The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27-29, 2001; Przytok near Zielona Gora, Poland

Automatic HDL Generation for A DES Codec for an Encrypted NFS Server based on an Extended Petri Net

Shin'nosuke Yamaguchi, Katsumi Wasaki, Yasunari Shidama, Pauline Naomi Kawamoto

Faculty of Engineering, Shinshu University
4-17-1 Wakasato Nagano-city, Nagano 380-8553, Japan, *xrei@cs.shinshu-u.ac.jp*

Abstract. *In this paper, we propose a method for automatically generating the Hardware Description Language (HDL) source for a parallel pipelined DES enciphering circuit for the Network File System based on an extended Petri net. For this work, the authors have proposed an extended Petri net, called a "Logical Coloured Petri Nets (LCPN)", suitable for the design of complex control system processes and discuss its methods of evaluation. First, the logical verification of a circuit and its timing checks based on zero–delays are done at the HDL level. Then, delay information for real devices are used to verify the timing and operation speeds at the FPGA level. It is the purpose of this research to improve the reliability of circuit operation at the time of design by verifying the correctness of pipeline and parallel operations with the proposed new technique.*

Keywords. *extended Petri Net, Hardware Description Language*

1 Introduction

There has been a rapid expansion recently in computer systems making it necessary to develop flexible microprocessor units (MPUs). In general, MPU sequence controllers have to control multiple executing units which work together concurrently and synchronously while maintaining operation synchronization. The sequence controller for the parallel/pipelined ALU has been used as an important block in MPUs for advanced computer systems and is usually designed based on sequence ladder languages/diagrams, state charts, decision tables or circuit diagrams. However, these methods have the following problems [14]:

(1) A variety of designing errors and bugs are easily introduced into parallelized behavior.

(2) Verification of the correctness of specifications is difficult.

(3) Tracing control flow/software is difficult by anyone other than the original designers.

A descriptive model of sequence control by Petri nets has become attractive due to its simplicity [1][2]. Petri net is a graphical model which makes understanding the control flows easy. The mathematical nature of the Petri net can be used to obtain information about the behavior of systems which operate in a dynamic environment. Especially when time or safety factors make actual simulations of a system unfeasible, a study of the relationships between the mathematical objects of a Petri net can reveal such conditions as deadlocks, traps, reachability of marking states, etc. which aid in the verification of system operations. Examples of applying conventional Petri nets(PNs) and Colored Petri

nets(CPNs) to system design and analysis have been given in such areas as hardware design by Jensen [3], deadlock verification of Ada programs by Murata et al [4], FA control by Nagao et al [5][10][11][12][13], and computer system models by Miller [14].

The following problem exists when PNs and CPNs are actually applied to the work of describing control systems. The operation of these nets (firing conditions of transitions and the movements of tokens) is uniquely fixed. It is necessary to use many transition and place elements to represent the branching of conditions in a process and as a result, the net scale increases. In this paper, we describe the automatic HDL generation for a parallel pipelined DES enciphering circuit based on an extended Petri net. For this, the authors proposed an extended Petri net called a"Logical Coloured Petri Net (LCPN)", suitable for the design of complex control systems processes and discuss its methods of evaluation in [6][7][13].

2 Logical Coloured Petri Net - (LCPN)

In this section, we describe the definition of a logical coloured Petri net (LCPN) which can be used to improve the description capabilities of conventional Petri nets and colored Petri nets. Specifically tokens in LCPN have data (colours) and firing conditions are given by an arbitrary logical expression which is written in terms of the presence of tokens in input places and the data values of tokens. This feature greatly simplifies the work of expressing diverging conditions in a system. Once an LCPN model is developed, it can be analyzed by using reachability trees and simulations [8][9].

Definition 1 *LCPN : A logical coloured Petri net is a tuple of sets* $N_E = (S_E, T_E, F_E, M_E)$ *which fulfills each of the following conditions.*

(1) Let $S_E = \{s_1, s_2, \cdots, s_n\}$ *and* $T_E = \{t_1, t_2, \cdots, t_m\}$ *be the sets of the place and transition elements, respectively. Let* $F_E = \{f_1, f_2, \cdots, f_l\} \subseteq (S_E \times T_E) \cup (T_E \times S_E)$ *being a set of arcs from places to transitions and transitions to places.*

(2) The mark of each place $s_i \in S_E$ $(i = 1, 2, \cdots, n;\ \ n = \mid S_E \mid)$ *can take the value of a natural number from 0 to N. This is denoted by* $\mu(s_i)$*. We assume* $\mu(s_i) = 0$ *if there is no(empty) marking on* s_i*. Here, we define the function* μ *as* $\mu : S_E \longrightarrow \{0, 1, 2, \cdots, N\}$

