Resource Sharing Estimation by Petri Nets in PISH Hardware/Software Co-design System

Paulo Maciel, Edna Barros, Mauro Silva and Fred Cruz Filho
Centro de Informática - Universidade Federal de Pernambuco
Recife, Brazil, 50740-540

Abstract

This work presents two approaches for computing the number of functional units in hardware/software codesign context. The proposed hardware/software codesign framework uses Petri net as common formalism for performing quantitative and qualitative analysis. The use of Petri net as an intermediate format allows to analyze properties of the specification and formally compute performance indices which are used in the partitioning process. This paper is devoted to describe the algorithms for functional unit estimation.

This work also proposes a method of extending the Petri net model in order to take into account causal constraints provided by the designers. However, an overview of the general hardware/software codesign method is also presented.

1 Introduction

Hardware/Software codesign is the design of systems comprising two kinds of components: specific application components and general programmable ones. Although such systems have been designed ever since hardware and software first came into being, there is a lack of CAD tools to support the development of such heterogeneous systems. The progress obtained by the CAD tools at the level of algorithm synthesis, the advance in some key enabling technologies, the increasing diversity and complexity of applications employing embedded systems, and the need for decreasing the costs of designing and testing such systems all make techniques for supporting hardware/software codesign an important research topic.

The choice of the components set (definition of a target architecture) and the partitioning of the description are critical tasks in a codesign system. This work considers Petri as an intermediate model that allows both qualitative analysis and metrics computation [5]. In the quantitative analysis phase (metrics computation), methods are applied for computing precedence relation degree, load balance [5], communication cost [4], computing cycle time [5], area estimates [8] as well as functional unit estimation. Those metrics guide the partitioning process.

This paper presents two algorithms based on Petri nets for estimating the number of hardware functional units needed to carry out a behavioral specification. The first one is based on reachability graph approach and the second one considers the use of invariants. Due to the large number of possible invariants, an approximation algorithm for implementing the structural method has been proposed. Its results are compared with those results obtained by the exact solution of the structural methodology as well as with those by the reachability based method.

Moreover, this work describes a methodology for including causal constraints on the Petri net model.

The next section presents an overview of the hardware/software partitioning approach. Section 3 introduces timed Petri net. Section 4 describes how to introduce causal constraints in the behavioral specification. Section 5 depicts the adopted hardware model and the methods proposed for estimating resource sharing. A case study is described in Section 6. Finally some conclusions and perspectives for future works are presented.

2 The Hardware/Software Partitioning Approach

The system uses occam as specification language. The main reason for using occam is its simple and elegant semantics, given in terms of algebraic laws. This feature allows the hardware/software partitioning to be carried out by applying a serie of algebraic transformations to the initial description preserving its original semantics. The set of transformation rules is applied according to the result of a clustering phase. The

0-7803-6583-6/00/$10.00 © 2000 IEEE

clustering phase groups processes taking into account the metric estimates (quantitative analysis) [5]. This analysis is performed considering a Petri net model of the occam specification [6] and consider criteria like communication cost [4], load balance, precedence relation degree [5] and area estimates [7]. After hardware/software partitioning, the processes to be implemented in hardware are synthesized and the software processes are compiled.

3 Timed Petri Nets

Petri nets are a formal specification technique that allow for a graphical, mathematical representation and have powerful methods which allow designers to perform qualitative and quantitative analysis [2].

Petri nets are used to model a logical point of view of the systems, however no formal attention is given to temporal relations and constraints [12]. The first temporal approach was proposed by Ramchandani [3].

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.

Definition 3.0.1 Timed Petri Nets - *Let* $Nt = (N, D, C)$ *be a timed Petri net, where* $N = (P, T, I, O, M_0)$ *is a Petri net,* $D : T \rightarrow IR^+ \bigcup 0$ *is a function which associates to each transition* t_i *the duration of the firing* d_i. $C : T \rightarrow c$ $(0 \leq IR \leq 1)$, $t \in T$ *is a choice function which assigns a free-choice probability to each transition of the net, where* $\sum_{t \in T_c} c(t) = 1$. $T_c \subseteq T$ *is a set of structural conflicting transitions.*

4 Causal Constraints

Besides the behavioral specification, sets of causal constraints may be useful when analysing the design space of a system [1]. These external assumptions allows one to analyse distinct implementation possibilities without re-writing the main specification in order to find an alternative which satisfies the non-functional requirements

In this work, external causal constraints are introduced into the Timed Petri net model. The causal constraints considered are precedence relation and mutual exclusion.

