Concurrent Error Detection in Asynchronous Burst-Mode Controllers

Sobeeh Almukhaizim* and Yiorgos Makris
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

Abstract

We discuss the problem of Concurrent Error Detection (CED) in a popular class of asynchronous controllers, namely Burst-Mode machines. We first outline the particularities of these clock-less circuits, including the use of redundancy to ensure hazard-free operation, and we explain how they limit the applicability and effectiveness of traditional CED methods, such as duplication. We then demonstrate how duplication can be enhanced to resolve these limitations through additional hardware for comparison synchronization and detection of error-induced hazards, which jeopardize the interaction of the circuit with its environment. Finally, we propose a Transition-Triggered CED method which employs a transition prediction function to eliminate the need for hazard detection circuitry and hazard-free implementation of the duplicate. As indicated by experimental results, the proposed method reduces significantly the cost of CED, with an average of 22% in hardware savings.

1 Introduction

Asynchronous circuits promise a wide range of benefits, including elimination of clock distribution networks and clock skew problems, improved performance, reduced power consumption, and modularity. Nevertheless, adoption of a fully asynchronous design style for general purpose circuits has been rather limited, mainly because of the lack of supporting CAD tools and methodologies. Indeed, asynchronous circuits present their own set of challenges, making the porting of design and test methods from the synchronous domain neither straightforward nor always possible. In certain control-dominated applications, however, the use of asynchronous circuits has resulted in irrefutable advantages. For example, an asynchronous implementation of an instruction decoder exhibits a performance that is at least three times better than the performance of a highly tuned synchronous version [1]. As a result, asynchronous controllers have attracted a lot of attention and several styles have been proposed for their design [2, 3, 4]. Among them, *Burst-Mode machines* constitute one of the most popular classes.

In this paper, we address the problem of Concurrent Error Detection (CED) in asynchronous Burst-Mode machines. CED methods are typically employed to monitor the behavior of a circuit and detect any deviation from the correct functionality due to transient errors such as Single Event Upsets (SEU). While a wide variety of CED methods have been developed for synchronous controllers, their asynchronous counterparts are intrinsically different, limiting the effectiveness of these methods. To demonstrate this problem, we discuss the applicability of duplication, the most common synchronous CED method, to asynchronous Burst-Mode controllers. We show that direct use of duplication is jeopardized by two inherent properties of these circuits:

- **Lack of a global clock:** Clock-less operation allows a circuit and its duplicate to produce results autonomously and at their own pace. As a result, even in error-free operation, the outputs of these circuits are not always equal. Therefore, in order to avoid false alarms, a comparison synchronization method is required.
- **Existence of redundant logic:** Redundancy in the implementation of the circuit is necessary to ensure hazard-free operation, as required by the communication protocol between a Burst-Mode machine and its environment. As a result of redundancy, some errors cause **only** hazards but no functional discrepancy, so they cannot be detected by comparison. Therefore, in order to monitor the correct interaction of the circuit and its environment, a hazard-detection method is also required.

In short, remedial action in the form of additional hardware needs to be taken. To address the first issue, we propose a comparison synchronization method which utilizes control information inherent to the operation of asynchronous Burst-Mode controllers. To address the second issue, we propose the addition of hazard detection circuitry to the output and state bits of the original circuit. Thus, duplication-based CED is enhanced to guarantee detection of all functional errors and hazards in asynchronous Burst-Mode machines.

We then propose a *Transition-Triggered* CED method that reduces the overhead incurred by the enhanced duplication described above. More specifically, our method uses a transition prediction function which is derived from the functionality of the asynchronous Burst-Mode machine. In conjunction with the comparison synchronizer, this function eliminates the need for a hazard-free duplicate and is used as a less expensive method to perform hazard detection.

The few asynchronous CED methods that exist in the literature assume the existence of *explicit* completion signals in order to synchronize the comparison [5, 6, 7]. In contrast, the

*The author is supported through a scholarship from Kuwait University.

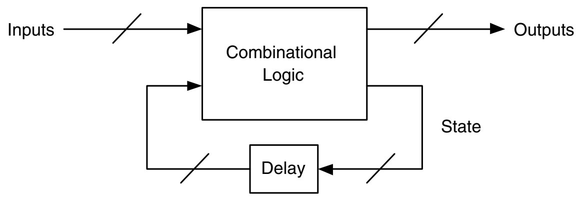

Figure 1. Huffman Asynchronous Circuits

proposed method utilizes the predefined behavior of Burst-Mode controllers for the synchronization. Moreover, errors in redundant logic that cause only hazards but no functional discrepancy were not considered in previous methods.

