International Journal of Technology (2016) 6: 1054-1067
ISSN 2086-9614

PETRI NET-BASED SYSTEM MODELLING FOR PERFORMANCE ANALYSIS OF RECONFIGURABLE SOFT-CORE PROCESSOR USING FPGA

Maheswari R.[1*], Pattabiraman V.[1]

[1]*School of Computing Science and Engineering, VIT University Chennai Campus, Chennai 600127, India*

(Received: September 2015 / Revised: April 2016 / Accepted: October 2016)

ABSTRACT

In any heterogeneous system, porting reconfigurable computing is often a high performance platform for a broad range of computationally challenging issues. However, efficiently utilizing the maximum potential of these reconfigurable systems is a difficult job without understanding their performance characteristics. This work proposes an analytic performance model using Petri Nets (PN) for a Reconfigurable OR1200 (ROR1200) soft-core processor with model validation and verifications. By modeling the ROR1200 system using Petri Nets, both behavioral and structural properties existing in parallel systems were analyzed. The Bound Level Analysis with respect to the dependency level of data is also performed on Soft Core Processors (SCP) like the ROR1200, the OR1200 and the MicroBlaze.

Keywords: Behavioral properties; Petri Net; ROR1200; Structural properties

1. INTRODUCTION

The analysis of the system is started by developing or creating the model of the system, which may help in further examination and potential findings of the behaviors of the system. Petri Net theory is one of the most promising tools used for the design and analysis of those system models (Marcin Radom et al., 2015).

1.1. Petri Net Performance Model

In any computing system, the performance of the system model can be analyzed using two approaches: first, the Analytical Model and second, the Simulation Model. Various prominent techniques used to measure the analytical models are: Markov chain models, Semi-Markov models, Queuing Network models, Petri Net models and Stochastic Process Algebra models. Markov chains are conventional analytical modeling used to express random processes. Due to their memory-less properties, Markovian chains are used to model those systems whose output behavior depends on their current state (John and Laurie, 1960). A queuing model is one of the most powerful techniques for modeling hardware based on contention and various scheduling strategies. A queuing model also holds for many efficient analysis techniques. One of the demerits or disadvantages of a queuing model is that, it is difficult to model blocking concepts, synchronization, mutual exclusion and software-based contention concepts. Stochastic process algebra (SPA) has emerged as a functional analysis modeling technique for concurrent systems. The major limitation of stochastic process algebra is its deficiency in expressing the system with respect to time distribution (Gilmore et al., 1996).

*Corresponding author's email: maheswari.r@vit.ac.in, Tel. +91-44-39931240, Fax. +91-44-39932555
Permalink/DOI: https://doi.org/10.14716/ijtech.v7i6.1959

Performance of the hardware systems like field programmable gate array (FPGA), application-specific integrated circuit (ASIC) are analyzed with respect to its respective qualitative and quantitative properties. For example, to analysis the fault tolerance of the hardware system, the feasibility of recovering the error is said to be a qualitative approach. The amount of time required to handle the error recovery routine is characterized as a quantitative property (Falko Bause et al., 2005). The Petri Net model is well suitable for both qualitative (time-less) and quantitative (time-dependent) analysis modeling (Monika et al., 1997). This model is supported in terms of modeling blocking concepts, synchronization, mutual exclusion and software-based contention concepts (Jennings et al., 2000). The Petri Net model is used to analyze the properties and issues coupled with parallel systems. Thus, the Petri Net is mainly used for modeling the dynamic concurrent actions of the systems. Moreover, the PN model helps to guarantee completeness of the design and permits improvisation in the correctness of the designed system. When compared to the state of the art, the proposed methodology provides novel contributions. These contributions use a reconfigurable framework for the real-time multimedia dataset that describes the different entities executing in the high performance reconfigurable computing (HPRC) and Reconfigurable Register File (RRF) stack to reduce temporal isolation. This reconfiguration technique increases the system performance and also the output quality experienced by the user without any temporal deviation (Wang et al., 2009). Analytical modeling is extensively useful in performance evaluation due to its superiority and flexibility (Kant, 1992). The performance of reconfigurable ROR1200 with HPRC system has been successfully modeled using Petri Nets (Lotfifar et al., 2008). This Petri Net methodology has evidenced itself to model the most important characteristics of modern computer systems exploring parallelism and concurrency (Gaubert et al., 1997). For reconfigurable computing, the Petri Net model and its associated analytical processes afford a promising modeling tool for system evaluation and validation (Hadis He Idari, 2013). The Petri Net model offers excellent analysis in terms of both qualitative and quantitative behavior for hardware systems (Maciel et al., 1998).

For a given system model in Petri Net analysis, the qualitative analysis facilitates in identifying feasible behaviors of the system without time factor considerations, whereas the quantitative assessment aims in analyzing them as a measure of occurrence probability (Franco Cicirelli et al., 2015). After an introduction to the Peri Net Performance Model in Section 1, this work is focused on analyzing various qualitative behavioral properties of PN models, such as reachability, boundedness, liveness, coverability analysis, reversibility, persistence, synchronic distance form and fairness analyzer, which is also discussed in Section 3. Section 2 describes the methodology with its architectural view of ROR1200. Various performance analytical factors along with the Petri Net based analysis are used to evaluate and to validate the ROR1200. These are discussed in Section 3 and Section 4. Section 5 concludes the work by summarizing the contributions and findings of the research work conducted for analyzing the proposed ROR1200 system.

2. METHODOLOGY

The Petri Net model offers an excellent analytical technique in terms of both qualitative and quantitative aspects for the hardware systems, specifically related to their behavior (Maciel et al., 1998). To enhance the performance of the open core processor, the ROR1200 was designed with such a reconfiguration technique (Maheswari et al., 2013).

