# High Level Synthesis of Controllers for Communication Protocols

Asjad M. T. Khan Sadiq M. Sait Gerhard F. Beckhoff

King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia

**Abstract**

*Petri nets are popular in the communication protocol community for modelling and analysis purposes. This paper gives a procedure for implementation of controllers in hardware from a high level Petri net model. With this the entire design-analysis- implementation cycle for protocols can be Petri net based. It is especially suited for data link, network, transport layer Protocols.*

## 1 Introduction

In high level synthesis approach to design [1] the designer views the system from a high level of abstraction. This specification is then input to a design automation (DA) system which translates it to lower levels of representation (physical). Silicon compilers, for example, translate high level input specifications to layout specifications suitable for fabrication. In this paper a design automation system that accepts Petri net models of digital systems and produces VLSI layouts is proposed.

Petri nets can model concurrent/distributed, asynchronous systems. They can also handle interprocess communication explicitly and model at a high level of abstraction, the system level. They are backed by established theory and recently a number of tools have also come up. With the increasing quest for speed and availability of VLSI the trend towards concurrent systems is bound to grow. Petri nets are a good candidate for modelling these complex systems at a high level of abstraction. In high level synthesis acceptable solutions can be obtained by restricting the scope of the problem to a certain class. Controllers for communication protocols (data_link protocols for example), are considered as an instance of complex distributed and concurrent systems. The choice of protocols is supported by the fact that Petri nets have been used by the communication protocol specification and verification community for design and analysis purposes and direct implementation from the model in hardware is desirable. The usual approach has been to implement the protocol in software after it's Petri net model has been found to be correct. Work on implementation in software from the Petri net model exists. However, the increasing communication link speeds make hardware implementation desirable. The terms and notations used for Petri nets are taken from the survey-cum-tutorial by Murata [2].

## 2 Previous Work

A survey on the existing and proposed hardware implementations of protocols appeared in the literature [3]. Several implementations of the Data Link layer protocols and network layer exist. Attempts have also been made to implement the Transport layer protocols. However, most of the work is restricted to Data Link, Network and Transport layer protocols.

Petri nets provide a formal method for modelling asynchronous systems. Initial approaches to Hardware synthesis from Petri nets models were directed towards asynchronous implementation and one of the areas of application was design of fast control circuits for computers. However, no efficient method of implementation was found. Recently an automated high level synthesis system has been proposed [4]. However, they impose restrictions on the modelling which does not agree with the way protocols are being currently modelled. Synchronous implementations have also been attempted. Kwan et al. [5] and Auguin et al. [6] proposed methods for decomposition and implementation of a class of Petri nets by PLAs but their decomposition was heuristic. In this work synchronous implementation is considered. In an earlier work [7] preliminary results on the implementation of protocol controllers was considered. This work reports further progress.

## 3 Petri Net Model

Petri nets evolved as mechanisms for interprocess communication. They can model interaction between different components. It is this which makes Petri nets useful in protocol modelling. The Petri net model of protocols is derived in two steps [8], at first each entity and the channel is modelled as a state machine or a net and then the components are connected together by explicit interconnection mechanism corresponding to the actual implementation to give the global system model. A number of interconnection mechanisms have been proposed [8]. For simple models of communication medium the basic mechanisms are given below.

**Shared Place**: A place shared between the sender and the receiver processes represents the medium. It represents the fact that the message has been sent by the sender and not yet reached by the receiver. This is really the actual behavior denoting the fact that the message is in transit in the medium. It is the basic

0-7803-0593-0/92 $3.00 1992 IEEE

interconnection mechanism for processes belonging to distinct entities and has been most widely used [8]. A shared place representation implies potentially unbounded buffering, the sender is released after sending the message.

**Merged Transition**: In this technique the sender and the receiver processes share a transition to exchange a message. This implies no buffering, direct transfer and a synchronized send/receive.

## 4 VLSI Synthesis of Protocol Controller

The implementation is in the form of a PLA which is suitable for VLSI. This method can be automated to realize a Petri net based DA system. The previous work [7] assumed that the input to the DA system is a composite protocol model with the entities demarcated. Here, the general case, when only the composite model is known, is treated. The implementation method can be logically divided into two phases, the *Frontend*, it takes the Petri net model of the protocol and produces a FSM for each of the controller entities and the *Backend* which generates a PLA implementation of the FSM.

