Technical Research on Describing Reconfigurable Systems by Object Oriented Petri net

Jun Guo[1,2] Sheqin Dong [1] Kegang Hao[2] Satoshi Goto[3]

[1]Dept. of Computer Science of Tsinghua University, Beijin 100084
[2]Dept. of Computer Science of Northwest University, Xi'an 710069
[3]Graduate School of IPS of Waseda University, Kitakyushu

Abstract

An object oriented Petri net was proposed in order to describe reconfigurable systems. The formal definitions of this kind of Petri net were presented carefully. The methods of subnet partition were discussed in details. And techniques of mapping objects to reconfigurable platform were discussed as well. The features of the mentioned Petri net were summarized briefly.

Keywords: Petri net; reconfigurable systems; Object oriented

1. Introduction

Reconfigurable computing (RC) is a novel computing pattern nowadays. RC systems assign intensive computing tasks to reconfigurable hardware. Owing to the high computing speed of hardware, the system performance increase greatly. Therefore, RC systems play important roles in the aspects of image processing, biology computing, cryptogram algorithms and multimedia communication [1].
The techniques and methods of designing RC systems are quite different from traditional computers for the reason that reconfigurable hardware is adopted in the architecture. Software/hardware co-design method was regarded as the effective way to design RC systems. Firstly, a suitable model should be chosen to describe the RC systems. There are several models could be used, such as data/control flow diagram, FSM, Petri nets and UML. The formal method was argued as a promising candidate by reason of easy understanding and verification. Petri nets are formal method for describing concurrent, asynchronous and resource collision systems. The functions of system could be verified by running Petri net model. And the performance of system could be analyzed at the same time. In this paper, a kind of Object oriented Petri net (OPN) was proposed for describing RC systems. Section 2 discusses the formal definitions of OPN. Section 3 discusses the methods of assembling and disassembling objects of OPN. Section 4 discusses the techniques of mapping OPN to hardware/software of RC systems. Section 4 summarizes the research work.

*This work is supported by NSFC 90307005 and NSFC 60473126

2. Object Oriented Petri Net

Petri nets have been studied for many years and formally defined in mathematics. Petri net model could be presented by graphic symbols, which could describe the static structure and dynamic behaviors of system. But traditional Petri net is non-hiberarchy and non-modular and also lacks the ability to process data flow and mutual action. Many scholars introduced new concepts to Petri nets for their own applications requirements, such as colored token, object oriented, open ports [2,3,4,5]. These extended Petri nets are modular, re-useable and have the data process and mutual action ability. In this section, a practical object oriented Petri net is defined formally for modeling RC systems.

Definition 1 The basic Petri nets is five-tuple *PN=(P,T,F,W,M),* Where,

P is a infinite set of places

T is a infinite set of transitions,$T \cap P = \Phi$

F is a infinite set of arcs $F \subseteq (P \times T) \bigcup (T \times P)$;

W: $F \rightarrow \{1,2,\ldots\}$ is weight function

M: $P \rightarrow \{0,1,2,\ldots\}$ is marking represent by tokens M_0 is initial marking

Petri net can be represented by graphic symbols. Fig. 1 shows a typical example of Petri net, which describes a producer/customer system. In fig.1, a circle represents for a place, a rectangle represents for a transition. Tokens are represented by black dots or an integer.

Dynamic behaviors of Petri net can be defined by transition firing, which describe the system state transfer.

Definition 2 *PN* is a Petri net $\forall x \in P \bigcup T$

$$\bullet x = \{y | y \in P \bigcup T \wedge (y,x) \in F\}$$

$$x \bullet = \{y | y \in P \bigcup T \wedge (x,y) \in F\}$$

•*x* is the pre-sets of *x*

x• is the post-sets of *x.*

Definition 3 $\forall p \in \bullet t$, if $M(p) \geq W(p,t)$ t is said to be enabled.

Definition 4 if the present marking is *M* enabled transition t firing will cause the marking change into *M'*

$$M'(p) = \begin{cases} M(p) - W(p,t), & p \in \bullet t - t\bullet \\ M(p) + W(t,p), & p \in t\bullet - \bullet t \\ M(p) - W(p,t) + W(t,p), & p \in \bullet t \bigcap t\bullet \\ M(p), & \textit{other cases} \end{cases}$$

A data flow Petri net was defined so that the model can cope with the data flows.

Definition 5 data flow Petri net is six-tuple *DPN=(Q,P,T,F,W,M)*

Where Q is data place set; transition is extended to data operation; G is guard function; *P F W M* is already defined.

Object oriented conceptions were introduced to Petri net in order to get modular structure and reusable ability.