2.1. Architectural View

The ROR1200 is a five-stage pipeline soft-core processor. The CPU system design of ROR1200 with reconfigurable HPRC (High Performance Reconfigurable Computing) unit, RRF (Reconfigurable Register File) and the Hazard Controller unit is shown in Figure 1 and the internal architectural view of ROR1200 is shown in Figure 2.

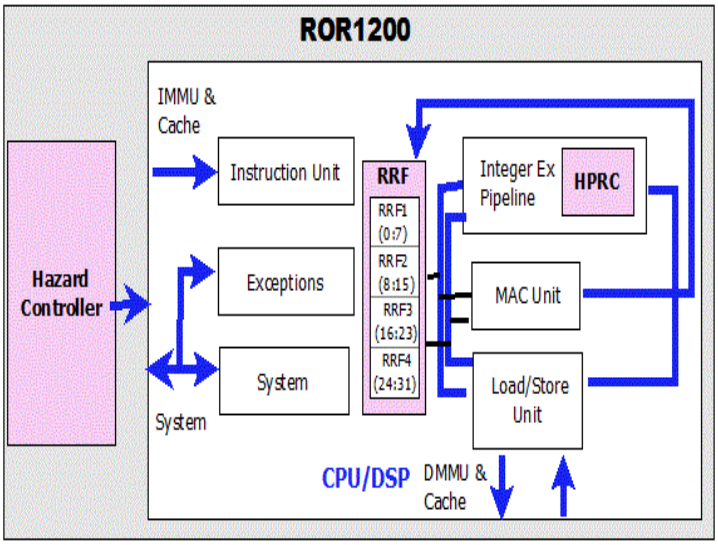


Figure 1 System design of ROR1200

.

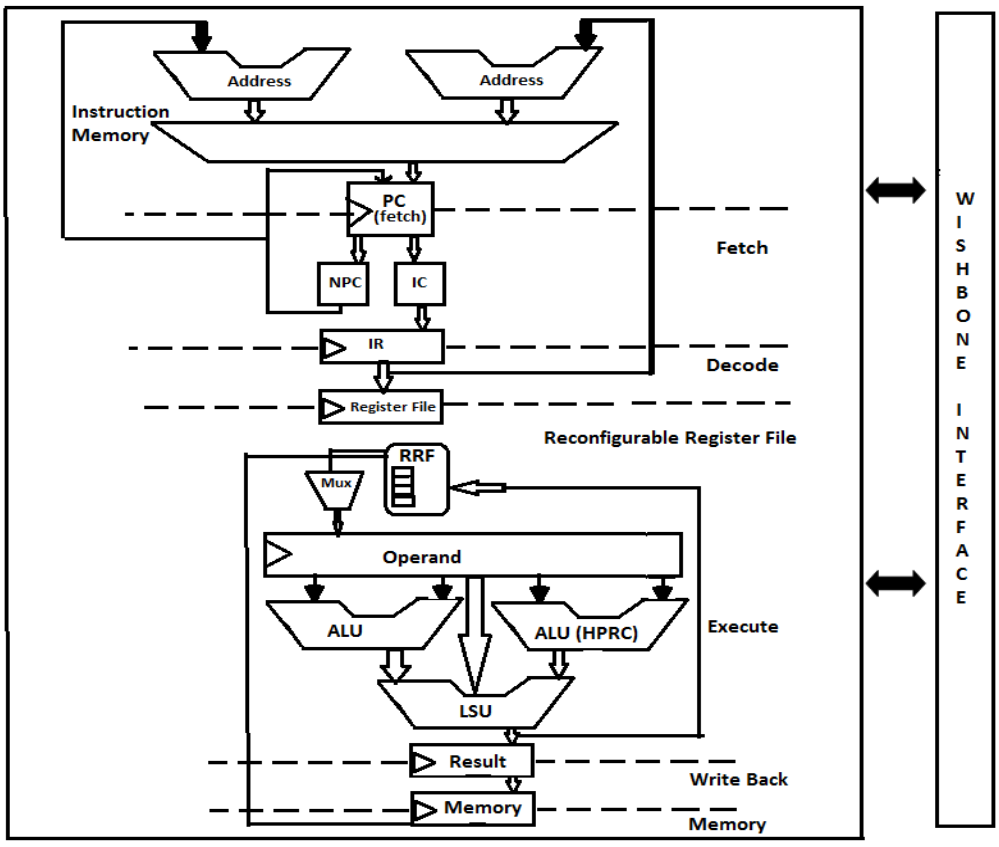


Figure 2 Internal architectural ROR1200

3. EXPERIMENTAL OUTPUT OF ANALYZING ROR1200 USING PETRI NET

Petri Net is a bipartite model which represents a dual mode of formal description, such as graphical and mathematical formulation, which ensures the logical interactions and dynamic modeling of complex design, including embedded systems. The graph-based Petri Net model consist of three major components of set-like places (P_i), transitions (T_i) and directed arcs (A_i). Formally, the Petri Net is expressed as $P_n=\{P_i, T_i, A_i\}$. Any Petri Net model is said to be in an execution mode when it is firing the set of transitions (T_i), which intrinsically moves the token from the input to its output node place. The final execution PN model consists of five tuples, such as (P, T, F, W, and M_0), where P is a finite set of places, T indicates the finite set of transitions, F represents finite sets of arcs connected from place to transition and also from transition to place, W is the number of weights on the arcs and M_0 indicates the initial marking, i.e. the number of tokens present in place P_0. Geometrically, the place is indicated by a circle and bar is used to represent the space between the places. In this data flow, if a firing happens between condition and events, then it is referred as an input function and if a firing exists between events and conditions, then the function is called an output function (Murata, 1989). In system modeling, all system conditions are represented by places and their events are mapped as transitions.

