(t + i) = x(t) + y(t).i$

$y(t + i) = y(t) - f(x).y(t).t.i-3.x(t).i$

$t' = t + i$, *new t value*

The differential equation *CDFG-Petri net* resulted in 40 places and 39 transitions. The Petri net was then split into 4 sub-nets of contexts, according to the two criteria. Table 1 shows the partitioning results.

Table 1 – Temporal partitioning results

Partitioning 1						**Partitioning 2**			
More Concurrency						***Minimizes Communication.***			
Ctx	**AE**	**AR**	**P**	**T**	**C**	**Ctx**	**AE**	**T**	**C**
1	4987	4218	84,58	4	13	1	4768	8	11
2	4377	3688	84,26	5	15	2	4512	5	17
3	4995	4678	93,65	6	19	3	2589	5	17
4	1053	855	81,19	2	9				
Σ	15412	13439	87,20	17	56	**Σ**	11869	18	45

Ctx=Context, **AE**=Estimated Area (equivalent gates), **AR**=Real Area (equivalent gates) **T**=Time(cycles) and **C** = Communication (bytes). FPGA area is $\boldsymbol{AH_{FPGA}}$ = 5000 equivalent gates. **P**=Precision= (**AR**/**AE**).100%.

The equivalent gates areas **AE** and **AR** in Table 1 were estimated based on the XC4000 Xilinx FPGA family [5]. The context execution time **T** is estimated on the basis of the critical transition path extracted from formulas *(18), (19)* and *(20)*. For the first contexts in both partitioning approach (Table 1), we can observe that in the *concurrency* criterion temporal partitioning takes less execution time than the *communication* criterion. It is due to the parallelism exploitation (**Partitioning1**) and communication reduction (**Partitioning2**). The partitioning 1 has been totally implemented in the chameleon board. The partitioning result had been translated manually into a RTL-level hardware description for each context and the Xilinx Foundation 3.1i. Synthesis areas results are showed in the column **AR** of Table 1. Good results have been obtained for area estimates with precisions around 86% in relation to Xilinx Syntheses Tool. In this version, based on a board with a single FPGA, the time for FPGA reconfiguration takes 16 ms. Depending on the number of contexts and application constrains the FPGA reconfiguration time may be suitable. However, for application in massive computational problems, the two stages pipeline model can by useful for removes reconfiguration latency and speed up the application.

6 CONCLUSION AND FUTURE WORKS

A switching context architecture based on Chameleon Platform, CDFG-Petri Net model and temporal partitioning algorithm are presented. In this architecture, large applications may be split into suitable hardware and software processes and run in a switching context approach. Estimates from a Petri net model also guarantees that hardware processes areas are suitable for the area available in the prototyping platform. An example has been presented in order to demonstrate the method for a low-end speed application. A two stages pipeline approach for hardware reconfiguration speed improvement during context switching has been suggested. It is possible to reduce the latency time between switching contexts since the FPGAs reconfiguration can overlap context execution. This pipeline strategy can also be internally used in dynamically reconfigurable FPGAs. A new platform with such architecture is under development.

7 REFERENCES

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

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

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

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

[5] 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.

[6] Nascimento, Paulo S. B.; Lima, Manoel; Maciel, Paulo; Silva-Filho, A. G.; Barros, Edna; Cavalcante, Sérgio, "CDFG – Petri Net Temporal Partitioning for Switching Context Applications", 15th Symposium on Integrated Circuits and Systems Design – SBCCI 2002, Proceedings, pp. 235-240.

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

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