The remainder of the paper is organized as follows. In Section 2, we briefly introduce the class of asynchronous Burst-Mode controllers. In Section 3, we discuss the shortcomings of duplication-based CED when applied to these circuits and we demonstrate the required remedial action. In Section 4, we describe the Transition-Triggered CED method that we have developed for reducing the cost of duplication in asynchronous Burst-Mode machines. Finally, in Section 5, we provide experimental results quantifying the hardware savings achieved by the proposed method.

2 Asynchronous Burst-Mode Machines

In this section, we introduce briefly the fundamentals of asynchronous Burst-Mode machines. We then outline the synthesis process for realizing an asynchronous Burst-Mode implementation from a given Finite State Machine (FSM) description [8] and we give an example.

2.1 Fundamentals

Burst-Mode machines constitute a class of *Huffman* circuits [9], which is widely used for designing and implementing asynchronous controllers [8, 10, 11, 12]. As shown in Fig. 1, Huffman circuits consist of a set of combinational functions, computing the next state and output of the circuit, and a set of feedback lines, storing the state of the circuit. No clock and no state registers are used in these circuits, however, delay elements are often added to eliminate *essential hazards*[1] [13]. Given the absence of a global clock, *communication protocols* are needed to ensure the correct interaction of an asynchronous circuit and its environment. These protocols define the properties of the stimuli that the environment is allowed to provide to the circuit, as well as the properties of the responses that the circuit will generate. Based on these protocols, several classes of circuits are distinguished.

The key aspect of the protocol used in Burst-Mode machines, as indicated by their name, is that the interaction of the circuit and its environment happens in *Bursts*. An *input burst* is defined as a set of bit changes in one or more inputs of the circuit, which are allowed to occur in any order and without any constraint in their relative time of arrival. Once an input burst is complete, and *only* then, the circuit responds through a hazard-free state and output change to the environment. We emphasize the protocol requirement for hazard-free state and output changes. Since no clock is used, synchronization between the circuit and its environment is based on the fact that any change in the state or output of the circuit signifies completion of an evaluation cycle. Therefore, all hazards should be eliminated to ensure correct circuit functionality and interaction with its environment.

In order to implement a circuit that complies to the aforementioned communication protocol, two features are added during the synthesis process. **First**, in order to make the functionality of the circuit critical-race[2] free, *dichotomies* are added to constrain the binary state encoding of the circuit [14]. Consequently, the resulting state codes ensure that a transition between two states never reaches a transient state with a different destination state for the current input. **Second**, to make the next state and output functions hazard-free, *redundant implicants* are added to their implementation [15].

The popularity of Burst-Mode machines owes itself in part to the extensive research efforts that have been invested in methods and tools for automating their design [8, 10, 11, 12]. For the purpose of this work, we used a comprehensive asynchronous Burst-Mode logic synthesis package called MINIMALIST [8]. The above constraints, along with several optimization algorithms are incorporated in MINIMALIST, yielding a minimal hazard-free logic implementation.

2.2 Example

An asynchronous Burst-Mode machine is described using a state transition table such as the one shown in Fig. 2. The rows in the table correspond to the current symbolic state, the columns correspond to the inputs and the entry indicates the next state and the outputs. For example, if the circuit is in state S_0, an input-burst of 1010 will cause a transition to state S_2 and will generate an output of 00. Let us now assume that the next input burst is 1001, i.e. input c is lowered and input d is raised, and that c is lowered first and then d is raised, i.e. $1001 \rightarrow 1000 \rightarrow 1001$. The circuit responds only after the input burst is complete, so between the time that c is lowered and the time that d is raised, the next state and output function do not change. Once the input burst is complete, the circuit will make a transition to state S_0 and will compute the output, which in this case remains the same, 00.

We note that, depending on the encoding of the states, a critical-race may occur during this transition. For example, if the states are encoded as $S_0 = 00$, $S_1 = 01$ and $S_2 = 11$,

[1]Essential hazards arise when a state change completes before the input change is fully processed. To prevent this early state change from propagating through the combinational logic, delay may be added to the feedback.