3.1. ROR1200 Petri Net Model

Parallel processing systems can be modeled as Petri Nets by assigning transition nodes to process and place nodes from inputs/outputs (Murata, 1984). A PN is used to illustrate the functionalities of all the subsystems of the model and it also describes the interaction between those subsystems through token exchange. The PN model is designed using the HiPS (Hierarchical Petri Net Simulator) tool for the execution unit of ROR1200, which is shown in Figure 3 and and the token transition between the places is shown in Figure 4.

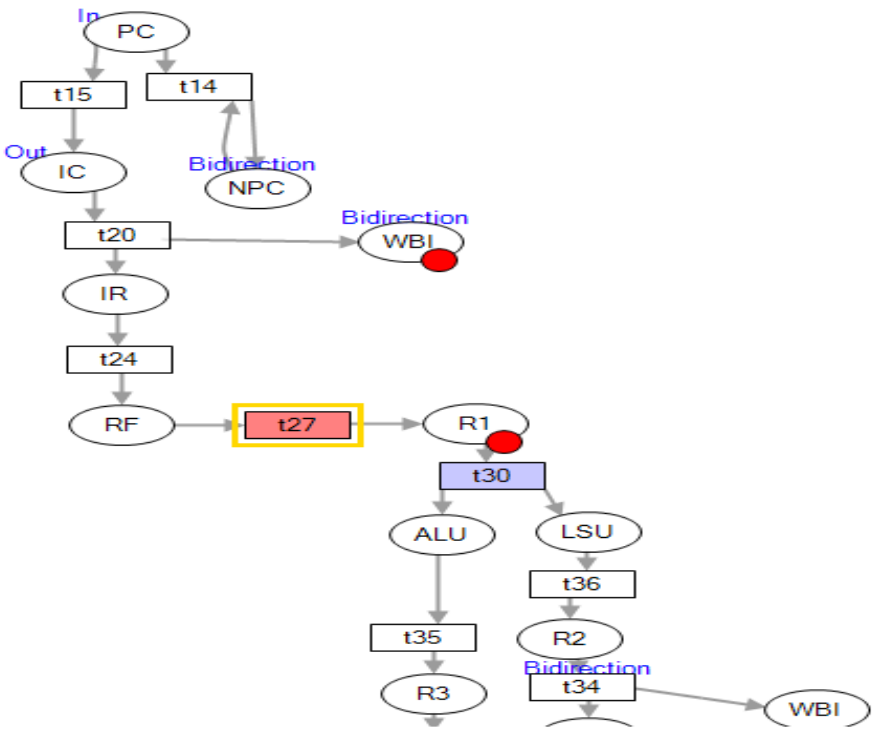


Figure 3 Petri Net execution unit of ROR1200

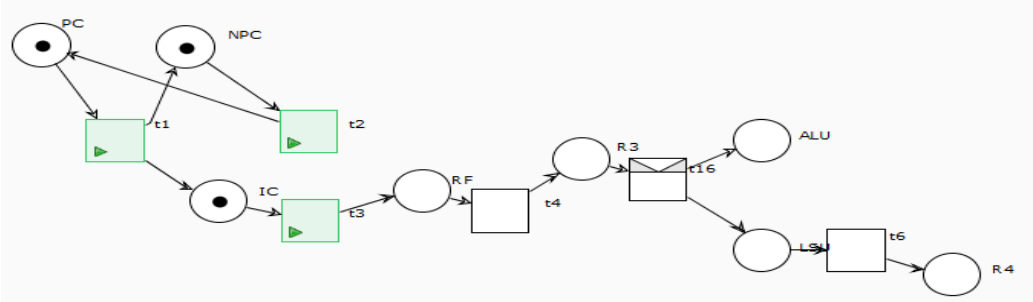


Figure 4 Petri Net execution unit of ROR1200 token transition between places

The PN model is driven by two execution rules, such as enabling and firing. When all the places in the transition hold at least one token, then the transition is said to be in an enabling state. During the firing transition, the token at the input place is moved to the output place such that the place ($P_i \epsilon PN$) has a new marking as shown in Equation 1.

$$M'(P) = M_0(P) - I(P_i, t_i) + O(P_i, t_i), \forall P \in PN \quad (1)$$

The place and transition of the each input node and corresponding output transition with initial marking is given as:

$P_i=\{P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8,P_9,P_{10}\}$ and
$T_i=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7\}$
$I(t_1)=\{P_1\}$ $O(t_1)=\{P_2,P_3\}$
$I(t_2)=\{P_2\}$ $O(t_2)=\{P_4,P_5\}$
$I(t_3)=\{P_3\}$ $O(t_3)=\{P_6,P_7\}$
$I(t_4)=\{P_4,P_5\}$ $O(t_4)=\{P_8\}$
$I(t_5)=\{P_6,P_7\}$ $O(t_5)=\{P_8\}$
$I(t_6)=\{P_8\}$ $O(t_6)=\{P_9,P_{10}\}$
$I(t_7)=\{P_9,P_{10}\}$ $O(t_1)=\{P_1\}$
$M_1=\{1,0, 0, 0, 0, 0, 0, 0, 0, 0\}$

where P is place, T is transition, I is input, O is output and M is token marking. The special cases of firing transitions are source transitions and sink transitions. In those cases, the transition always is enabled, but it does not generate any token. The average firing transition delay for a token at marking (M_n) is given as $[d(t_n, M_n)]^{-1}$, where d is the delay.

3.2. Behavioral Properties

When the properties of the PN model are evaluated based on their dependency on the initial marking M_0, then the property is known as a behavioral property. The formalization and analysis of behavioral properties of the Petri Net enable us to identify errors at an early stage of the design. The various behavioral properties analyzed in the PN model are reachability, boundedness, liveness, coverability analysis, reversibility, persistence, synchronic distance form and the fairness analyzer.

