A Highly Efficient Method for Extracting FSMs from Flattened Gate-Level Netlist

Yiqiong Shi, Chan Wai Ting, Bah-Hwee Gwee and Ye Ren
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
{*yqshi, cwting, ebhgwee, renye*}@ntu.edu.sg

***Abstract*— This paper proposes a novel method for extracting Finite State Machines (FSMs) from flattened gate-level netlist. The proposed method which employs a potential state register elimination technique and a two-level FSM separation strategy is highly applicable to control-intensive circuits. The potential state register elimination technique is based on control signal identification whereas the two-level FSM separation strategy is based on enable tree identification and the strongly connected components algorithm. To demonstrate the efficacy and to illustrate the unique features of the proposed FSM extraction method, the Synopsys DesignWare DW8051 microcontroller is used as the benchmark circuit for comparison and simulations. Results show that the proposed method reduces the complexity of the extracted FSMs in terms of number of state registers in an FSM by more than 90% as compared to the reported technique.**

I. Introduction

Highly complex multimillion-gate state-of-the-art ICs are challenging to computer-aided design (CAD) tools and inevitably call for the 'divide and conquer' strategy. As part of the design flow, specifically for formal verification [1] and for high-level design-for-testability (DFT) [2], it is imperative that modern CAD tools are able to comprehensively extract high-level modules (of a circuit) from its low level description. In other IC-design related processes/issues, transforming a detailed design description into successively higher levels of abstraction is also used in copyright infringement investigation and competitive analysis [3].

One important aspect in the field of circuit synthesis and analysis is the modeling and understanding of finite state machines (FSMs) [4]. An FSM is a generalized type of sequential logic in which the next state of the machine can be written as a function of the current state and the value of the input signals to the machine. FSMs are commonly employed as controllers in digital circuits, and hence, have a tremendous impact on the behavior and performance of the circuits, especially for control-intensive circuits such as microcontrollers.

The methods and algorithms for the extraction of FSMs reported in the literature have been focused on extracting FSMs from hardware description language (HDL) [5], [6], which are useful for the synthesis of digital circuits, but not for function understanding and verification. The only current-art technique for extracting FSMs from the flattened gate-level netlist of a circuit in the literature, to the best of the authors' knowledge, is reported in [7], but with a number of shortcomings when it is applied to control-intensive circuits. Firstly, when a circuit with multiple FSMs is being analyzed, this reported technique is unable to separate the different FSMs. The separation of the FSMs is crucial for the understanding of the overall structure of the circuits because an FSM with a large number of state registers, and hence states, is practically impossible to analyze and understand. Secondly, this reported technique does not provide any concrete method for differentiating an FSM from an accumulator or similar arithmetic logic blocks. Both types of modules have feedback paths connecting registers' output back to input, but serve distinctive purposes in a digital circuit.

In view of the abovementioned shortcomings of the current-art technique for extracting FSMs from the flattened gate-level netlist of a digital circuit, this paper proposes a new method that employs a systematic approach to overcome the shortcomings and is applicable to not only data-flow intensive circuits but also control-intensive circuits. The novel features of the proposed method include a potential state register elimination technique based on control signal identification and a two-level FSM separation strategy based on enable tree identification and the strongly connected components algorithm [8]. The Synopsys DesignWare DW8051 microcontroller, a high-performance and configurable 8051 core, is synthesized and flattened using Synopsys Design Compiler and used as the benchmark circuit for illustrating the various features as well as to demonstrate the efficacy of the proposed method. Results show that the proposed method reduces the complexity of the extracted FSMs in terms of the number of state registers in an FSM by more than 90% as compared to the reported technique.

This paper is organized as follows. Section II describes the overall workflow of the proposed method for extracting FSMs from flattened gate-level netlist, emphasizing on the novel features. Section III discusses the results of applying the proposed method on the synthesized flattened DW8051 microcontroller, and Section IV concludes the paper.

II. FSM Extraction Method

The proposed method for extracting FSMs from flattened gate-level netlist is depicted in Fig. 1, and the various Algos are summarized below.

- Algo 1: to identify potential state registers by identifying registers whose output returns to its input through a series of combinational circuits;

978-1-4244-5309-2/10/$26.00 ©2010 IEEE

- Algo 2: to identify actual state registers from the identified potential state registers;
- Algo 3: to group different registers according to their enable signals and generate register groups;
- Algo 4: to group state registers according to the identified register groups;
- Algo 5: to identify the combinational circuits forming the feedback paths (loops) of the state registers; and
- Algo 6: to further partition state registers based on the strongly connected components.