[2]A critical-race hazard exists if two state variables change value and the machine's next state depends on the order of arrival of these changes [9, 13].

Inputs: a, b, c, d

States	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
S_0	S_0,00	S_0,00	-	-	-	-	-	-	S_0,01	S_0,00	S_2,00	-	S_1,00	-	-	-
S_1	S_0,00	-	-	-	-	-	-	-	S_1,10	S_1,10	-	-	S_1,00	-	-	-
S_2	-	-	-	-	-	-	-	-	S_2,00	S_0,00	S_2,00	S_2,00	-	-	-	-

Outputs: x, w

Figure 2. Symbolic State Transition Table

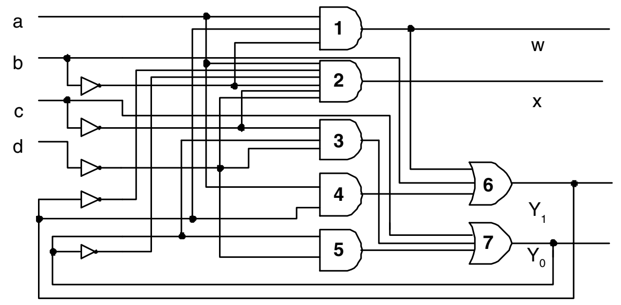

Figure 3. Asynchronous Circuit Implementation

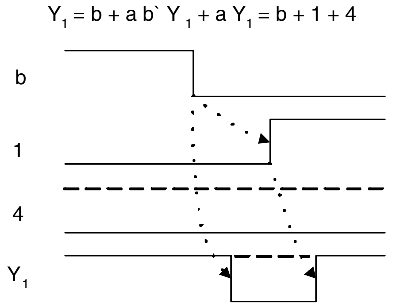

Figure 4. Timing Diagram Illustrating a Hazard

then the transition from S_2 to S_0 may go through a transient state of 01, which is the state encoding of S_1. In combination with the current input burst of 1001, this will produce a next state of S_1 and an output of 10, both of which are incorrect. Thus, this state encoding would be invalid for the circuit.

A dash in a table entry signifies that the corresponding combination of current state and input is not permitted by the communication protocol between the circuit and the environment. For example, if the circuit is in state S_1, an input-burst of 0010 is not allowed to occur. The synthesis process of MINIMALIST starts by performing state minimization on the *symbolic* state transition table, constrained such that the reduced state transition table has a hazard-free logic implementation [12]. In the example of Fig. 2, the state transition table is already minimal. Next, dichotomies are added to ensure a critical-race free state encoding. Solving the dichotomies results in the state encoding $S_0 = 00$, $S_1 = 01$, and $S_2 = 10$ for the example circuit and the symbolic states are replaced by their binary values. The last step is to generate a minimal cost *hazard-free* logic implementation of the circuit [11]. Fig. 3 shows the resulting implementation of the example asynchronous Burst-Mode machine, which includes some logic redundancy to ensure hazard-free operation.

3 Duplication in Burst-Mode Controllers

In this section, we discuss the shortcomings of duplication-based CED when applied to asynchronous Burst-Mode controllers and we propose remedies.

3.1 Shortcomings

Duplication-based CED is the simplest and most commonly used method in synchronous circuits. It employs a copy of the original circuit and a comparator to continuously check the results of the two circuits and identify any error-induced discrepancies. Duplication, however, cannot be directly applied on asynchronous Burst-Mode controllers due to two reasons: (i) the lack of a global synchronization clock and (ii) the existence of redundant logic in the circuit.

The lack of a synchronizing clock introduces uncertainty as to when the responses of the two circuits should be compared. Process variations, input skew and the fact that the two circuits are separate entities are few of the reasons why two identical circuits may compute the correct response with different delay. Consequently, the output of the comparator may temporarily indicate an error, which in this case is a false alarm. The use of duplication-based CED is no longer straightforward without the requirement that the outputs be checked *only* when both are supposed to be ready.