3.2.1. Reachability

The reachability property of the Petri Net model helps to analyze the dynamic behavior of parallel systems. In the given bounded system, when the firing sequence from the initial marking M_0 is transformable to M_n, then it is proven that reachability property is true to the given model. Figure 5 represents the reachability outcome of model PN. For an unbounded net,

the reachability tree will be infinitely large. To create a finite tree, a special symbol ω is defined such that, $\omega + n = \omega$ for any integer n.

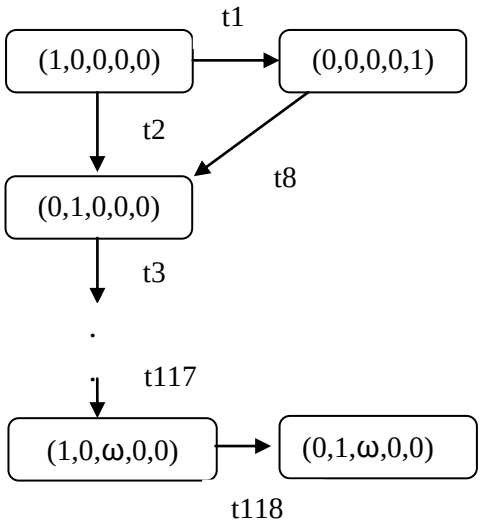


Figure 5 Reachability graph

Thus the reachability analysis (R_A) represented in Figure 5 is proving that PN model holds $M_n \in R_A(N, M_0)$.

3.2.2. Boundedness

The PN model is said to be k-bounded, when the total number of tokens does not cross the bound value *k* at each place *P* with the marking *M*.

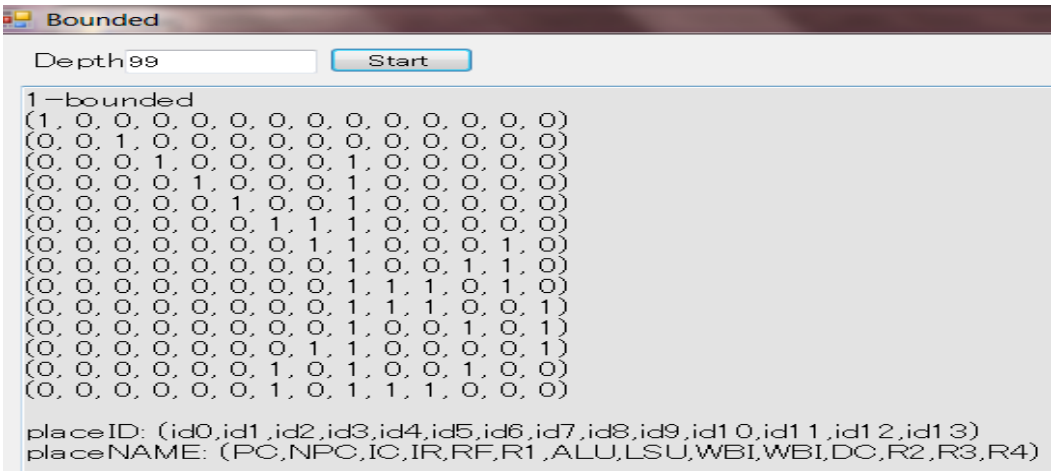


Figure 6 Boundedness analysis

Figure 6 shows the boundedness analysis for the depth *value=99*, also satisfying the reachability property of the system. Since it is 1-bounded, the target system is said to be in a safe condition. In general, the equation for boundedness is given in Equation 2.

$$M = \{M(P_i) \leq k, M \in R_A(M_0)\} \quad (2)$$

Ensuring the boundedness property to safe guarantees, there will not be any buffer/register overflow in the designed system, irrespective of the firing sequence of the token. The PN *(N,M_0)* with initial marking M_0 is safe when the maximum number of tokens in places is identified as *1* for every $M \in R(N, M_0)$ (Saúl-Alonso et al., 2016).

3.2.3. Liveness

Irrespective of the firing sequence, trueness to this liveness property convinces us that the designed model operates in a deadlock-free mode.

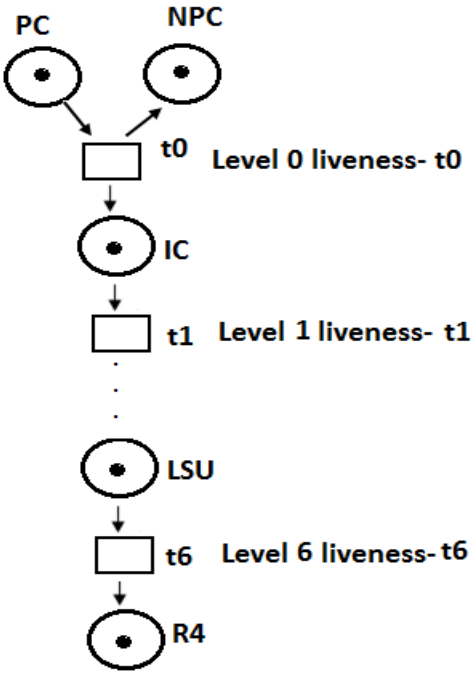


Figure 7 Liveness checker

The execution result towards liveness is given in Figure 7. In general, the liveness is defined in Equation 3.

$$L(k) \Rightarrow L(k-1) \Rightarrow L(k-2) \ldots \Rightarrow L(1) \tag{3}$$

The liveness is true if very transition of the PN is transparent.

3.2.4. Coverability analysis

The property coverability is also referred as potential fireability. Any transition *t* in the given marking *M'* is said to be coverable, if the model PN is true for the given condition, given in Equation 4.

$$M'(P) \geq M(P) \forall P \in PN \tag{4}$$

If *M'* is not coverable by the net, then that transition *t* is said to be dead. Figure 8 represents the coverability of the PN model from initial marking M_0 to M_7. The coverability graph designed for ROR1200 holds 243 vertices and 877 edges.

