Proceedings of the 5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems, Hangzhou, China, April 16-18, 2006 (pp180-185)

An Extension of Self-Reconfigurable FSM with Reduced Reconfiguration Sequences Concept

DRISSA TRAORE MAO ZHI GANG
Center of Microelectronics,
School of Astronautics
Harbin Institute of Technology
CHINA

Abstract :In this paper we propose a novel concepts of self-reconfigurable finite state machines in which a delta transition is used for reconfiguration initialization and others reconfiguration sequences are exterior events, a new model of VLSI implementation to describe a certain number of state machines implemented in hardware that may be reconfigured during operation. Hardware is called self-reconfigurable finite state machine with reduced reconfiguration sequences if a delta transition is initialized by the Finite machine itself and some others reconfiguration sequences to update output function or transition function are initiated from exterior event. We propose an efficient concept to reduce the reconfiguration sequences which reduces the switching activities for low power and the reconfiguration can be sped up with a reconfiguration initialization clock. The boundary of feasibility when reconfiguring a given FSM specification into a new target one is presented. Experimental results with some FSM benchmarks show compression of reconfigurable sequences data with minimum loss of area compared to Markus machine.

Key Words: FSM, self-reconfigurable FSM, Delta transition, reconfiguration sequence, Markus machine [9].

1 Introduction

Finite state machine are important components in digital electronics such as in control circuit, network processing, testing etc… but reconfigurable finite state is still in its infancy. With reconfigurable hardware it is possible to change the functionality and wiring of hardware overtime [1…10]. Most of FPGA circuits are SRAM based and reconfiguration takes place by reconfiguring the actual configuration bit stream for logic function and switches over row or columns of an array of reconfigurable logic blocks (CLB). Reconfigurable finite state machine have been studied in some applications. In FPGA based design for control circuit the reconfigurable hardware approach is like virtual memory management approach and the control circuit is modeled in form of hierarchical finite state machine (HFSM) [8,10].

Ref [10] Implements reconfigurable finite state machine in FPGA, this is achieved by swapping pre allocated areas on a chip in partially dynamically reconfigurable FPGA or by reloading memory based cells in statically configured FPGA.Dynamic modifications to FSM behavior permit reutilization of FSM circuits during execution time, i.e. the same hardware can implement different functionality. Self-reconfigurable finite state machine use the Principe of the context of software. In software word, the first generation of microprocessors made use of the idea of self-reconfiguration program code step wise in order to execute a program that cannot fit into the existing memory. Indeed self reconfigurable FSM can be seen as circuit that reconfigure themselves gradually. The paper [9] propose the concept of self reconfigurable finite states machine with the ability to change output and transition function or both gradually during run operation and algorithm for the overhead of automatic reconfiguration of a given FSM. . In [12] we used the genial idea of reconfigurable finite state machine to build a VLSI dynamically hardware for finite state machine in which can be mapped some application such as tries when constrained by slow memory access and difficult to scale when running in processor.

The real application domain for self-reconfigurable finite state machine is communication system.

Basically, self-reconfigurable FSM are needed for application such autonomous sequential modules that can be considered as components of more complicated digital system. For example [9] examines a Mealy FSM that reads a sequence of bits and then detects two or more successive ones in the sequence. Reconfiguration considered in [9] permits the behavior of the FSM to be changed in such a way that it will detect two or more successive zeros in the sequence.
In this paper we propose a new model of self-reconfigurable Finite machine with reduced reconfigurable sequences, which does not ignore self-loop and the boundary of feasibility when migrating a FSM specifications to a new one.
This paper is organized as follows: In the next sections we review definition, example, and previous work on FSM and self reconfigurable FSM and propose our self-reconfigurable concept, finally we conclude with a hardware implementation.