Logic redundancy in Burst-Mode machines prevents hazards from occurring during error-free operation. As a result, some errors may cause *only* hazards but no functional discrepancy. For example, if the current state in the circuit of Fig. 3 is S_1 and the input changes from 1100 to 1000 then the next state should be S_1, as indicated in Fig. 2. However, an error inducing a logic value of '0' at the output of gate 4 will result in a hazard at the next state lines. This is illustrated in the timing diagram of Fig. 4. The dashed line represents the logic value if the error was not present. In this example, the change of input b affects the next state function Y_1 before the change in gate 1 reaches Y_1. During that time, Y_1 is at the logic value of '1' due to gate 4. Consequently, an error in gate 4 will generate a hazard at Y_1 but will not affect correctness of the results. Interestingly, the number of such errors is significant, exceeding 30% in many circuits.

3.2 Proposed Remedies

Based on the communication protocol, an asynchronous Burst-Mode machine changes its state and output only after an input burst is complete. Similarly, the environment is not allowed to provide a new input burst until the output burst of the circuit is complete, i.e. until it has finished evaluation of the previous input burst. This restriction forms the basis for synchronizing the comparison between the original and

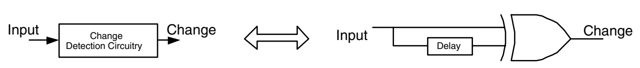

Figure 5. Change Detection Circuit

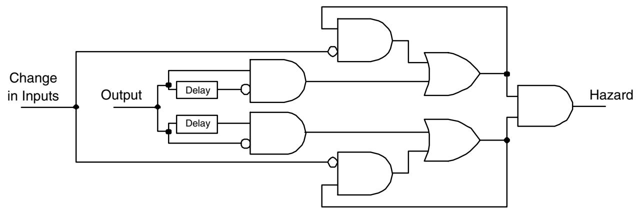

Figure 6. Hazard Detection Circuit

the duplicate circuit. Essentially, we can use the arrival of a new input burst as a valid point in time to check the response of the circuit for the previous input burst. For this purpose, the change detection circuit of Fig. 5 is added to every input line, generating a short comparison window after each input change. When the last bit of the input burst changes, the circuit starts evaluating its new state and output, which will be checked by the comparison window generated by the first bit change of the next input burst.

Error-induced hazards can be detected by adding the circuit of Fig. 6 to the next state and output bits of the circuit. Hazard detection is based on the fact that no more than one transition, either rising or falling, is allowed on these lines after each input burst. The hazard detection circuit is able to observe the occurrence of both a rising and a falling transition on a state or output bit. Every time an input changes, the transitions detected in the hazard detection circuit are reset. Subsequently, the upper feedback loop monitors the line and latches a rising transition while the lower feedback loop does the same for a falling transition. If both transitions are observed, then the hazard signal is asserted.

The enhanced duplication-based CED method is illustrated in Fig. 7. The change detection circuit is added to every input line and the hazard detection circuit is added to every output and state line of the original circuit *only*. Since the duplicate circuit does not interact with the environment, hazard detection in the duplicate circuit is not required. Finally, the error indication signal is asserted if the comparator detects a mismatch between the two circuits after an input change *or* if a hazard is observed in the original circuit.

4 Proposed Transition-Triggered CED

In an effort to reduce the cost of duplication in asynchronous Burst-Mode machines, we propose in this section a *Transition-Triggered CED* method which is shown in Fig. 9. Our method starts with the observation that hazards in the duplicate circuit do not affect the communication protocol between the original circuit and the environment. Therefore, the redundant logic used to make the original circuit hazard-free is not necessary in the duplicate circuit. By allowing hazards to occur in the duplicate circuit, its logic implementation can be optimized to reduce the cost. Hazards on the

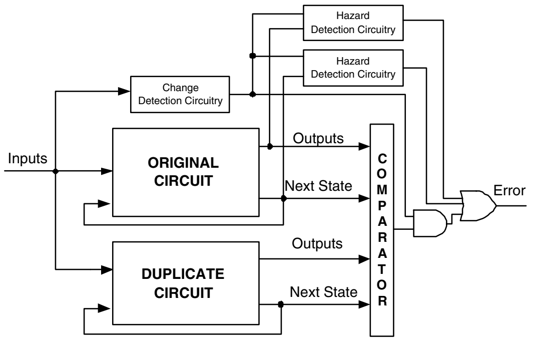

Figure 7. Enhanced Duplication-Based CED

feedback lines of the optimized duplicate, however, jeopardize the operation of the asynchronous circuit. To avoid this problem, we use multiplexors on the state lines of the optimized duplicate circuit to select between the current state and the next state. The multiplexors are controlled using an additional **Transition Prediction Function**, which signifies the end of an input burst. The implementation of the transition prediction function needs to be *hazard-free* to avoid propagation of an incorrect next state through the multiplexors.