Definition 6 object oriented Petri net is OPN=*(O,DPN)* DPN is data flow Petri net O={o_1, o_2 ,…,o_v} is the infinite set of objects.

Definition 7 Class is the abstract of similar objects, consists of name, internal structure, ports. The internal structure of class contains all elements of OPN. Ports are places or transitions which communicate with outside objects.

According to definition 7, elements inside object are encapsulated. Objects communicate with each other through ports. Both place and transition can work as port. As is different from previous object oriented Petri net [2,3,6], where only places could play as ports. That means port can not sent messages actively. Messages can be sent actively through ports of transition in OPN. It is good for objects to communicate easily and get the ability of mutual action.

From the point of object oriented view, object is the instance of class. Class can be cited to create many instances of objects. As a result, objects can be reused in OPN. Reusability helps to modeling large scale systems.

Fig.2 shows the OPN model of producer/customer system. There are 2 objects encapsulated in rectangle: producer and customer. Transition t12, t22 and place p14, p24 are ports. Inside the object is a subnet of OPN. The details of object can be concealed.
Investigation of the example shows that OPN consists of four elements in fact: place, transition, arc and object. A system contains many objects, which connected with each other through ports. Considering the Petri nets, system consists of object subnet, which can be regarded as a module. Therefore OPN is a modular structure.

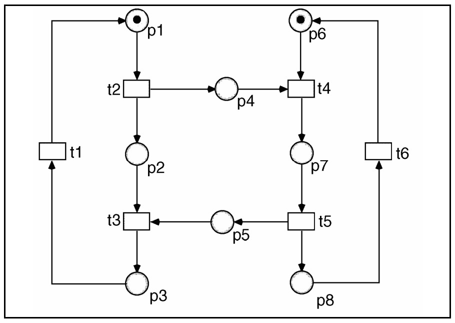

Fig. 1: Producer/customer system

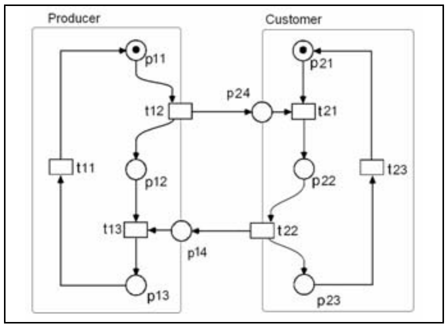

Fig. 2: Objects in producer/customer system.

3. Objects Assemble and Decompose

The objects in RC systems should be configured according to application goals under the constraints of runtime, cost and resource. In OPN model, a balance of multi-aim optimum can be achieved by assembling objects. Every task of the system can be described as an object subnet in OPN. To assemble all the objects will set up the application system. On the other hand, a complicated subnet can be decomposed into several simple objects, which is more suitable to realize by software or hardware.

3.1. Add Objects

Connecting objects via ports will form a system model. A new object can add to the system. Fig.3 shows a new object *buffer* is added to producer/customer system. *Buffer* can store 10 products. Message is transferred via ports. The model is a producer/buffer/customer system. With the help of object conception and port, system can be extended easily and the net structure is simpler than previous one [2].

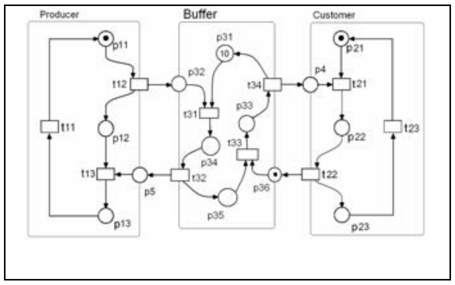

Fig. 1: Add an object

3.2. Disassemble Object

To decompose objects is a similar issue of Petri nets partition, which is a NP difficult problem. Many algorithms were studied to solve this problem and the work is still going on. In this section, attention is paid on general stratagem to partition Petri nets with sequence, parallel and loop structures.

3.2.1. Sequence Structure Partition

General speaking, OPN with sequence structure could be separate at any arc and form two object subnets, fig.4 (a). The places or transitions connected with the arcs will change into ports. But granularity should be considered while partitioning. Coarse granularity is anyhow better than fine granularity at system level models because it is easy to understand and describe. Fine granularity partition will produce too many object subnets, which needs a lot of ports for communication.
Although one place or transition can be regarded as an object, we do not suggest partitioning it into an object because an object should contain attribute and method.