### 4.1 Frontend

This phase consists of two basic steps, decomposition of the protocol model into two entities representing the $N^{th}$ layer and for each entity, obtaining the FSM for it's implementation.

#### 4.1.1 Decomposition of Protocol

The decomposition of communication protocol composite model is a special case of the general decomposition of a Petri net into concurrent components [9]. The protocol is a distributed implementation of the service specification. Although each entity is a FSM (sequential), it will contain some events(local) which do not require any interaction with the peer processes and can occur concurrently with some events in the peer processes. The proposed method can be applied only if there are concurrent events in the composite model. The outline of the algorithm is stated next:

1. Given a Petri net, PN. It is assumed that it is live and bounded as the correctness of protocol demands these properties [10].

2. Find the reachability graph for the Petri net.

3. The existence of concurrent transitions can be detected by searching for n-cube patterns. An n-cube has the n transitions as its sides. It represents all the $n!$ possible firing sequences of length $n$ that can occur as paths between the endpoints of the internal diagonal and $2^n$ states lie on the vertices of the cube. If such a pattern occurs in the reachability graph then these n transitions are concurrent.

4. For the concurrent transitions build a simple, undirected graph, called concurrency graph, CoG, as:

   CoG=(V,E) where,

   | | |
   |---|---|
   | $V \subseteq T$, | is the set of nodes of the graph with each node representing a transition which is concurrent to some other transitions |
   | $E \subseteq V \times V$ | is the set of edges, with $e_{i,j} \in E$ iff node(transition) $i$ is concurrent with node(transition) $j$ |

5. Find if the graph is bipartite. If yes, then find partite components too. Else this procedure cannot be used. Each partite component belongs to an entity.

6. The transitions that do not occur in parallel with any other transition and were not featured in the concurrency graph, CoG are necessarily sequential and are related to the concurrent transitions by precedence relation. These can be assigned to the partite components which contains the related transitions. This resolves the partitioning problem.

7. The places that are common between the entities are the shared places and represent channels. Transitions of each subnet which were connected to the channel places are annotated with an input condition, an output, or both. The transitions can fire only when their input places have token and their input condition is true (the signal arrives from the sender or receiver). When a transition fires the output signal is produced (it is assumed to travel to the other process).

The above stated algorithm can be explained by the following points:

**Theorem 1** *A Petri net has n transitions in parallel iff for all reachable markings the transition firings manifest themselves as an n-cube with the n transitions as its sides.*

**Proof 1** Regarding Petri nets, with the condition/event model of system, places represent conditions and transitions denote events. The firing rule of Petri net allows only one transition to fire at a time. If two transitions, $t_1$ and $t_2$, are enabled at the same time, only one of them will fire. However there is no deterministic way to choose which one of these will fire. This restriction on firing will lead to both possible sequences, $t_1$ firing first and then $t_2$ firing or $t_2$ firing first and $t_1$ following, occurring in the reachability graph. This appears as a 2-cube pattern in the graph. This happens because with two transitions enabled and no deterministic choice on firing there is no ordering of the two events. Any of them may fire This is illustrated by a simple example shown in Figure 1. If in a Petri net, $n$ transitions are simultaneously

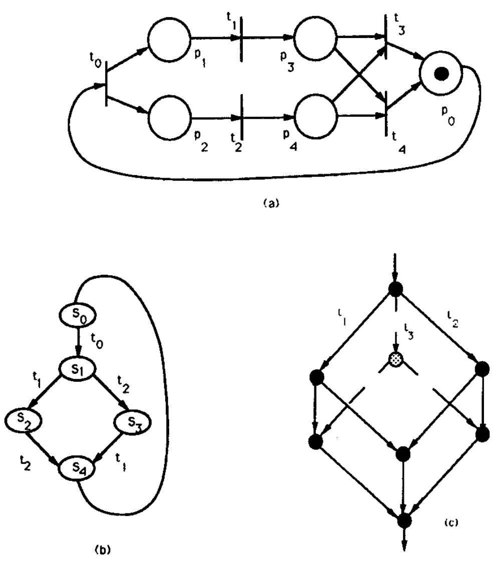

Figure 1: (a) A Petri net with concurrency (b) the reachability graph for the Petri net (c) A 3-cube structure for three concurrent transitions

enabled in a marking then as there is no ordering on their occurrence and all the possible permutations will occur in the reachability graph. For a set with $n$ elements, all possible permutations are $n!$, each of length $n$ without any repetition. In an n-cube, between the two endpoints of the internal diagonal there are $2^n$ points and $n!$ paths each of length $n$.