Of all the Algos, Algo 1 and 5 are similar to the reported technique in the literature, while the rest are unique to the proposed method and will be discussed in detail in the subsequent sections.

A. Potential State Register Identification and Elimination (Algo 1 & 2)

Algo 1 in Fig. 1 identifies potential state registers using the reported technique [7] of identifying registers whose output feeds back to its input through a series of combinational circuits.

However, not all registers with feedback paths are deemed to be state registers. For example, arithmetic logic blocks such as accumulators use the output of its registers as input to compute the new values, and could feature similar structure as FSMs but having different functions.

The principle of the proposed Potential State Register Elimination algorithm (Algo 2 in Fig. 1) is on the basis that FSMs are commonly employed as controllers in digital circuits, i.e. to control certain parts of the circuits. In other words, the output of the FSMs will most likely lead to some control signals of the circuits (e.g. registers' enable port and multiplexers' select port). Consequently, the proposed Potential State Register Elimination algorithm examines each of the potential state register identified by Algo 1 to determine whether it is connected to one or more control signals through a series of combinational circuits. A potential state register whose output is connected to control signals is identified as an actual state register, whereas the others are eliminated.

B. State Register Grouping Based on Enable Tree Identification (Algo 3 & 4)

The input to the Enable Tree Identification algorithm (Algo 3 in Fig. 1) is the flattened gate-level netlist, and the output is a collection of register groups, where each register group has a unique enable signal. In other words, registers controlled by the same enable signal are grouped together. The remaining ungrouped registers and registers without enable port are also collectively grouped as 'solos'.

The principle of identifying enable trees and group registers according to their enable signals is on the basis that registers controlled by the same enable signal are generally more closely related, e.g. representing various bits belonging to the same data word. In the context of FSM extraction, it is highly unlikely that registers controlled by different enable signals belong to the same FSM. Hence, the identified register groups

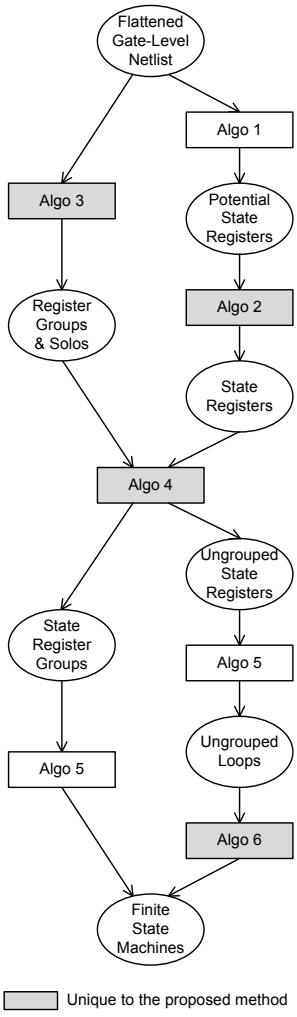


Fig. 1: Flow graph of the proposed FSM extraction method.

are used by Algo 4 to separate state registers belonging to different FSMs.

The Enable Tree Identification algorithm is depicted in Fig. 2(a). The flattened gate-level netlist is first converted to di-graph matrix such that graph theory and associated algorithms can be applied. All connections other than input to buffers, inverters and enable ports of registers are then disconnected. In other words, the original di-graph matrix is modified by setting all entries other than those representing input to buffers, inverters and enable ports of registers to zero. When connected components algorithm is applied to the modified di-graph matrix, any ensuing components with more than one register will have all its registers connected to each other through the enable ports, i.e. connected to the same enable signal, with or without buffer/inverter chain.

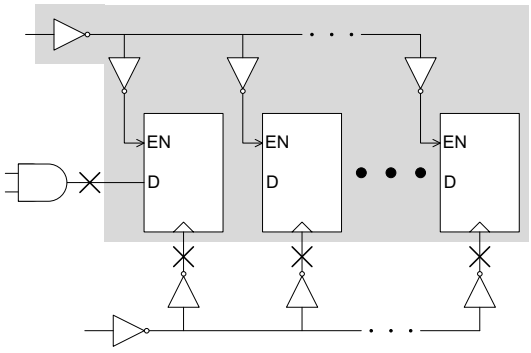
Fig. 2(b) illustrates the Enable Tree Identification algorithm in Fig. 2(a) with an example. It can be seen that since all

connections other than input to buffers, inverters and enable ports of registers are disconnected (depicted by crosses in Fig. 2(b)), only the shaded portion of the circuit will be included as part of the enable tree, i.e. the AND gate and the inverters driving the clock ports of the registers will be excluded. The registers will then be grouped since they are controlled by the same enable signal.