2 Definitions

A finite state machine is defined in the standard way as a tupple $M=(\varepsilon, A, Q, q_0, \delta, \lambda)$ ε is a finite set of input symbols, $A \neq \varnothing$ is a finite set of states, $q_0 \in Q$ is a reset state, $\delta(q,a): Q \times \varepsilon \rightarrow Q$ is the transition function, and $\lambda(q,a): Q \times \varepsilon \rightarrow A$ is the output symbol In the specification of the state transition graph of the FSM each state correspond to one node in the graph, and there exists an edge e_{ij} between two states $q_i and q_j$ with label a/b If $\delta(q_i,a)=q_j$ and $\lambda(q,a)=b$.
The specifications of FSM are based on state transition graphs (STG) or state table. Fig.1 show a state table of one finite state machine.

3 Previous work

3.1 Self-reconfigurable FSM theories

A reconfigurable FSM is a 10 tupple $(I, O, S, S_0, F, G, H_f, H_g, H_i, R)$ in which:
- S is a finite set of internal states of the finite state machine
- I is a finite set of input states of the finites of the FSM, which either are symbolic or are represented as a binary vector of values of its input signals
-O is a finite set of output states of the FSM, which are either symbolic or are represented as binary vector of values of its output signals.

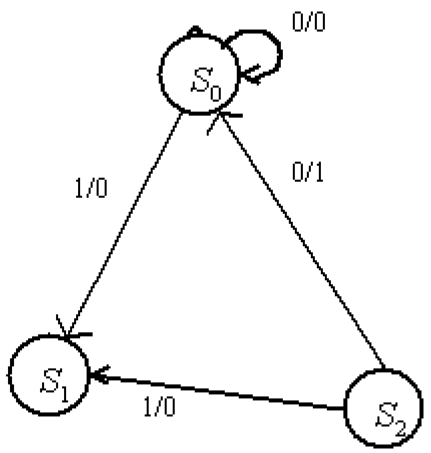


Fig.1 Example of STG of a FSM

- $F(i,s)$ Is a relation from the (input state, present state) pairs, also called total states, to the next state $(i.e..F \subseteq I \times S \times S)$.
- $G(i,s)$ Is a relation from the input (input state, present state) pairs to the output states $(i,e..G \subseteq I \times S \times O)$.
- $S_0 \subseteq S$ Is a set of initial (or reset) states?
-R is a finite set of reconfiguration states of the FSM, which either are symbolic or are represented as a binary vector of values of its reconfiguration input signals.
- The transition reconfiguration function $H_f(r), r \in R$ is mapping from the reconfiguration state to the total state $F(i',s), i' \in I, s \in S, i.e..F(i',s)$ may reconfigured by an update as follows:
$F(i',s) := H_f(r)$.
-The output reconfiguration function $H_g(r), r \in R$ is a mapping from the reconfiguration state to the output state $G(i',s), i' \in I, s \in S, i.e..\ G(i',s)$ may be reconfigured by an update as follows:
$G(i',s) := H_g(r)$.

-$H_i\left(i,r\right), i' \in I, r \in R$ is a mapping from the (input state ,reconfiguration state) tupple to the internal input state $i' \in I$.
The concept is shown in the schematic of Fig.2 .in this schematic the transition function F may be updated by H_f and the output function G may be updated by H_g. H_f and Hg depend on the reconfiguration state, whereas H_i depends on the input state I and the reconfiguration state R. The function H_i is defined such that during normal operation of the finite state machine, I' = I and during the reconfiguration process, I' is depending on r only.

3.2 Reconfiguration Program

Given an FSM M and an FSM M'. It is always possible to specify a finite sequence of reconfiguration transition in order to transform M into M'.

Consider a given state machine M we want to reconfigure into a target state machine M'. First we need to know what transitions have to be reconfigured when migrating M to M'.