4.1 Causal Precedence Constraint

This constraint defines a causal precedence execution order of operations within concurrent processes. If two operations represented by two transitions op_i and op_j should be carried out taking into account a precedence order, a place p_k should be introduced in the Petri net model such that $p_k \in O(op_i), I(op_j)$.

Definition 4.1.2 Causal Precedence Constraint - *Let* $Nt = (N, D, C)$ *be a timed Petri net,* op_i, $op_j \in T$ *transitions representing two operations and* $p_k \in P$. *If* $p_k \in O(op_i), I(op_j)$, *the operations* op_i *and* op_j *have a causal precedence relation* $(op_i \succ op_j)$.

4.2 Mutual Exclusion Constraint

This constraint is related to exclusive operation execution of concurrent processes. If two concurrent operations, represented by two transitions op_i and op_j, should be mutually exclusively executed, in the timed Petri net model, a safe marked place p_k, as input and output of these operation, may be introduced in order to exclude possible concurrent execution.

Definition 4.2.3 Mutually Exclusive Constraint - *Let* $Nt = (N, D, C)$ *be a timed Petri net,* op_i, $op_j \in T$ *transitions representing two concurrent operations and* $p_k \in P$ *a safe place. If* $p_k \in O(op_i)$, $I(op_i)$, $O(op_j)$, $I(op_j)$; $M_0(p_k) = 1$, $p_k \in P$ *and* p_k *is defined as a mutual exclusion relation* $(op_i \otimes op_j)$.

5 Estimation of Functional Unit Numbers

The hardware implementation of a process can be composed of: data-paths and controllers [5]. The data-path circuit consists of registers, functional units and multiplexers. The functional-unit area of a process is related to the area of ALUs, adders, multipliers etc needed to carry out arithmetic/logical operations.

Definition 5.0.4 Functional Unit Area - *Let* $FUN(NS, OP\_TYPE)$ *be functional unit number of the type* OP_TYPE *related to a set of processes* NS. *Let* $OPS(NS)$ *be the set of distinct operations types in the process* NS. *Let* op_j *be an operation within an arithmetic/logic expression* e.
$AH_{op}(NS) = \sum_{\forall OP\_TYPE \in OPS(NS)} FUN(NS, OP\_TYPE) \times AH_{OP\_TYPE}$ *gives functional unit area of a set of processes, where* AH_{OP_TYPE} *is the*

area associated to the operator that implements an operation of the type OP_TYPE, considering a given data type (number of bits).

Therefore, one aspect that should be considered is the estimation of the necessary number of functional units to execute a given behavioral description. This aspect has to be taken into account possible functional unit sharing. Two approaches have been proposed for functional unit estimation: a method based on reachability graph (dependent on the architecture) and a structural approach.

5.1 Reachability Based Method

First, let us consider a model extension in order to capture the number of functional units of the proposed architecture. The extended model is represented by the net $N = (P, T, I, O, M_0, D, C)$, which describes the program, a set of places P' in which each of its places (p') is a functional unit type adopted in the proposed architecture; the marking of each of these places represents the number of functional units of the type p'; the input and the output arcs that interconnect the places of the set P' to the transitions which represents the arithmetic/logic operations ($ALU_{op} \subset T$).

In the extended model the number of conflicts in the net increases due to the allocation of operations to functional units. These conflicts require the use of a pre-selection policy. Such a policy is implemented by assigning equal probabilities to the output arcs from places (representing functional units types) to the enabled transitions $t_j \in ALU_{op}$ ($O(p, t_j)$, $p \in P'$) in each reachable marking M_z. Thus, more formally:

Definition 5.1.5 -Extended Model : *Let a net $N = (P, T, I, O, M_0, D, C)$ a program model, a set of places P' the functional unit types adopted in the architecture such that $P \bigcap P' = \emptyset$ and $M_0(p)$, $p \in P'$ the number of functional units of the type p. Let a net $N_e = (P_e, T_e, I_e, O_e, M_0^e, D_e, f)$ the extended model such that $P_e = P \bigcup P'$, $T_e = T$, $I_e(p, t_j) = 1$ and $O_e(p, t_j) = 1$, $\forall t_j \in ALU_{op}$, $\forall p \in P'$, otherwise $I_e(p, t_j) = I(p, t_j)$ and $O_e(p, t_j) = O(p, t_j)$. $M_0^e : IN^{P \bigcup P'} \rightarrow IN$ and $D_e = D$. Let M_z a reachable marking from M_0, $C : T \rightarrow c$ $(0 \leq IR \leq 1)$, $t \in T$ is a choice function which assigns a free-choice probability to each transition of the net, where $\sum_{t \in T_c} c(t) = 1$. $T_c \subseteq T$ is a set of structural conflicting transitions.*