(3) The capacity (maximum number of marks) of each place s_i *is 1. The set of all possible marks from (1) and (2) is represented as a mapping from* S_E *to* $\{0, 1, 2, \cdots, N\}$*. Thus, we define the cartesian product set* M_E *as* $M_E = \{0, 1, 2, \cdots, N\}^{S_E}$

(4) ${}^*t_j$ *represents the set of all places (input places) which have an arc extending to* $t_j \in T_E$ $(j = 1, 2, \cdots, m; m = \mid T_E \mid)$*. Similarly,* $t_j{}^*$ *represents the set of all places (output places) with an arc extending from* t_j*.*

$$\begin{aligned} {}^*t_j &= \{s \in S_E :\ (\exists f)(f \in F_E, f = (s, t_j))\} \\ t_j{}^* &= \{s \in S_E :\ (\exists f)(f \in F_E, f = (t_j, s))\} \end{aligned}$$

(5) The firing evaluation of a transition t_j *for an arbitrary marking* $\mu \in M_E$ *examines the firing condition* Φ^j*.* $\Phi^j(\mu \mid^* t_j)$ *is described by a logical expression in terms of the state* $\mu \mid^* t_j$ *of the places which belong to* ${}^*t_j$ *and is used to determine the next marking* $\mu' \in M_E$*.*

t_j *is called firable if* Φ^j *is evaluated to be true. It* t_j *is fired, a mark is removed from each place in* ${}^*t_j - t_j{}^*$*. Places in* $t_j{}^*$ *depend on the state of* ${}^*t_j$ *and are modified by the following* C^j*.*

$$C^j : \{0,1,2,\cdots,N\}^{*t_j} \longrightarrow \{0,1,2,\cdots,N\}^{t_j*}$$

$$if \quad \Phi^j(\mu|^* t_j) \; is \; true \quad then$$

$$\mu' = \begin{cases} 0 & : on \quad {}^*t_j - t_j{}^* \\ C^j(\mu|^* t_j) & : on \quad t_j{}^* \\ \mu & : otherwise \end{cases}$$

On the other hand, t_j is not firable if Φ^j is false. At this time, the state $\mu' = \mu$ is unchanged.

We can show the next marking resulting from a transition evaluation as a mapping f^j from state $\mu \in M_E$ of the places to $\mu' \in M_E$.

3 Modelling of Parallel Accumulators by LCPNs

3.1 Outline of FPGA development environment

In this section, we describe the design and analysis for the parallel accumulator with a Petri net tool made for trial purposes. Fig. 1 shows the data flow of the FPGA devel-

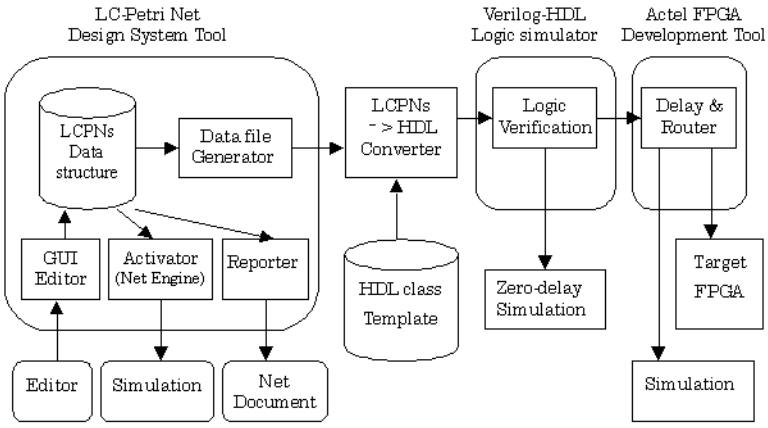

Figure 1: The FPGA development environment.

opment environment. This development environment is composed of the LCPN Design System Tool (see Fig. 2)[10], LCPN to HDL Converter, Verilog-HDL Logic Simulator, and Actel FPGA Development Tool.

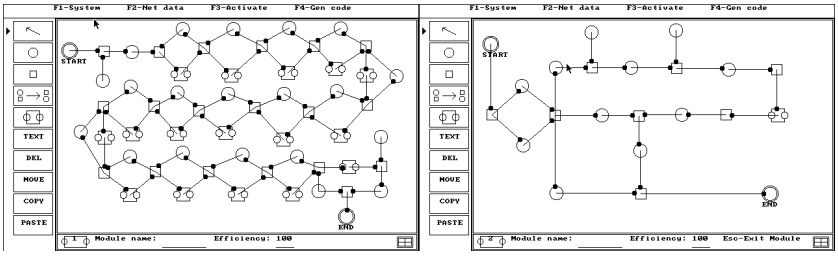

Figure 2: LCPN Design System Tool

At this stage, the tools examine the design's defects (validity of operations, presence of deadlocks, synchronous relation, etc. of the pipeline operation) in the accumulator

model. By using the Petri net, we can efficiently detect the presence of pipeline hazards and correct them. We can also verify the stability of the synchronous relations of parallel circuits from the net.