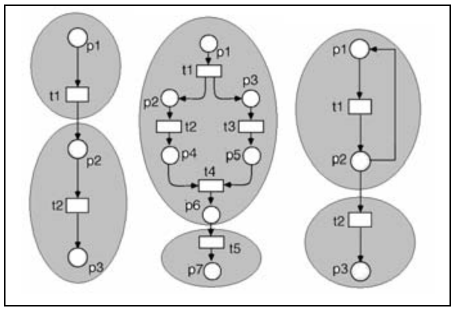

Fig. 1: OPN Partition

3.2.2. Parallel Structure Partition

Tasks assigned to hardware run in parallel on RC systems. System performance mainly depends on the parallel task partition. Therefore, Parallel tasks should be partition into the same object subnet so that the tasks could be mapping to hardware in next stage, see fig.4 (b), where t2, t3 are parallel tasks.

3.2.3. Loop Structure Partition

Loop tasks usually spend many CPU cycles. Therefore, they are assigned to hardware in RC systems and should be partitioned into one object, which will reduce the communication and resource consumption, see fig.4 (c).

4. Objects Realized by Software

Objects in OPN will be finally mapped to hardware or software of RC systems. Objects mapped to software finally realized by a segment of code, while objects mapped to hardware finally realized by logic circuits usually designed by hardware description language. Code will run on general processor of RC system and logic circuits will be completed by reconfigurable devices.

4.1. Objects Realized by Software

Objects related with control flow may be map to software. Objects containing few computing tasks could also be mapped to software. Many program languages are competent for designing objects. C++ is chosen because it is object oriented language and widely used. The object *buffer* in fig.3 is designed in C++ as below.

```
Class   Buffer
Public: p32=0, p36=1;
Private: p31=10, p33=0, p34=0, p35=0;
Method:
If(p32>=1)&(p31>=1)
   p34+1,p32-1,p31-1;
endif//t31
If(p34>=1)
   p34-1,p35+1,p5+1;
endif//t32
If(p35>=1)&(p36>=1)
   p33+1,p35-1,p36-1;endif//t33
If(p33>=1)
   p31+1,p4+1,p33-1;
endif//t34
```

4.2. Objects Realized by Hardware

Objects containing intensive computing tasks may be map to hardware in order to get high computing speed. Verilog hardware description language is used to design the hardware objects. An example of multiply-add unit for matrix multiplier is showed below. The corresponding OPN showed in fig. 5, where data places are represented by ellipses.

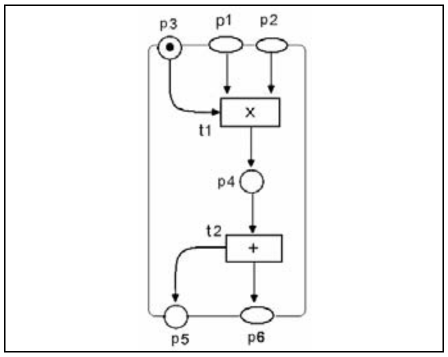

Fig. 1: OPN of multiply-add unit

```
module  M_Adder (p1,p2,p3, p5,p6 ) ;
input p1,p2,p3;output p5, p6;
reg     p4,p5, p6;
always   @ p3
begin
   p4 =p1*p2 ;
   p6 =p6+p4;
   p5 =1;
end
endmodule
```

5. Summary

A object oriented Petri net was proposed for describing RC systems, which is modular, reusable and have the ability to process data flows and mutual actions. The proposed OPN is practical to model RC systems because objects could be assemble and disassemble easily. A framework of describing RC systems by OPN model and mapping objects to hardware or software was proposed as well. To analyze the runtime and resource consumption of RC systems by OPN model will be studied in the future work.

6. References

[1] T.J. Todman, G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk and P.Y.K. Cheung. Reconfigurable computing: architectures and design methods. IEE Proc. Comput. Digit. Tech., Vol. 152, No. 2, March 2005, 193-207

[2] Y.K. Lee, S.J. Park, OPNets: an object-oriented high-level Petri net model for real-time system modeling, Journal of Systems Software 20 (99) (1993) 69– 86.

[3] C. Sibertin-Blane, R. Bastide, Object-oriented Structure for High Level Petri Nets, 11th Conference of Application and Theory of Petri Nets, Toulouse, France, 1990.

[4] K. Jensen, Coloured Petri nets: A high level for System Design and Analysis, in Advances in Petri nets 1990, Lecture Notes in Computer Science,Vol.2483,pp.343-416, Springer-Verlag, 1991.

[5] Hao Kegang, Open nets-model for mutual concurrent systems. J. of Northwest Univrsity. Vol.27, No.6, 1997.

[6] Chun-Che Huanga, Wen Yau Liang.Object-oriented development of the embedded system based on Petri-nets. Computer Standards & Interfaces 26 (2004) 187–203