Once the concurrent transitions are detected, the concurrency graph, CoG, can be built. In general, the concurrent components can be found using the following two results:

**Theorem 2** The $\chi$(G)-coloring of a graph gives the minimum color classes which are independent sets.

The coloring problem is NP-Complete. However, for the special case of 2-party communication protocols, the number of components to be detected is two and the following result can be used.

**Theorem 3** If G is a 2 chromatic graph then necessarily G is bipartite and the color classes obtained by any 2-coloring of G are partite sets of the bipartite graphs.

Hence, the problem of finding the two entities reduces to finding the partite components of the CoG graph. Some of the transitions that do not occur in the CoG graph are necessarily sequential. The transitions with which they are sequential can be found from the reachability graph (or the incidence matrix) and they can be assigned to those partite sets which contain the transitions to which they are strongly linked sequentially. The two subsets then define a partition on the set of transition of the Petri net.

Once the transitions belonging to each entity are known, the two entities can be separated from the composite model by cutting the arcs connecting the transitions to shared places and assigning input/output to the transition to model communication with the other entity. This gives the annotated Petri net of each entity.

**Complexity of the Procedure:** The procedure presented has high computational complexity. The first step involves finding the reachability graph. For a Petri net, the reachability graph is an exhaustive procedure and requires space and time exponential in the number of places $|P|$.

The task of detection of n-cubes from the reachability graph can be done in polynomial time (a modified shortest path detection algorithm based on breadth first technique can be used) but the graph is exponential in $|P|$.

The detection of bipartite components can be done using a depth first search technique in polynomial time $O(V + E)$ where, $V$ is a subset of $T$ and $E$ is $T_1 \times T_2$, $T_1$ and $T_2$ being the transitions belonging to each entity.

### 4.1.2 FSM Extraction

The process of obtaining the FSM from the annotated Petri net has two steps: first, refinement of the entity and next generating a FSM for the underlying unannotated Petri net.

**Refinement:** As discussed in the section on modelling, each modelled entity contains some local actions which occur within the entity itself. These do not involve any communication with the peer but are related to the correct operation of the protocol (e. g. timer in a data link protocol). One way of handling refinement for local actions is to associate labels to transitions signifying local actions. In the protocol analysis stage these labels are ignored and no interpretation is associated, however, in the implementation stage these are interpreted as input conditions and outputs. Alternatively, this can be done at this stage (after entities have been separated).

**FSM from annotated Petri net:** The reachability graph of the underlying unannotated Petri net is isomorphic to an FSA which accepts the language of firing sequences if the graph is bounded [11]. The conditions and the output are then used to replace the transitions labelling the arcs of the transition. This then defines a FSM. The procedure of obtaining FSM from the annotated Petri net is summarized below:

**Procedure 1** 1. **for** each entity

2. find the reachability graph of underlying unannotated Petri net,RG=(V,E), where, nodes of the

graph, V=R($M_0$) and the set of labelled, directed arcs, E has an arc from $V'$ to $V''$ labelled by $t$ if for the corresponding markings, $M'$ and $M''$, $M'[t > M''$

3. **do** for the reachability graph
4. **for** every arc
5. substitute the transition labelling the transition, $t$ by the input conditions, $x, y, \ldots$ and outputs, $a, b, \ldots$ associated with the transition as, $< x, y, \ldots > /(a, b, \ldots)$, where, all the conditions $x, y, \ldots$ should be satisfied for the transition to occur and the change produces outputs $a, b, \ldots$. This is finite and isomorphous to a FSM.