Next, the tools convert the net model which has been verified into existing hardware description language (Verilog-HDL). Verilog-HDL defines well each function module of the circuit (comparators, full adders, etc.) as a "Class" and associates the Petri net model from the "Class" of a sub-net and the circuit. The HDL source code files are all automatically generated with the HDL generator made by our group.

Finally, the HDL of the parallel operation machine generated automatically is mounted on the gate array with an FPGA layout tool which verifies the timing and operation speed at the circuit level. We can decrease the labor of verifying stability with regard to such problems as pipeline hazards, because we need only the checks of timing, operation speed, and operation results using the delay information of real devices.

3.2 Method of Converting LCPNs to HDL

We show the method of conversion from the LCPN structural data file (generated from the Petri net tool) to HDL below.

STEP 1 Expand sub-nets defined by the user
STEP 2 Replace sub-nets with simpler sub-nets (adder, multiplier modules, etc.)
STEP 3 Class generation of Verilog-HDL from sub-nets and transitions
STEP 4 Replace place elements with corresponding latches.
STEP 5 Replace arcs with wire instances.
STEP 6 Resolve information for the class template and connection wiring
STEP 7 Generate the target HDL source of Verilog-HDL

Figure 3: *LCPN to HDL Converter*

The LCPN to HDL converter performs the conversion sequence [STEP 1]-[STEP 7] from the net data file to HDL source file automatically. This conversion engine and the template database for trial purposes are made with the Borland Delphi5.0 development environment based on the Microsoft Windows95/98/NT operating system. This implementation has reduced the cost of HDL design and programming and gate array testing on site to one third of what development would have cost using conventional procedural language-based methods.

4 Example of Parallel DES Accumulator with LCPNs

In this work we used the LCPN to automatically generate HDL source code for a parallel accumulator in an actual DES enciphering circuit. The circuit was embedded into

a network file system (NFS) server in order to evaluate its efffctiveness. We codesigned the NFS with LCPN in [15].

The NFS described in RFC-1813, allows a local file system to mount a file system through a network. The NFS server and client distinguish files in a server by a unique file handle that is an integer number in 64 bits containing information of the file (name, i-node, update time, etc.)

The Data Encryption Standard(DES) cryptogram is an encoding algorithm announced in 1975. The DES cryptogram encodes a plain sentence in 64 bits by a key in 56 bits. Fig. 4 shows the one step operation in the DES cryptogram.In function f, the sentence is replaced by eight arrays (S box). The DES cryptogram respectively carries out its operations 16 times.

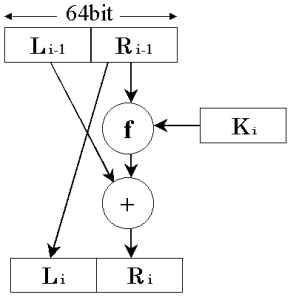

Figure 4:One step of DES cryptogram

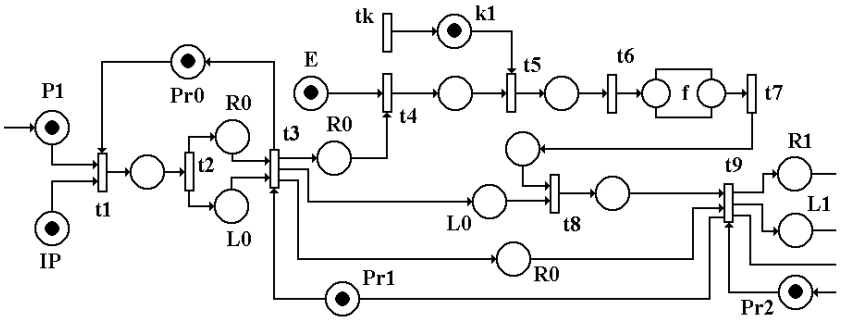

Figure 5:LCPN Example of Parallel DES Accumulator(subnet)

Next, the processing for execlusive OR is redesigned as a pipeline structure of 16 stages. Fig. 5 shows an example of a parallel DES accumulator with designed LCPNs based on this sequence. Here, we put the data divided into 64 bits in place P_1. Then, it is transposed by IP and then divided into L_0 and R_0. After the operation with the key schedule K_1 and the extension transposition E, etc. is executed, the exclusive OR of R_0 and L_0 is taken and the output is placed in R_1. In the first step, L_1 takes the value of R_0 as it is. We are attempting pipelining of 16 stages by carrying out exclusive processing controlled by place $Pr_n(n = 1, 2, \cdots, 16)$. While the file contents are being encoded, information on the file is handled in a different route.