In such a model, the concurrence is constrained by the number of available number of functional units ($M_0(p)$, $p \in P'$) provided by the designer. The main goal of the proposed approach is to estimate the minimal number of functional units that can achieve best performance taking into account an upper bound number. Therefore, the designer, provides the number of available units (adders, ALUs etc), then the execution time (CT) is computed by reachability based methods. The following step comprises the reduction in the number of functional units ($M(p) = M(p) - 1$, $p \in P'$) in order to compute a new execution time (CT'). If $CT' > CT$, the necessary number of unit has been reached. This number of functional units is used in the proposed method for initial allocation.

The proposed algorithm is:

- *Input*:

  a net $N_e = (P_e, T_e, I_e, O_e, M_0^e, D_e, C)$.

  the number of available functional units of a given type ($M_0(p)$, $\forall p \in P'$).

- *Output*:

  the optimum number of units $M_{opt}(p)$, $p \in P'$ taking into account the resources constraints provided by the designer.

  the minimal execution time (CT) regarding to $M_{opt}(p)$.

- **Algorithm**:

Compute the execution time $CT(N_e)$
$CT = CT(N_e)$
For each place $p \in P'$, do:
$M(p) = M(p) - 1$
Compute a new execution time $CT(N_e)$
if $CT(N_e) \leq CT$
$CT = CT(N_e)$
$M_{opt}(p) = M(p)$, $\forall p \in P'$
else or if $M(p) = 0$, $\forall p \in P'$
end.

The number of necessary units can also be reached by taking into account either the speed up, the efficiency or the efficacy provided by the use of multiple processors [5].

Although the interesting results have been obtained, one should observe that the operation are allocated to certain types of functional units. If the system level is being considered, it may be too early to perform this activity.

5.2 Structural Based Method

The method presented in this section provides an upper bound number of functional units needed for carrying out each operation type. This method takes into account precedence/mutual exclusion relation between operators of the same type within a process.

In order to estimate this bound, the behavioral description is analyzed in terms of a causal precedence

relation of operators in such a specification, that is, either operators which are combined to be executed in sequential fashion, the ones that have a causal relation because of communication, or the ones that are in mutual exclusion due to shared variables or semaphore implementation. Shared variable and semaphore implementation is out of the scope since occam has been adopted as a specification language and the communication is represented by synchronous actions. Nevertheless, it is important to stress that the method proposed is able consider mutually exclusive statements represented by these implementations.

Considering a Petri net model obtained from an occam description according a translation method presented in [5]. However, a closure[1] is applied to this net in order to turn it strongly connected. The causal precedence relation can be analyzed by means of p-minimum invariants (IP_i) [5]. The first step of the method proposed in order to compute the number of functional units needed to carry out the description is the calculation of invariant supports. After that, the transition-paths[2] have to be computed. The estimated number of functional unit (of a given type) needed to execute a description is the minimal number of transition-paths that covers a transition set (representing operations) of a given type.

Taking into account that a set of processes represented by NS and a transition set of a given type $TS(NS, OP\_TYPE)$, if the transitions of such a set are covered by one transition-path, generated by one p-minimum invariant, their execution can be carried out by only one functional unit. In order to carry out each statement of a given type considering a specific set of processes ($TS(NS, OP\_TYPE)$), an upper bound number of functional units is be provided by the minimal number of p-minimum invariants [11] needed to generate the smallest transition-paths set (STPS) that contains each transition of $TS(NS, OP\_TYPE)$.

Theorem 5.2.1 *Let $N = (P, T, I, O, M_O, D)$ be a strongly connected net covered by p-minimum invariants. Let TPS_i be a transition-path set of a sub-net SN_i obtained from a p-minimum invariant IP_i of a net N. Let $TS(NS, op_l) \subseteq T$ be the transition set of a given operation type related to a processes set NS. An upper bound functional unit number of the type op_l related to NS is $FUN(NS, op_l) = \#STPS$, where $\#STPS$ denotes the minimal number of transition-path set TPS_i that contains each transition of $TS(NS, op_l)$.*

The proof of such theorem can be found in [7].

The exact solution of that problem (set-covering problem) may be computationally very complex, hence an approximation algorithm has been proposed to compute $FUN(NS, op_l) = \#STPS$. The method proposed is based on clustering technique and results in $O(n^3)$ complexity, where n is the number of transition-paths.