3.2.5. Reversibility

The reversibility property ensures the presence of the cyclical behavior of the system. The PN system model holds the reversible property, for any given node *N*, if the net is reachable to its initial marking M_0 or any initial state *M* given in Equation 5.

$$R(M) \rightarrow M' \forall M \in N \tag{5}$$

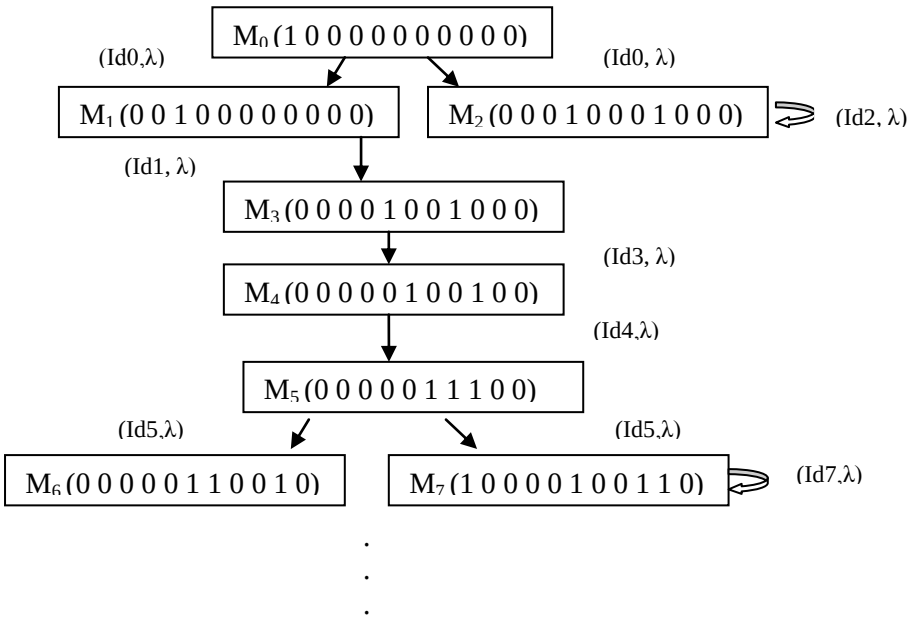


Figure 8 Coverability analyze

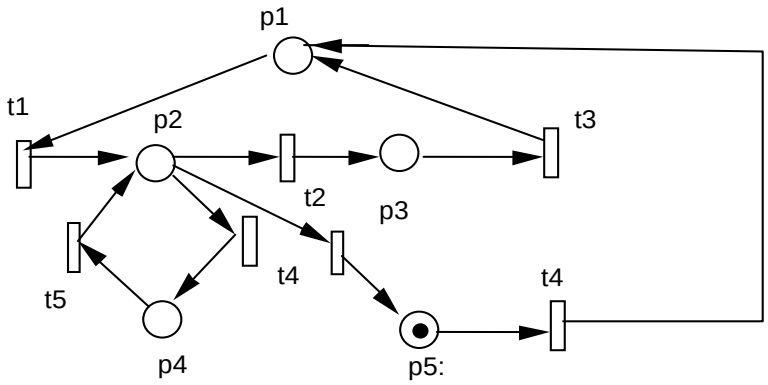


Figure 9 Reversibilty analysis

Figure 9 shows the reversibility analysis of the PN with an invariant activation. Thus existence of the reversibility property always ensures its recoverability behavior even under system failure.

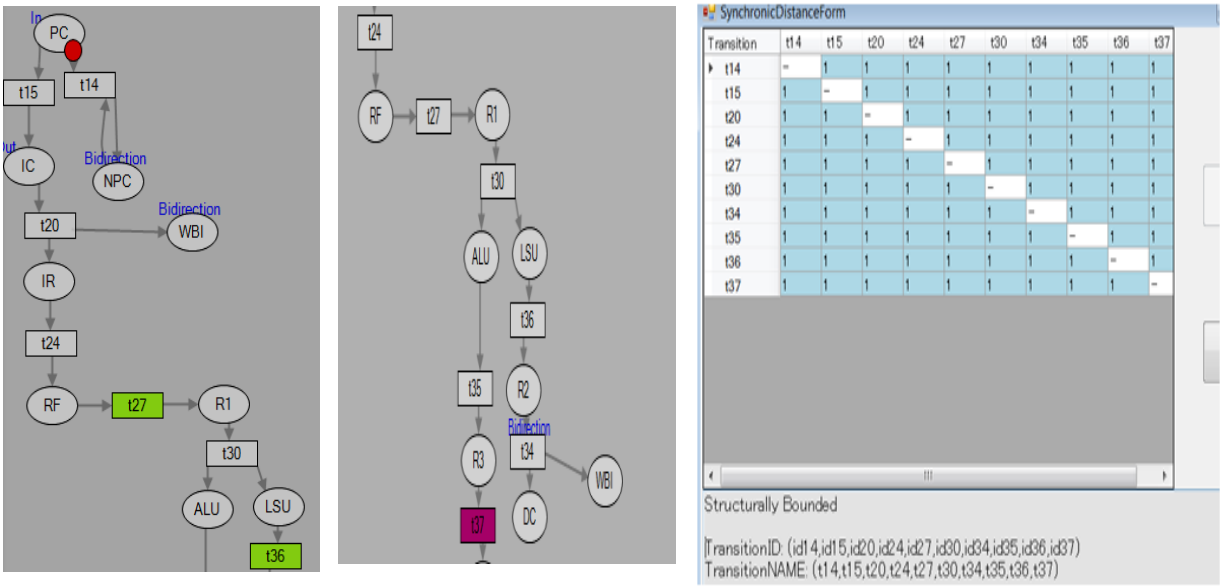
3.2.6. Persistence

The persistence property plays a very important role in analyzing parallel processing concepts. For any two active transitions (*t1* and *t2*) in the net, if the firing sequence of transition *t1* does not deactivate the other transition *t2*, then the PN model holds the persistence property. The PN model with persistence is also referred as a conflict-free Petri Net model. In the given PN model, all the marked graphs (N,M_n) have the persistence property.