In this pass, data is coded according to the character exchange style cryptogram. In this way, it is possible to keep shared file system information concealed while recording coded data.

To read back the encoded data, we repeat the f and exclusive OR operations 16 times, using the key schedule in reverse order. The net for one 16 stage DES pipeline for encryption file system code/decoding required 832 places, 400 transitions, and 1426 arcs.

5 Conclusion

We proposed and defined the concept of a logical coloured Petri net (LCPN) which improves the description capability of current PNs and CPNs. We modelled hardware for parallel pipelined DES accumulators for a network file server system by using LCPNs and analyzed the net model with a Peti net development tool environment. We implemented the accumulator hardware generated automatically from a net whose stability was verified and confirmed its operation on the simulator of the system.

In the future, we plan to study further the operation of LCPN and improve it as we apply it to CASE tools for developing system control software.

References

[1] T.Murata : "Petri Nets: Properties, Analysis and Applications ", Proc. IEEE, Vol. 77, No. 4, pp.541-580, 1989.

[2] J.L.Peterson : "Petri Net Theory and the Modelling of Systems", Prentice-Hall, Inc., 1981.

[3] K.Jensen : "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Vol. 1, Springer–Verlag, 1992.

[4] T.Murata, B.Shenker, and S.M.Shatz : "Detection of Ada Static Deadlocks Using Petri Net Invariants", IEEE Trans. Softw. Eng., Vol. 15 , No. 3 , pp.314–326, 1989.

[5] Y.Nagao, H.Ohta, H.Urabe and S.Kumagai : "Petri Net Based Programming System for FMS" IEICE Trans. Fundamentals. , Vol. E75–A, No. 10, pp.1326-1334, 1992.

[6] K.Wasaki, Y.Fuwa, M.Eguchi, and Y.Nakamura : "Extended Petri Nets for Control System Software", Technical Report of IEICE, COMP93-12, SS93-6, pp. 37-44, 1993.

[7] K.Wasaki, Y.Fuwa, M.Eguchi and Y.Nakamura : "Capacity of the Logical Colored Petri Net (LC-net) as a CASE Tool", Technical Report of IEICE, CAS93-69, pp. 101-108, 1993.

[8] P.N.Kawamoto, Y.Fuwa and Y.Nakamura: "Basic Concepts for Petri Nets with Boolean Markings", Journal of Formalized Mathmatics, Vol.4, No.1, 1993.

[9] P.N.Kawamoto, M.Eguchi, Y.Fuwa and Y.Nakamura : "The Detection of Deadlocks in Petri Nets with Ordered Evaluation Sequences", Technical Report of IEICE, SS92-29, KBSE92-50, pp. 45-52, 1993.

[10] M.Kitazawa,P.N.Kawamoto, Y.Fuwa, Y.Nakamura: "Petri Net Software Development for Parallel/Distributed Systems", *Proc. Parallel and Distributed Computing and Systems(PDCS'96)*, pp.418-420, 1996.

[11] T.Nakao, T.Yamagishi, P.N.Kawamoto, Y.Fuwa, Y.Nakamura: "A Petri Net Based Protocol Design Tool for Wireless Data Communication", *Proc. Multi-Dimensional Mobile Communications 96*, pp.307-311, 1996.

[12] A.Deodhar, E.C.Tan, A.Wahab : "FPGA Implementation of Discrete–Time Systems", *Fourth International Conference on Control, Automation, Robotics and Vision Proc.*, TE-2-2, pp.1328-1332, 1996.

[13] K.Wasaki, Y.Fuwa, M.Eguchi, Y.Nakamura : "Logical Coloured Petri Net Expanded to be Suitable making the Control System Model", *Fourth International Conference on Control, Automation, Robotics and Vision Proc.*, TA-2-2, pp.708-713, 1996.

[14] R.E.Miller : "A Comparison of Some Theoretical Models of Parallel Computations", *IEEE Trans. on Computers*, Vol. C-22, No. 8, pp.710-717, 1973.

[15] S.Yamaguchi, K.Wasaki, S.Shidama : "A process design for the Network File System model based on the Logical Coloured Petri Net", *Sixth International Conference on Control, Automation, Robotics and Vision (ICARCV'2000)*, December 2000.