Given a given finite state machine M= $\left(I, O, S, S_0, F, G\right)$ and a target finite state machine M'=$\left(I', O', S', S'_0, F', G'\right)$
.Let
T'=$\{(i, s_x, s_y, o) : i \in I', s_x \in S, s_y = F\left(i, s_x\right),$
$o = G'\left(I, S_x\right)\}$
denote the total set of transition of M', then a transition $t_d = (i, s_x, s_y, o) \in T$ is called delta transition and needs to be reconfigured in order to mutate M into M' if at least one of the following conditions hold:

$-i \notin I$ Or $s_x \notin S$ or $s_y \notin S$, or $o \notin O$ or
$s_y \neq F\left(i, s_x\right) \wedge i \in \left(I \cap I'\right) \wedge s_x \in \left(S \cap S'\right)$, Or
$o \neq G\left(i, s_x\right) \wedge i \in \left(I \cap I'\right) \wedge s_x \in \left(S \cap S'\right)$.

Let $T_d \subseteq T'$ be the set of all delta transition

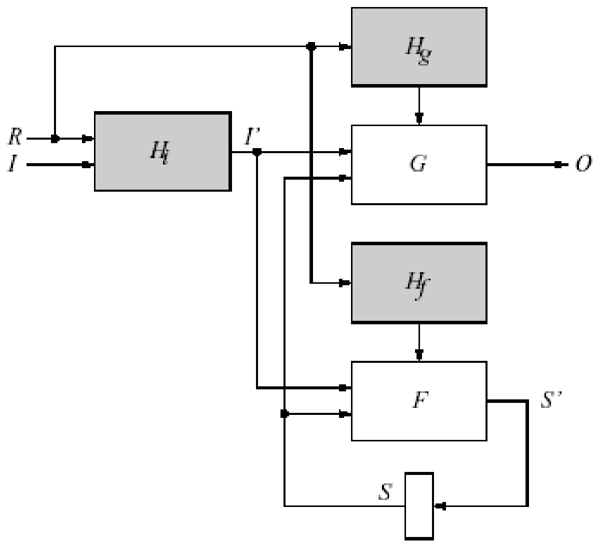


Fig.2 Schematic of reconfigurable FSM

4 Self reconfiguration FSM with reduces reconfiguration sequences concept

4.1 Concepts and Example

A self reconfigurable FSM D_0 is a 9 tupple: $(\varepsilon, R, A, Q, q_0, \delta, \lambda, M, C, E_E)$ In which:
* $(\varepsilon, A, Q, q_0, \delta, \lambda)$ describes an FSM according to finite state machine.
* R is a set of reconfiguration sequences called reconfigurator and generates the vectors R_λ, R_δ, I_R during reconfiguration initialization and $R_{\lambda I}, R_{\delta I}$ during exterior reconfiguration event to update the output function and transition function. And the output of the reconfiguration function is a mapping from the current state and the C during initialization and mapping from E_E during exterior event.
* $R_{\delta I}(i) := R_{\lambda I}(i) := I_R(i) := R(S_i, C)$ During initialization
$R_\delta(i) := R_\lambda(i) := R(E_E)$ During exterior event
*C is a delta transition; it determines toward which delta transition the machine has to traverse for initialization, it may be an exterior event.
* E_E is a set of transitions, it determines toward which transition the machine has to traverse during exterior event, it may be an exterior event.

* $M(I_R, \varepsilon, E_E)$ is a mapping from the input State, reconfiguration state I_R to the internal input I' during initialization and a mapping from the input state. Reconfiguration state E_E to the internal input I'. The concept is shown in the block diagram of Fig.3 In this model, during initialization process the transition function is updated by R_δ and the output function is updated by R_λ. R_λ And R_δ depend on the current state, whereas M depends on the input state ε and the reconfigurator input I_R. During exterior reconfiguration process the transition function and output function are updated respectively by $R_{\lambda I}$ and $R_{\delta I}$ depending on E_E. The function M can be a multiplexer and is defined such that during normal operation of the hardware $I' = \varepsilon$ and during initializations reconfiguration process I' depend on I_R only and depend on E_E in the exterior reconfiguration process. Notice the difference between our concept of self reconfigurable FSM and the self-reconfigurable theories in previous work [9] where all the reconfiguration sequences are generated by exterior events. In our case some sequences are generated by the current state of the machine itself.