Interestingly, the transition prediction function can be used in conjunction with the comparison synchronizer to further reduce the cost of CED in two ways. **First**, the hazard detection circuit that was added to each output and state line can be eliminated: the transition prediction function indicates when changes are *allowed to occur* at the next state and output signals of the original circuit, while the change detection circuit of Fig. 5 indicates when such changes *actually occur*. An error-induced hazard in the original circuit would result in a mismatch between these two signals and may, thus, be detected without the addition of explicit hazard detection circuitry. **Second**, the functionality of the duplicate circuit can be optimized further. Since the hazard-free operation of the original circuit *during an input burst* is checked by the transition prediction function and the response change signal, comparison between the original circuit and the optimized duplicate is necessary *only* after an input burst is complete. Therefore, the functionality of the optimized duplicate for incomplete input bursts can be considered as “don’t-care”, allowing for further hardware reduction in its implementation.

The functionality of the optimized duplicate and the transition prediction function is illustrated in the state transition table of Fig. 8. In contrast to the table of Fig. 2, only entries with a change in the output or next state lines are defined for the duplicate circuit; the rest are “don’t cares”. The transition prediction function is defined for every defined entry in the state transition table of the original circuit and takes a logic value of ‘1’ only when a transition appears in the next state or output lines at the end of an input burst. The state transition table shown in Fig. 8 is used to design the proposed Transition-Triggered CED method illustrated in Fig. 9. The state of the transition prediction function and the optimized duplicate circuit changes after an input burst is complete and

States	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
S_0	-, --0	-, --0	-	-	-	-	-	-	S_0,011	-, --0	S_2,001	-	S_1,001	-	-	-
S_1	S_0,001	-	-	-	-	-	-	-	S_1,101	-, --0	-	-	-, --0	-	-	-
S_2	-	-	-	-	-	-	-	-	-, --0	S_0,001	-, --0	-, --0	-	-	-	-

Figure 8. Optimized Symbolic State Transition Table with a Transition Prediction Function

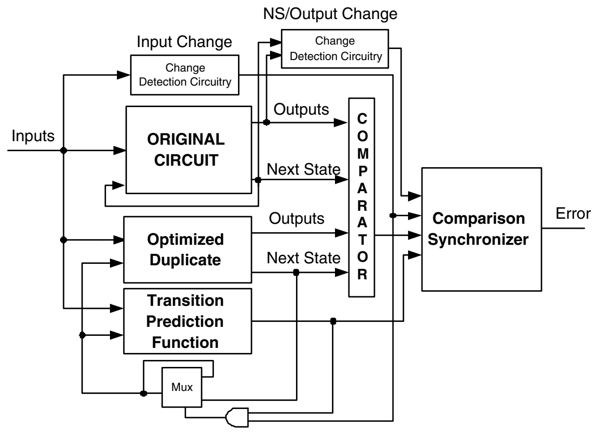

Figure 9. Transition-Triggered CED

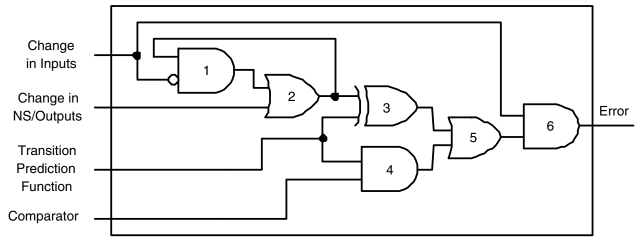

Figure 10. Comparison Synchronizer

the next input burst starts. The structure of the comparison synchronizer is illustrated in Fig. 10. The comparison synchronizer uses four control signals to generate the error indication signal: the change in the inputs, the change in the next state or outputs, the transition prediction function and the result of the comparator. The signal indicating a change in next state or output between input changes is stored using gates 1 and 2. Every time an input changes, two comparisons are performed: First, the change in the next state and outputs signal is compared to the transition prediction function using gate 3, in order to detect any unexpected transitions in the original circuit. Second, the result of the comparator is enabled, using the transition prediction function in gate 4, to check the response of the original circuit for correctness. The logic OR of these two comparisons, which is computed in gate 5, constitutes the error indication signal.