#### 4.1.3 Frontend Implementation

The basic tool for implementation of frontend was P-NUT (Petri Net UTilities) developed at the University of California at Irvine.

### 4.2 Backend

The backend is basically a silicon compiler for FSM. It takes a description of the state machine to be implemented and gives the layout of the PLA implementing it. It is based on a subset of tools made at the University of California, Berkeley, called **1986 VLSI Tool Kit** and **VLSI Design Tools**. The entire process is automated. The first of the tools **meg** (Mealy Equation Generator) takes a high level input of the FSM and outputs the equations of the output and next_state variables in terms of present_state and input variables. This is then fed to **eqntott** which outputs the truth table for implementation by a two level circuit(PLA personality). This format is compatible with the **espresso** input which is used to minimize the function. The output can then be fed to a PLA generator, **mpla** which outputs the layout in the desired technology and style. What remains after that is to place the PLA in a pad frame, connect the input of FSM and the next state outputs, route the inputs and outputs to I/O ports and the chip would be ready to be sent for fabrication. These tasks can also be done using available tools. Finally, the FSM can be extracted from the PLA using circuit extractor **mextra** and simulated using **rnl** to verify correctness of operation.

## 5 Summary and Conclusions

A method for implementing communication protocols in hardware was presented. It was assumed that the composite Petri net model of the protocol is available. A procedure for automatically detecting the entities of the communication protocol was given. This procedure has exponential complexity (in number of places of Petri net) as it uses reachability graph. No restrictions have been put on the Petri net model. However, if certain restrictions are made on the model then the task is easier. The option of using a subclass has been investigated in [9]. The task of building an automation system requires that P- NUT, decomposition algorithms and VLSI DESIGN Tools be integrated.

## Acknowledgments

Thanks are due to Dr. T. Morgan for providing P-NUT, the University of Washington for the VLSI Design Tools, Dr. M. S. T. Benten for his help with the softwares and King Fahd University of Petroleum and Minerals, Dhahran for support.

## References

[1] M. C. Farland, A. C. Parker, and R. Composano, "The high level synthesis of digital systems," *Proceedings of the IEEE*, vol. 78, Feb. 1990.

[2] T. Murata, "Petri nets: properties, analysis and application," *Proceedings of the IEEE*, vol. 77, pp. 541–580, Apr. 1989.

[3] A. S. Krishnakumar and K. Sabnani, "VLSI implementations of communication protocols – a survey," *IEEE Journal on Selected Areas in Communication*, vol. 7, pp. 1082 –1090, Sep. 1989.

[4] T. H. . Y. Meng, R. W. Brodersen, and D. G. Messerschmitt, "Automatic synthesis of asynchronous circuits from high level specification," *IEEE Transactions on Computer Aided Design*, vol. 8, Nov. 1989.

[5] C. L. Kwan, P. L. Beux, and C. Michel, "The design of structured digital systems controlled by PLA," in *Microcomputer Architecture, Euromicro 1977*, (J. D. Nicoud, J.Wilmink, and R. Zaks, eds.), North Holland Publishing Company, 1977.

[6] M. Auguin, F. Boeri, and C. Andre, "Systematic method of realization of interpreted Petri nets," *Digital Processes*, vol. 6, no. 1, pp. 55–68, 1980.

[7] A. M. T. Khan, S. M. Sait, and G. F. Beckhoff, "VLSI implementation of controllers for communication protocols from their Petri net models," in *Second Great Lakes Symposium on VLSI*, 1992. Accepted for presentation.

[8] M. Diaz, "Modelling and analysis of communication and cooperation protocols using Petri net based models," *Computer Networks*, no. 6, pp. 419–441, 1982.

[9] A. M. T. Khan, *VLSI Implementation of a Class of Digital Systems from their Petri Net Models.* Master's thesis, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 1992. Expected January 1992.

[10] D. P. Sidhu, "Protocol design rules," in *Protocol Specification, Testing and Verification I*, (C. Sunshine, ed.), pp. 283–300, Elsevier Science Publishers B. V., North Holland, 1982.

[11] R. Valk, "Petri nets and regular languages," *Journal of Computer and System Sciences*, vol. 23, pp. 299 –325, 1981.