Example 4.1 considers two FSMs M and M' with their state transition diagram show in Fig.4 and Fig.5. Suppose that the machine is in M and we want to migrate to M' .Let us assume the machine is in state S_0 and D = { D_1, D_2 ,D_3} is the set of delta transitions when a reconfiguration sequence takes place for each delta transition. D_1 = D_2 $=(01, S_1, S_0, 1)\}$, $(00, S_1, S_2, 1)$, $(10, S_2, S_3, 0)$, $(01, S_3, S_0, 1)$ } D_3 = $\{(11, S_0, S_1, 0)$, $(01, S_1, S_0, 1)\}$. We can see that D_1 and D_2 have the same reconfiguration program and also the longest so we can attribute these delta transitions for initialization and D_3 as exterior reconfiguration event. We can gain four reconfiguration sequences. For this case the reconfiguration sequences is reduced by about 60%.

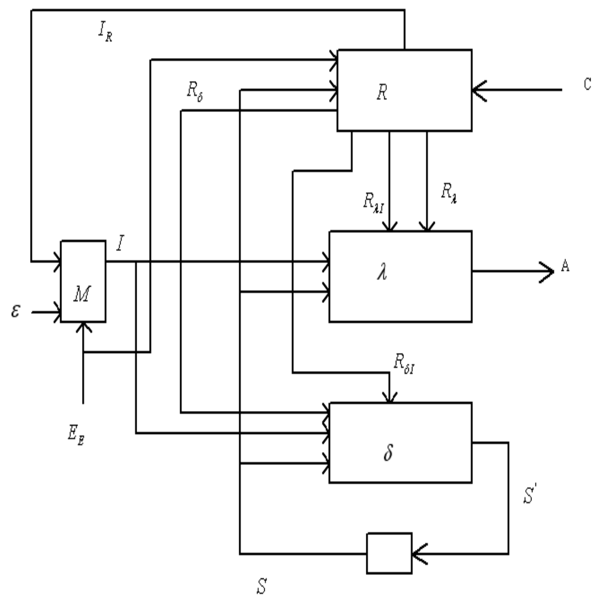
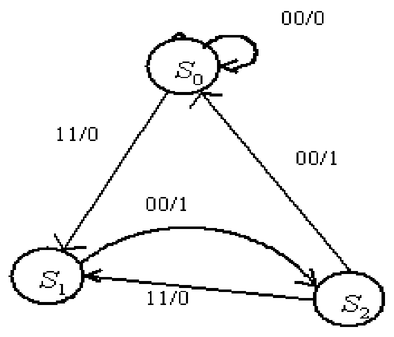
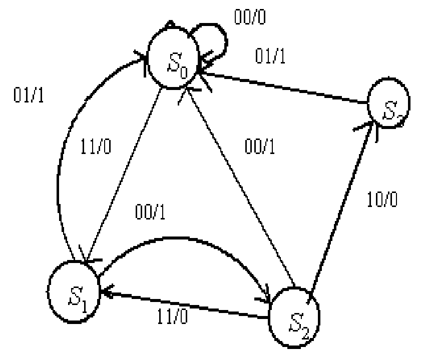


Fig .3 schematic of reconfigurable FSM with reduced sequences



M

Fig.4 STG of M



M'

Fig.5 STGs of M'

4.2 Definitions And Theorem

4.2.1 Definition

Given a finite state machine M and a finite state machine M' and a set of states including the states of M and M', a set of inputs including the inputs of M and M', a set of outputs including the outputs of M and M', the problem of feasible self-reconfigurable FSM with reduced reconfiguration denotes the decision problem whether the delta transition used for initialization will not be traversed with exterior reconfiguration event.

4.2.2 Theorem

According to definition 4.2.1 given the specifications of a state machine M and a state M' it is always possible to migrate from M into M' with reconfiguration initialization reducing the reconfiguration sequence if the delta transition used for initialization is not traversed during the exterior reconfiguration event.