Algo 4 intersects the identified state registers from Algo 2 with the identified register groups from Algo 3 to generate state register groups as well as the remaining ungrouped state registers. In summary, Algo 3 and 4 separate state registers belonging to different FSMs based on their enable signals. The remaining ungrouped state registers will be further partitioned by the following steps (see below).

MAP netlist $\rightarrow$ di-graph matrix $\mathbf{A}$;
$a_{i,j} = 0$ if and only if
$j \notin$ {buffer, inverter, enable port of register}
$\mathbf{CC} = Connected.Components(\mathbf{A})$;
FOR $i = 1 \rightarrow ||\mathbf{CC}||$
  IF $\exists \mathbf{R} \subseteq \mathbf{CC_i}$
    && $||\mathbf{R}|| > 1$
    PUSH $\mathbf{CC_i} \rightarrow \mathbf{GROUP}$
  ELSEIF $\exists \mathbf{R} \subseteq \mathbf{CC_i}$
    && $||\mathbf{R}|| == 1$
    PUSH $\mathbf{R} \rightarrow \mathbf{SOLO}$
  ELSE DISCARD $\mathbf{CC_i}$
  ENDIF
ENDFOR

(a)



(b)

Fig. 2: (a) Proposed Enable Tree Identification algorithm, and (b) an example.

C. Combinational Logic Identification and Loop Grouping (Algo 5 & 6)

Algo 5 in Fig. 1 identifies the combinational logic belonging to the FSMs using the reported technique [7] of identifying the feedback paths (loops) of the state registers.

To further partition the ungrouped state registers, the Loop Grouping algorithm (Aglo 6 in Fig. 1) applies the strongly connected components algorithm [8] on the identified loops of the ungrouped state registers ('Ungrouped Loops' in Fig. 1) from Algo 5. If the loops of two state registers intersect, i.e. the next states of two state registers are affected by the same combinational circuit(s), the two state registers are grouped together as the state registers of the same FSM.

Fig. 3 illustrates the proposed Loop Grouping algorithm with an example. It can be seen that R1 has a feedback path formed by G1 and G2, while R2 has a feedback path formed by G2 and G3. Since the two loops share a common gate, G2, they will be identified as a strongly connected component. Hence, the Loop Grouping algorithm will group them together as part of the same FSM.

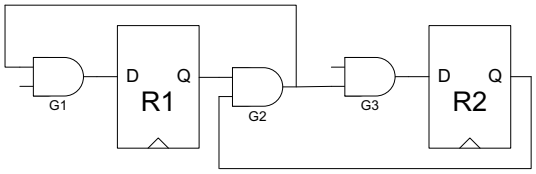


Fig. 3: An example illustrating the proposed Loop Grouping algorithm.

The grouped loops identified by the Loop Grouping algorithm are combined with the grouped loops generated by Algo 5 based on state register groups identified by Algo 4, and all the FSMs are identified.

III. Extracting FSMs from DW8051 Macrocell

The proposed method for extracting FSMs from flattened gate-level netlist was applied to the Synopsys DesignWare DW8051 macrocell, a high-performance and configurable 8051 core. The DW8051 macrocell was synthesized and flattened out using Synopsys Design Compiler. The flattened gate-level netlist contains a total number of 5330 instances including 48 IO pads and 739 registers.

Table I depicts the efficacy of the proposed method in reducing the number of state registers in an FSM, i.e. in reducing the complexity of the extracted FSMs. It can be seen that without any of the proposed techniques, i.e. based on reported technique [7], a total number of 154 state registers were identified, forming a huge FSM with 2^{154} states. Using the proposed Potential State Register Elimination algorithm, 36 registers were eliminated, resulting in a total number of 118 state registers. The percentage reduction in the number of state registers in an FSM with respect to [7] is 23.4%.

The proposed state register grouping technique based on the proposed Enable Tree Identification algorithm identified 16 state register groups, with the largest state register group containing 32 state registers. This is shown as the worst case in Table I, corresponding to a percentage reduction of 79.2% with respect to the 154 state registers identified based on the reported technique [7]. The average number of state registers in an FSM after Enable Tree Identification is 7.4, corresponding to a percentage reduction of 95.2%.