5 Experimental Results

In this section, we compare the area overhead of the proposed CED method to duplication-based CED. The specifications of the circuits used in these experiments are provided along with MINIMALIST in [16]. The circuits are first synthesized using MINIMALIST to generate an asynchronous implementation in *pla* [17] format. Next, the change detection, hazard detection and duplicate circuits are added to the original circuit, as described in Section 3.2, to perform duplication-based CED. The optimized duplicate circuit of the proposed Transition-Triggered CED is produced using *espresso* [17] based on the specification of the original circuit. The transition prediction function is generated using MINIMALIST to ensure hazard-free behavior. The change detection circuit, state multiplexors, and output comparator are added to the original circuit, as described in Section 4, to perform the proposed Transition-Triggered CED.

The results are analytically presented for the individual components of duplication-based CED in Table 1 and the proposed Transition-Triggered CED in Table 2. The gate count of the circuits is normalized to the equivalent number of 2-input NAND-gates. Under the first major heading in Table 1, we provide details about the circuits that were used: name, number of primary inputs, number of states and number of primary outputs. The literal and gate count of the original circuit, the duplicate circuit, the hazard detection circuit, the comparator and the CED synchronizer are presented in the second, third, fourth, fifth and sixth major heading, respectively. The last major heading summarizes the total literal and gate count for duplication-based CED. The results for the proposed Transition-Triggered CED method are illustrated in Table 2. Under the first two major headings, we repeat the original circuit information that appears in Table 1. The literal and gate count of the optimized duplicate circuit, the state multiplexors, the transition prediction function, the comparator and the CED synchronizer are presented in the third, fourth, fifth, sixth and seventh major heading, respectively. The last column summarizes the total literal and gate count for the proposed Transition-Triggered CED.

The average gate-count reduction of the proposed method over duplication-based CED over all benchmark circuits is 22.56%. In small benchmark circuits, the cost of the hazard detection circuit is very high relatively to the cost of the original circuit and, thus, the proposed method outperforms duplication-based CED significantly. For example, this is the case for $tangram - mixer$ and $hp - ir$. Moreover, the proposed method also reduces significantly the cost for large circuits, where the ratio of the cost of the hazard detection circuit to the original circuit is small. For example, this is the case for $pe - send - ifc$ and $p1$, where the cost is reduced by more than 22%. The cost of the optimized duplicate circuit is very close to the cost of the original circuit for small benchmarks. For example, this is the case for circuits $concur - mixer$ and $hp - ir$. This is attributed to the small amount of redundant logic used to design a hazard-free implementation of the specification of these circuits. In more

Circuit		Original		Duplicate		H. D. Circuit		Comparator		Synchronizer		Total	
Name	I/S/O	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates
concur-mixer	3/3/3	26	16	26	16	78	70	38	25	22	17	190	144
martin-q-element	2/2/2	14	9	14	9	46	41	22	14	14	12	110	85
opt-token-distributor	4/6/4	74	41	74	41	112	98	56	35	30	23	346	238
pe-send-ifc	5/5/3	110	58	110	58	96	84	48	30	38	30	402	260
tangram-mixer	3/2/2	17	10	17	10	46	41	22	14	22	17	124	92
p1	13/11/14	458	238	458	238	278	247	134	85	142	77	1470	885
while_concur	4/4/3	41	24	41	24	96	84	48	30	30	23	256	185
rf-control	6/6/5	75	37	75	37	128	112	64	40	48	36	390	262
hp-ir	3/2/2	13	8	13	8	46	41	22	14	22	17	116	88
while	4/3/3	27	16	27	16	78	70	38	25	30	23	200	150

Table 1. Experimental Results for Duplication-Based CED

Circuit		Original		Opt. Duplicate		Muxes		Transition Func.		Comparator		Synchronizer		Total	
Name	I/S/O	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates
concur-mixer	3/3/3	26	16	22	14	12	7	6	3	38	25	60	47	164	112
martin-q-element	2/2/2	14	9	12	7	6	4	4	3	22	14	36	29	94	66
opt-token-distributor	4/6/4	74	41	46	22	18	11	23	14	56	35	86	65	303	188
pe-send-ifc	5/5/3	110	58	48	27	18	11	20	10	48	30	86	65	330	201
tangram-mixer	3/2/2	17	10	13	8	6	4	7	4	22	14	44	34	109	74
p1	13/11/14	458	238	185	99	24	14	92	49	134	85	276	180	1169	665
while_concur	4/4/3	41	24	21	12	18	11	18	11	48	30	78	59	224	147
rf-control	6/6/5	75	37	34	19	18	11	25	15	64	40	112	84	328	206
hp-ir	3/2/2	13	8	10	6	6	4	8	5	22	14	44	34	103	71
while	4/3/3	27	16	13	9	12	7	17	9	38	25	68	53	175	119