3.2.7. Synchronic distance form and fairness analyzer

The degree of interdependence/mutual-dependence of transitions between any two places is identified by synchronic distance form. In the PN model shown in Figure 10a, transitions t_{15} and t_{16} indicate that the net executes in a parallel manner. The fairness analysis falls under two categories, such as Bounded Fair net (B-Fair) and unconditional fair net (global net). The B-fair net with finite transition is represented by a green color transition (t_{27} and t_{36}) in the PN model

as shown in Figure 10a. The global fair with an infinite transition is represented in the purple color transition *(t_{37})* to indicate the general transition of an unbounded net as represented in Figure 10b and the transition table is shown in Figure 10c.



(a) B-Fair net (b) Unconditional net (c) Transition table

Figure 10 Synchronic distance form and fairness analyzer

3.3. Structural Properties

The properties, which depend on the topological structure of the PN model, are known as the structural properties. The structural analyzer tool in the HiPS is used to analyze these structural properties.

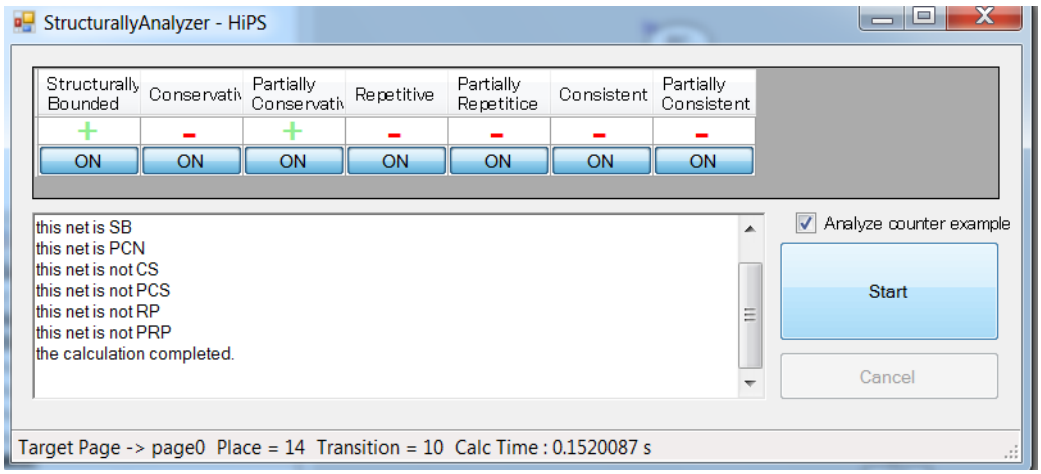


Figure 11 Structural analysis

The resulting outcome, as shown in Figure 11, represents that the given PN model is structurally bounded, holding structurally bounded and partially conservative properties.

3.3.1. Incidence Matrix

The dynamic behavior of a parallel system is analyzed using an incidence matrix whose transition table and incidence matrix are shown in Figure 12.

Incidence Matrix - HiPS

Transition / Place	id0(PC)	id1(NPC)	id2(IC)	id3(IR)	id4(RF)	id5(R1)	id6(ALU)	id7(LSU)	id8(WBI)	id9(WBI)	id10(DC)	id11(R2	id12(R3)	id13(R4)
id14(t14)	−1	0	0	0	0	0	0	0	0	0	0	0	0	0
id15(t15)	−1	0	1	0	0	0	0	0	0	0	0	0	0	0
id20(t20)	0	0	−1	1	0	0	0	0	1	0	0	0	0	0
id24(t24)	0	0	0	−1	1	0	0	0	0	0	0	0	0	0
id27(t27)	0	0	0	0	−1	1	0	0	0	0	0	0	0	0
id30(t30)	0	0	0	0	0	−1	1	1	0	0	0	0	0	0
id34(t34)	0	0	0	0	0	0	0	0	0	1	1	−1	0	0
id35(t35)	0	0	0	0	0	0	−1	0	0	0	0	0	1	0
id36(t36)	0	0	0	0	0	0	0	−1	0	0	0	1	0	0
id37(t37)	0	0	0	0	0	0	0	0	0	0	0	0	−1	1

Figure 12 Transition table of Incidence Matrix

The stated equation for the incidence matrix A for the PN model is shown in Equation 6.

$$M_P = M_{P-1} + A^T C_P \tag{6}$$

where P is place with $P = 1,2..n$, C_P is control vector and M_P is marking at place P. The destination marking M_d is reachable from the initial marking M_0 through a set of firing sequences $\{P_1,P_2..P_n\}$ with the given state shown in Equation 7.

$$M_d = M_0 + A^T \sum_{P=1}^{d} C_P \tag{7}$$

Figure 13 represents the firing sequence obtained for the initial marking $M_0=(1001000)^T$ by substituting M_0 in Equation 6 above.

$$\begin{pmatrix}1\\0\\0\\1\\0\\0\end{pmatrix} + \begin{pmatrix}-1&0&1&0&0&1\\1&-1&0&0&0&0\\0&1&-1&0&0&0\\0&0&1&-1&0&0\\0&0&0&1&-1&0\\0&0&0&0&1&-1\end{pmatrix} * \begin{pmatrix}1\\1\\0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\\1\\0\end{pmatrix}$$

Figure 13 Firing sequence for intial marking $M_0=(1001000)^T$

3.3.1.1. Necessary condition of reach-ability

By modeling ROR1200 using the Petri Net, both behavioral and structural properties existing in a parallel system are analyzed. The design of the PN model for the ROR1200 allows the translation from net level metrics into system level metrics. At the end of modeling, the various performance indices that need to be mapped with designed system are Mean sojourn time (S_t), Average number of tokens (T_{avg}),Place idle time (P_{idle}), Total number of transitions (N_t), Transition effective firing rate (E_t). Sojourn time is the average time spent by the system at a marking M. The mean sojourn time is shown in Equation 8.

$$S_t = \frac{\sum_{i=1}^{n} N*\tau}{N_t} \tag{8}$$

where N is total number of tokens at initial clock cycle, n is number of transition, τ is interval of the cycle, N_t total number tokens occurred at place P_i from initial clock cycle until the current cycle. The average number of tokens is shown in Equation 9.

$$T_{avg} = \frac{\sum_{i=1}^{n} N*\tau}{N_t} \tag{9}$$

where N_s is total simulation time.