- *Input*:
  set $UTPS$,
  set $TS(NS, op_l)$.
- *Output*:
  $FUN(NS, op_l) = \#STPS$.
- **Algorithm**:

$Z = UTPS$, $STPS = \emptyset$, $M = \emptyset$ - dynamic sets -,
$STPS = ComputeCoverability(UTPS, TS(NS, op_l))$
If $STPS \neq \emptyset$
 $FUN(NS, op_l) = \#STPS$
 STOP.
WHILE $Z \neq \emptyset$
 $X, Y = FindTPSs(Z)$
 $Delete(Z; M)$
 $Insert(STPS; X, Y)$
 $Delete(Z; X, Y)$
 $M = \bigcup_{\forall TP_i \in STPS} TP_i$
 $Insert(Z; M)$
 $V = M \cap TS(NS, op_l)$
 If $\#V == \#TS(NS, op_l)$
  $FUN(NS, op_l) = \#STPS$
  STOP.
$c, a = 0$
$FindTPSs(Z)$; $(\#Z = N)$
 For i=0 to N-1, do:
  For j=i+1 to N, do:
   $a = ComputeCloseness$
    $Metric(TPS_i, TPS_j)$
   If $a > c$
    $c = a, K = TPS_i,$ $L = TPS_j$
 For i=0 to N-1, do:
  For j=i+1 to N, do:
   $a = ComputeCloseness$
    $Metric(TPS_i, TPS_j)$
   If $a == c \wedge K \neq TPS_i \wedge$
    $L \neq TPS_j$
    $E = K, F = L,$
     $G = TPS_i,$
     $H = TPS_j$
    $K, L = FindLeast$ $SimilarTPSs($ $E, F, G, H)$
   If $a == c \wedge K \neq TPS_i \wedge$
    $L == TPS_j$
    $E = K, F = L,$
     $G = TPS_i, H = L$
    $K, L = FindLeast$

[1] a transition connecting the final place to the start place. A start place is is the initial condition of a program. When a final place is marked, it means the program completion.

[2] a transition-path is the set of output transitions of the support of a p-minimum invariant.

```
    SimilarTPSs(
    E, F, G, H)
  If a == c ∧ K == TPS_i ∧
   L ≠ TPS_j
   E = K, F = L, G = K,
    H = TPS_j
   K, L = FindLeast
    SimilarTPSs(
    E, F, G, H)
Return(K,L)
FindLeastSimilarTPSs(E, F, G, H)
 If #(E ∩ F) ¡ #(G ∩ H)
  Return (E, F)
 Else
  Return (G, H)
ComputeClosenessMetric(TPS_i, TPS_j)
 V = TPS_i ∪ TPS_j ∩ TS(NS, op_l)
 Return(#V)
ComputeCoverability(UTPS, TS)
 C = ∅
 While i ≤ N − 1 ∨ C ≠ ∅, do:
  V = TPS_i ∩ TS(NS, op_l)
  If #V == #TS(NS, op_l)
   C = TPS_i
  i = i + 1
 Return(C)
```

3

The method proposed , however, only provides an upper bound. This approach does not deal with temporal precedence relation between statements, that is, statements that are neither under a causal precedence relationship or in exclusive choice, but have a temporal precedence relation.

5.3 Experiments

In this section, the approximation algorithm presented in Section 5.2 (structural based method) is applied to a set of small examples. These results are compared to the solutions provided by exact-solution algorithm (structural based method). The reachability graph based algorithm proposed in Section 5.1 is also applied to these examples. In the following the occam description of the examples is described.

- Example **A**

```
INT a,b,c,d:
PAR
 SEQ
 a:=a+1
 PAR
   b:=b+1
   c:=c+2
 IF d < 0
    d:=d+1
  d >= 0
    d:=d+2
```

Example **B**

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

- Example **C**

```
CHAN OF INT ch1,ch2:
PAR
 INT a,b:
 SEQ
   a:=a+1
   ch1 ? b
   a:=a+b
 INT c:
 SEQ
   IF c < 0
      c:=c+1
    c >= 0
      c:=c+2
    ch1 ! c
    ch2 ! c
 INT d:
 SEQ
   ch2 ? d
   d:=d+3
```

Example **D**

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

For theses examples, Table 1 presents the functional unit numbers estimated by considering the approximation algorithm presented (column $FUN_{ap.\_st}$) in Section 5.2 as well as the exact solutions when considering such an approach (column FUN_{e_st}). The obtained results are the same.