Table 2. Experimental Results for Transition-Triggered CED

complex circuits, such as $while\_concur$ and $rf - control$, the cost of the optimized circuit is almost 50% of the cost of the original circuit. Moreover, the cost of the optimized duplicate of the largest asynchronous controller, $p1$, is less than 42% of the cost of the original circuit. As the circuit specification becomes more complex, the percentage of redundant logic that can be saved by Transition-Triggered CED over duplication-based CED also increases.

6 Conclusion

Duplication-based CED is not directly applicable to the class of asynchronous Burst-Mode controllers due to the lack of a global clock, the existence of redundant logic, and the communication protocol between these circuits and the environment. We showed that these obstacles can be overcome at the expense of additional hardware for comparison synchronization and hazard detection, resulting in a comprehensive CED method that detects not only the functional correctness of the circuit, but also its correct interaction with the environment. Furthermore, we demonstrated that a transition prediction function can be used to reduce the incurred overhead. The proposed Transition-Triggered CED method eliminates the additional hazard detection circuitry and permits a less expensive implementation of the duplicate circuit. Thus, as indicated through experimental results, an average of 22% in hardware savings is achieved.

References

[1] W.-C. Chou et al., "Average-case optimized technology mapping of one-hot domino circuits," in *ASYNC*, 1998, pp. 80–91.

[2] C. Shi and J. Brzozowski, "An effecient algorithm for constrained encoding and its applications," *IEEE TCAD*, vol. 12, no. 12, pp. 1813–1826, 1993.

[3] K. Y. Yun and D. L. Dill, "Automatic synthesis of extended burst-mode circuits," in *ICCAD*, 1999, pp. 118–132.

[4] B. Lin and S. Devadas, "Synthesis of hazard-free multi-level logic under multiple-input changes from binary decision diagrams," *IEEE TCAD*, vol. 14, no. 8, pp. 974–985, 1995.

[5] T. Verdel and Y. Makris, "Duplication-based concurrent error detection in asynchronous circuits: shortcomings and remedies," in *DFT*, 2002, pp. 345 – 353.

[6] S. J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems," in *FTCS*, 1995, pp. 228–237.

[7] D. A. Rennels and H. Kim, "Concurrent error detection in self-timed VLSI," in *FTCS*, 1994, pp. 96–105.

[8] R. M. Fuhrer and S. M. Nowick, *Sequential Optimization of Asynchronous and Synchronous Finite-State Machines: Algorithms and Tools*, Kluwer Academic Publishers, 2001.

[9] D. A. Huffman, *The Synthesis of Sequential Switching Networks*, Addison-Wesley, 1964.

[10] S. M. Nowick, *Automatic Synthesis of Burst-Mode Asynchronous Controllers*, Ph.D. thesis, Stanford University, 1993.

[11] S. M. Nowick and D. L. Dill, "Exact two-level minimization of hazard-free logic with multiple-input changes," *IEEE TCAD*, vol. 15, no. 8, pp. 986–997, 1995.

[12] R. M. Fuhrer and S. M. Nowick, "Optimista: State minimization of asynchronous FSMs for optimum logic," in *ICCAD*, 1999, pp. 7–13.

[13] S. H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley-Interscience, 1969.

[14] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machine," *IEEE TCAD*, vol. 4, no. 3, pp. 269–285, 1985.

[15] E. J. McCluskey, "Minimization of boolean functions," *Bell System Technology Journal*, vol. 35, pp. 1417–1444, 1956.

[16] "Tools (MINIMALIST)," Available from `http://www1.cs.columbia.edu/async/`.

[17] E. M. Sentovich et al., "SIS: a system for sequential circuit synthesis," ERL MEMO. No. UCB/ERL M92/41, EECS UC Berkeley CA 94720, 1992.