The proposed Loop Grouping algorithm further partitioned the state registers, resulting in a total number of 32 FSMs. The

TABLE I: Efficacy of the Proposed Techniques in Reducing the Number of State Registers in a State Machine

		No. of State Registers in a State Machine	Percentage Reduction w.r.t Reported Technique
Reported Technique [7]		154	-
With Proposed Potential State Register Elimination		118	23.4%
With Proposed Enable Tree Identification	Worst Case	32	79.2%
	Average	7.4	95.2%
With Proposed Loop Grouping	Worst Case	9	94.2%
	Average	3.7	97.6%

largest FSM contains 9 register (worst case), corresponding to a percentage reduction of 94.2% with respect to [7]. The average number of state registers in an FSM after Loop Grouping is 3.7, corresponding to a percentage reduction of 97.6%.

In summary, the proposed method effectively identified and separated a total number of 32 FSMs from the flattened gate-level netlist of the DW8051 macrocell, and an example is depicted in Fig. 4. All the identified FSMs are of reasonable size and much smaller than the FSM identified based on the reported technique, and hence, are much easier to analyze, model and understand.

The runtime of the various Algos in Fig. 1 is depicted in Table II, where the Algos unique to the proposed method are highlighted. It can be seen from Table II that the total time taken for the proposed method to extract FSMs from flattened gate-level netlist of the DW8051 is 9.169 seconds, of which 2.32 seconds are taken by the proposed techniques for potential state register elimination and FSM separation (Algo 2, 3, 4 and 6). In other words, the runtime overhead of the proposed method compared to the reported technique is 33.9% for this case study.

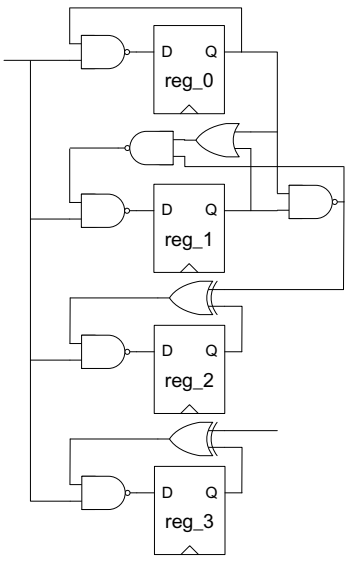


Fig. 4: An FSM extracted from the flattened gate-level netlist of the DW8051 microcontroller.

TABLE II: Runtime of Various Algos in Fig. 1

Algo	1	**2**	**3**	**4**	5	**6**
Runtime (sec)	4.546	**1.104**	**1.193**	**0.019**	2.303	**0.004**

IV. Conclusion

This paper presents a novel method of extracting FSMs from flattened gate-level netlist, which employs a potential state register elimination technique based on control signal identification and a two-level FSM separation strategy based on enable tree identification and the strongly connected components algorithm. The proposed FSM extraction method was verified on synthesized and flattened benchmarking DW8051 microcontroller netlist and the result demonstrates the efficacy of the proposed method, in particular when applied to control-intensive circuits. The future work will involve the identification of the State Graph or State Table from the extracted netlist for each FSM to understand the functionality of the state machines as well as identify any error in the extraction of the FSMs caused by the underlying assumptions, e.g. the rarity of registers controlled by different control signals being part of the same FSM.

References

[1] S. Novakovsky, S. Shyman, and Z. Hanna, "High capacity and automatic functional extraction tool for industrial VLSI circuit designs," in *Proc. 2002 IEEE/ACM Int. Conf. CAD*, Nov. 2002, pp. 520–525.

[2] I. Parulkar, M. A. Breuer, and C. A. Njinda, "Extraction of a high-level structural representation from circuit descriptions with applications to DFT/BIST," in *Proc. 31st DAC*, June 1994, pp. 345–350.

[3] J. Kumagai, "Chip detectives," *IEEE Spectrum*, vol. 37, no. 11, pp. 43–48, Nov. 2000.

[4] L. Yuan, G. Qu, T. Villa, and A. S.-Vincentelli, "An FSM reengineering approach to sequential circuit synthesis by state splitting," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1159–1164, June 2008.

[5] J.-C. Giomi, "Method of extracting implicit sequential behavior from hardware description languages," US Patent 5,774,370, Tech. Rep., June 1998.

[6] M. E. Gilford, G. N. Walker, J. L. Tredinnick, M. W. Dane, and M. Reynolds, "Recognition of a state machine in high-level integrated circuit description language code," US Patent 6,675,359, Tech. Rep., Jan. 2004.

[7] K. S. McElvain, "Methods and apparatuses for automatic extraction of finite state machines," US Patent 6,182,268, Tech. Rep., Jan. 2001.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.