Table 1: Number of Functional Units

Example	FUN_{e_st}	$FUN_{ap.\_st}$	FUN_r
A	3	3	2
B	2	2	2
C	2	2	2
D	4	4	2

It should be highlighted that for many other small examples, which I was able to obtain the exact solutions, the approximation algorithm proposed provided equivalent results.

Table 1 also presents the obtained results for the function units estimation (column FUN_r) based on the reachability approach (see Section 5.1). One may observe that the functional unit numbers are smaller than those estimated by the structural approach, but should also remember that such an approach needs allocation process.

[3] Functions $Insert(A;e)$ and $Delete(A;e)$ inserts and removes an element e to/from a dynamic set A, respectively [10]

Table 2: Area

Metrics	Estimated	Implemented
Reg+Mux	464	650
FU	4291	4291
Control	1026	1230
Total	5781	6171

6 Results

This section presents some results for a vending machine specification. The occam specification is composed of seven (7) processes. Excution time, load balance, mutual exclusion degree, communication cost, area metrics were compute. Considering these metrics, the hardware/software partitioner generated two clusters. The first one is a software cluster and the second one a hardware cluster.

The hardware cluster was mapped onto 2 FPGAs from XC4000 Xilinx family. Table 2 shows the metrics estimated and its real implementation. In this table, the close results reached by the methodology are compared with the real implementation, in terms of count gates. For this example, the average accuracy is 94% (for all these area metrics).

The values of the implementation are given by the rapid prototyping Xilinx Foundation tool in terms of gates counts.

7 Conclusion

This work compares two method for functional unit estimation. The reachability approach provides interesting results in terms of accuracy. However, an allocation in a early phase of design is needed. The structural method does not allocate operation to functional units, it considers causal precedence relation and mutual exclusion, but it does not take into account temporal precedence relation. This method provides an minimum upper bound number of functional units. For this approach, an approximation algorithm has been presented and its results compared with the exact structural solution and with those obtained by the reachability based method. The results obtained for the case study presented were similar when using both the reachability based and the structural based methods.

As future works, we intend to use Petri net to consider schedulling and power consumption estimation in the PISH codesign methodology.

References

[1] A. Mazzeo, N. Mazzocca, S. Russo, C. Savy, V. Vittorini *Formal Specification of Concurrent Systems: a Structured Approach* The Computer Journal, Vol 41, n. 3. 1998.

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

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

[4] P. Maciel, E. Barros, W. Rosenstiel. *Computing Communication Cost by Petri Nets for Hardware/Software Codesign.* 8th IEEE International Workshop on Rapid System Prototyping, Chapel Hill, North Carolina, USA, June 24-26, 1997.

[5] P. Maciel, E. Barros, W. Rosenstiel. *A Petri Net Model for Hardware/Software Codesign.* Design Automation for Embedded Systems Journal, Kluwer Academic Publishers, $n^0$4, Vol 4, October, 1999.

[6] P. Maciel, E. Barros, W. Rosenstiel. *A Petri Net Based Approach for Performing the Initial Allocati on in Hardware/Software Codesign.* 1998 IEEE International Conference on Systems, Man, and Cybernetics, San Diego, California, USA, October 11-14, 1998.

[7] P. Maciel, E. Barros, W. Rosenstiel. *A Petri Net Approach for Estimating Hardware Area Considering Clock Period.* IEE 15^{th} International Conference on CAD/CAM Robotics & Factories of the Future, Águas de Lindóia, Brazil. August, 1999.

[8] P. Maciel, E. Barros, M. Lima, D. Silva, W. Rosenstiel. *Resource Sharing Estimation by Petri Nets in PISH Codesign System.* High Performance Computing 2000, Washington DC, USA, 15-20 April, 2000.

[9] Xilinx, 1997. *Gate Count Capacity Metrics for FPGAs.* XAPP 059 Feb. 1, 1997, V1.1.

[10] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms.* McGraw Hill, 1998

[11] J.M.Colom,M.Silva. *Convex Geometry and Semiflows in P/T Nets. A Comparative Study of Algorithms for Computation of Minimal P-Semiflows.* Lecture Notes in Computer Science, vol-483, p. 79-112, Springer-Verlag, Edited by G. Rozenberg 1990.

[12] W.M. Zuberek. *Timed Petri Nets Definitions, Properties and Applications.* Microelectronic and Reliability, vol. 31, no. 4, pp 627-644, 1991.