The transition effective firing rate is shown in Equation 10.

$$E_t = \frac{N_f}{N_s} \tag{10}$$

where N_f is total number of firing transitions. The simulation result of PN gives 96.7% of the confidence interval; the mean sojourn time ranges between 1% to 6%, and the average error rate falls between 1.91% and 2.28%. The average number of tokens is identified as 1.83 with a reduced place idle time of 0.08 ns. The various properties are analyzed and the model outcomes with the stated equation for the Petri Net model are shown in Table 1.

Table 1 Petri Net model properties analysis

Property	Reachability	Boundedness
Equation	$M_n \in R_A(N,M_0)$	$M = \{M(P_i) \le k, M \in R_A(M_0)\}$
Inference	PN of ROR1200 has 165 reachable markings out of which 133 markings are legal markings.	PN is k-bounded, therefore the net is safe.
Property	*Liveness*	*Coverability*
Equation	$L(k) \Rightarrow L(k-1) \Rightarrow L(k-2) \ldots \Rightarrow L(1)$	$M'(P) \ge M(P) \forall P \in PN$
Inference	PN guarantees Deadlock free model	Minimal covering is ensured
Property	*Reversibility*	
Equation	$R(M) \rightarrow M' \forall M \in N$	
Property	*Incidence Matrix*	
Equation	i) $M_P = M_{P-1} + A^T C_P$ where $P = 1,2..n$ ii) $M_d = M_0 + A^T \sum_{P=1}^{d} C_P$	
Inference	The output markings of incidence matrix ensure the over-approximation of actual reachable marking.	

4. BOUND LEVEL ANALYSIS

In terms of technological perception, further research is required to explore the application to meet high performance standards with a lower power at a reduced cost. Performance analysis is made on both extreme cases, such as upper bound analysis and lower bound analysis, based on the dependency level. In upper bound parallel analysis, all the *N* input operations do not have data dependency, whereas in the lower bound serial analysis, all the input *N* operations have data dependency on their predecessors (Ling-Pei, 2002). These two analyses provide information required to verify whether the given task is bounded by memory or not, which in turn confine the speedup potential of that region whenever the obtained memory bandwidth is greater than the upper bound. Depending upon these values, the required clock period is estimated to speed up the process.

4.1. Upper Bound Analysis

For a non-dependence dataset, any parallel computational unit targeting high performance always requires the minimum number of cycles to complete the execution. It has been characterized by the task which disregards data dependency with all parallel operations supporting maximum pipeline processing, which is expressed in Equation 11.

$$N_{op} = \sum_{k=1}^{N} i_n \tag{11}$$

where N_{op} indicates the number of operations, i_n is the number of inputs, N is the number of multiple functional units connected to a reconfigurable register file.

The latency of the upper bound level is calculated as shown in Equation 12.

$$L_u = \max_{i=1}^{N}(\sum_{l=1}^{L} I_l * i_n) \tag{12}$$

where L_u indicates latency of upper bound level, I_l represents the initiation interval of i^{th} functional unit with latency L.

4.2. Lower Bound Analysis

The lower bound analysis is performed in pipeline processing, when there exists a maximum dependency of data for the longest execution. The throughput for this lower bound is obtained by sequencing all executions of the functional unit.

The latency of lower bound level is calculated as in Equation 13.

$$L_l = \min_{i=1}^{N}(\sum_{l=1}^{L} T_l * i_n) \tag{13}$$

where L_l indicates latency of lower bound level, T_l represents the inexpensive, smallest simple instruction task of i^{th} functional unit with latency L.

4.3. Bound Ratio

The bound ratio B_r gives the ratio of upper and lower bound levels

$$\mathrm{Br} = \frac{L_u}{L_l} = \frac{\max_{i=1}^{N}(\sum_{l=1}^{L} I_l * i_n)}{\min_{i=1}^{N}(\sum_{l=1}^{L} T_l * i_n)} \tag{14}$$

Table 2 represents the performance analysis of upper bound and lower bounds using a Media Benchmark on three Soft Core Processors (SCPs), such as OR1200, MicroBlaze, and ROR1200.

Table 2 Performance analysis of Upper and Lower Bounds

Soft-core Processor	Commercial		Open-source					
	MicroBlaze		OR1200		ROR1200			
Functional Unit	*FU1*	*FU2*	*FU1*		*FU1*		*FU2*	
Operators	+/-	*	+/-	*	+/-	*	+/-	*
T_l/I_l	1/1	4/2	1/1	8/4	1/1	4/2	1/1	4/2
Total Latency (ns)	12	28	32		8		28	
Upper Bound L_u (ns)	28		32		28			
Lower Bound L_l (ns)	42		72		40			
Bound Ratio B_r	1.5		2.25		1.43			

From Table 1 it has been inferred that the upper and lower bound values are directly proportional to the value of N. The lower bound analysis is used to perform the task load test with the measurement of the instruction size N and the task T. Similarly, the upper bound analysis helps to achieve the characteristics test with the distribution of functional units between the tasks. The bound level analysis graph is shown in Figure 14.