5 Experimental results

In order to simulate our concepts for area, we have synthesized self-reconfigurable hardware with dk27, mc, bbtas benchmark FSMs example with the different concepts. For synthesis we have used synopsys tolls, the optimizations was done under same conditions and same constraints. The experimental results in the following tables show compression of reconfigurable sequences data with minimum loss of area compared to Markus machine.

FSM	sequences	Reconfigurable data	Reconfigurator cells
bbtas to dk27	24 sequences	120 bits	70 cells
dk27 to bbtas	27	135 bits	87
bbtas to mc	17	85 bits	68
mc to bbtas	25	125	81
dk27 to mc	16	80	68
mc to dk27	21	105	73

Table .1 Markus machine

FSM	sequences	Reconfigurable data	Reconfigurator cells
bbtas to dk27	18 sequences	90bits	85 cells
dk27 to bbtas	21	110	96
bbtas to mc	14	56	77
mc to bbtas	20	100	90
dk27 to mc	13	39	74
mc to dk27	14	42	73

Table.2 Reconfigurations with initialization

6 Conclusion

.

In this paper, we introduced the concept of self reconfigurable FSM with reduced reconfiguration sequence. Contrary to self-reconfiguration FSM theories developed in [9] where all the reconfiguration sequences are generated by the exterior, a delta transition is used for initialization, which can reduce the switching activities for low power and speed up the reconfiguration. We have also presented the boundary of feasible self-reconfigurable FSM with reduced reconfiguration sequences. Future work will be the automatic generation of the delta transition for initialization

References:

[1] Venla Tapatho N. Rayapati and Bozena K Aminska 'dynamic reconfigurable schemes for megabit BiCMOS SRAMs and performance evaluation" *microelectronic reliab vol 37 No 5pp*

[2] Yukio MITSUYAMA, Zaldy andales, tAKAO onoye, and Isao SHIRAKAWA"VLSI 1mplementation of dynamically reconfigurable hardware –based cryptosystem"*2000 symposium on VLSI circuit digest of technical papers.*

[3] Jurgen Becker, THILO Pionteck, Christian Haberman, Manfred Glesner" design and implementation of a coarse grained dynamically reconfigurable Hardware architecture" *2001 IEEE*

[4] H.ITO R.Konishi, H.Nakada K.Oguri A.Nagoya "Dynamically reconfigurable Logic LSI- PCA-

1*"2001 symposium on VLSI Circuits Digest of technical papers.*

[5] Sayfe Kiaei, Jaisimha K.Durgam " VLSI design of dynamically reconfigurable ARRAY"*1989 IEEE.*

[6] Lizy Kurian John, Eugence John" a dynamically reconfigurable interconnects for array processors."*IEEE Transaction on VLSI system 1998.*

[7] T.Fujii et al " a dynamically reconfigurable logic engine with a multicontex/multi –mode unified cell architecture," *in ISSCC Digest of technical papers, pp364-365, Feb 1999.*

[8] V.Sklyarov " reconfigurable models of finite state machines and their implementation in FPGAs"*journal of systems architectures 47(2002) 1043-1064*

[9] Markus Koster, Jurgen Teich" self reconfiguration finite state machine theory and implementation " *proc of the 2002 design, automation and test, Europe conference and exhibition IEEE.*

[10] Armaldo Oliveiro Valery Sklyarov" implementation of virtual control circuit in dynamically reconfigurable FPGAs' *IEEE*

[11] R.Shelar, H.Naraya, M.P Desai"decomposition of finite state for area, delay minimization" *IEEE ICCD99 Austin 10-oct99 pp620-625.*

[12] D.Traore, M.Z.Gang " VLSI dynamically reconfigurable hardware for finite state machine design and analysis" *6 Th int on ASIC, Shanghai China Oct 24-27.*ISBN 0-7803-9210-8.IEEE, pp 734-738