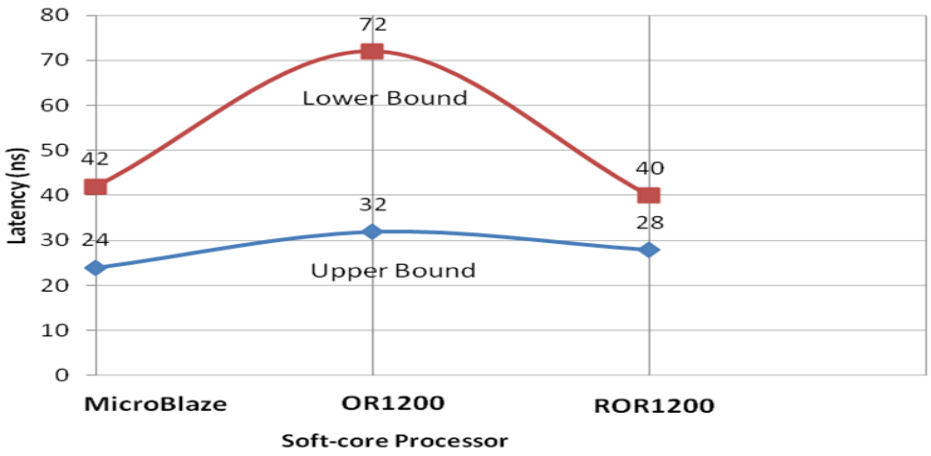


Figure 14 Bound analysis graph

5. CONCLUSION

To ensure proof of the concept, the Petri Net model for ROR1200 was designed to verify the various behavioral and structural properties like reachability, boundedness, liveness, coverability, reversibility, persistence, synchronic distance form and fairness analyzer for safeness and an incidence matrix with stated equations. The various analysis metrics in Bound Level Analysis, such as upper bound analysis, lower bound analysis and the bound ratio with respect to the dependency level of data and its latency were carried out to validate the system. The results obtained for ROR1200 were the upper bound latency (L_u) as 28 ns, lower bound (L_l) as 40 ns with the bound ratio (*Br)* as 1.42, which show improved performance when compared to the OR1200 and MicroBlaze soft-core processors.

6. REFERENCES

Bause, F., Kemper, P., 2005. *QPN-Tool for Qualitative and Quantitative Analysis of Queuing Petri Nets*. Computer Performance Evaluation Modelling Techniques and Tools, Springer Lecture Notes, Volume 794, pp. 321–334

Cicirelli, F., Nigro, C., Nigro, L., 2015. Qualitative and Quantitative Evaluation of Stochastic Time Petri Nets. *In*: Proceedings of the Federated Conference on Computer Science and Information Systems, Volume 5, pp. 763–772

Gaubert, S., Mairesse, J., 1999. Modeling and Analysis of Timed Petri Nets using Heaps of Pieces. *IEEE Transactions on Automatic Control*, Volume 44(4), pp. 683–697

Gilmore, S., Hillston, J, Holton, R., Rettelbach, M., 1996. Specifications in Stochastic Process Algebra for a Robot Control Problem. *International Journal of Production Research*, Volume 34(4), pp. 1065–1080

Heidari, H., 2013. Analysis of Reconfigurable Processors Using Petri Net. *International Journal of Computer Network & Information Security*, Volume 9(1), pp. 28–36

Heiner, M., Popova-Zeugmann, L., 1997. On Integration of Qualitative and Quantitative Analysis of Manufacturing Systems using Petri Nets. *In*: Proceeding 42nd IWK '97, Ilmenau, Volume 1, pp. 557–562

Jennings, A.D., Nowatschek, D., Prickett, P.W., Kennedy, V.R., Turner, J.R., Grosvenor, R.I., 2000. Petri net Based Process Monitoring. *In*: Proceedings of Comadem 2000, Houston, USA: MFPT Society pp. 643–650

Kant, K., 1992. Introduction to Computer System Performance Evaluation. New York: McGrawHill Colege. First Edition

Kemeny, J.G., Snell, J.L., 1960. *Finite Markov-Chains*. Van Norstrand, Springer Verlag, Second Edition

Kung, L-P., 2002. *Obtaining Performance and Programmability using Reconfigurable Hardware for Media Processing*. Massachusetts Institute of Technology, pp. 93–100

Lotfifar, F., Shahhoseini, H.S., 2008. Performance Modeling of Partially Reconfigurable Computing Systems. *In*: Proceedings of the 6th IEEE/ACS International Conference Systems and Applications (AICCSA), Doha, pp. 94–99

Maciel, P., Barros, E., Rosenstiel, W., 1998. A Petri Net based Approach for Performing the Initial Allocation in Hardware/Software Co-design. *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, Volume 1, pp. 505–510

Maheswari, R., Pattabiraman,V., 2013. Dynamic Reconfigurable Computing in OR1200 Core of Embedded System. *International Journal of Computer Technology and Electronics Communication*, Volume 2(2), pp. 1–9

Murata, T., 1984. *Modeling and Analysis of Concurrent Systems*. Book of Software Engineering, Van Norstrand Reinhold Company Inc.

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications. *In*: Proceeding of the IEEE, Volume 77(4), pp. 541–580

Nuño-Sánchez, S-A., Ramírez-Treviño, A., Ruiz-León, J., 2016. Structural Sequence Detectability in Free Choice Interpreted Petri Nets. *IEEE Transactions on Automatic Control*, Volume 61(1), pp. 198–203

Radom, M., Rybarczyk, A., Formanowics, P., 2015. Cluster Based Analysis of Petri Net Properties. *In*: BIOTECHNO 2015: The Seventh International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies, pp. 8–10

Wang, G., Chen, D., Chen, J., Ma, J., Chen, T., 2009. A Performance Model for Run-Time Reconfigurable Hardware Accelerator. *In*: Proceedings International Symposium on Advanced Parallel Processing Technologies, Volume 5737 of the series Lecture Notes in Computer Science, pp. 54–66