An Algorithm to Remove Asynchronous ROMs in Circuits with Cycles

Md. Nazrul Islam Mondal, Koji Nakano, and Yasuaki Ito
Department of Information Engineering, Hiroshima University
1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan

***Abstract*—Field Programmable Gate Arrays (FPGAs) are a dominant implementation medium for digital circuits which are used to embed a circuit designed by users instantly. FPGAs can be used for implementing parallel and hardware algorithms. Most of FPGAs have Configurable Logic Blocks (CLBs) to implement combinational and sequential circuits and block RAMs to implement Random Access Memories (RAMs) and Read Only Memories (ROMs). Circuit design that minimizes the number of clock cycles is easy if we use asynchronous read operations. However, most of FPGAs support synchronous read operations, but do not support asynchronous read operations. The main contribution of this paper is to provide one of the potent approaches to resolve this problem. We assume that a circuit using asynchronous ROMs designed by a non-expert or quickly designed by an expert is given. Our goal is to convert the circuit with asynchronous ROMs into an equivalent circuit with synchronous ones. The resulting circuit with synchronous ROMs can be embedded into FPGAs.**

***Index Terms*—FPGA, Read only memories, Asynchronous read operations, Circuit rewriting algorithm.**

I. Introduction

An FPGA is a programmable VLSI (Very Large Scale Integration) in which a hardware designed by the users can be embedded quickly. Typical FPGAs consist of an array of programmable logic blocks (slices), memory blocks, and programmable interconnects between them. The logic block contains four-input logic functions implemented by a LUT and/or several registers. Using four-input logic functions, registers, and their interconnections, any combinational circuit and sequential logic can be implemented. The memory block is a dual-port RAM which can perform read and/or write operations for a word of data to two distinct or same addresses in the same time. Usually, the dual-port RAM supports synchronous read and synchronous write operations. The read and write operations are performed at the rising clock edges. The dual-port RAM outputs data of a specified address after the rising edge. Similarly data is written to a specified address at the rising edge of clock if write enable is high. Design tools are available to the users to embed their hardware logic into the FPGAs. Some circuit implementations are described [1], [2], [4], [9] to accelerate computation.

In this paper, we focus on the asynchronous and synchronous read operations of memory blocks as follows:

Asynchronous read operation
: The memory block outputs the data specified by the address given to the address port. When the address value is changed, the output data is updated immediately within some delay time. In other words, the output data port always outputs $M[d]$, which is the data stored in the input address value d.

Synchronous read operation
: Even if the address value is changed, the output data is not updated. The output data is updated based on the address value at the rising edge of clock. More specifically, the output data port outputs $M[d]$, where d is the address data at the previous point of rising clock edge.

Let *AROMs* and *SROMs* denote ROMs with asynchronous and synchronous read operations, respectively. In general, the circuit design is simpler and easier to the designers, in particular to the non-expert circuit designers if AROMs are available. In asynchronous read operation, the value of a specified address can be obtained immediately. However, in synchronous read operation, one clock cycle is required to obtain it. Nevertheless, block RAMs embedded in most of the current FPGAs do not support asynchronous read operation for increasing its operating clock frequency.

A circuit implementation with AROMs is better than SROMs implementation, because of less power consumption, easy to design etc. But it has some problems like small in size so that it does not support the designer's demand, more expensive, and less speedy [3], [5], [6]. To cut the clock distribution power, an asynchronous circuit design in FPGAs is very much suitable, described in [7], [10], [11]. However, it is not supported by the current FPGAs.

On the other hand, a circuit implementation with SROMs is dominating the modern digital circuit design industry, because it supports the modern FPGA architecture although it has some drawbacks to design like clock distribution, more power consumption etc [3], [6]. So we should use SROMs when we need a function of ROMs.

The main contribution of this paper is to present a circuit rewriting approach that converts *an asynchronous circuit* consisting

> *Combinational Circuits (CCs), Registers (Rs), and ROMs with asynchronous read operations (AROMs)*

into *an equivalent synchronous circuit* consisting

> *Combinational circuits (CCs), Registers (Rs), and ROMs with synchronous read operations (SROMs).*

Note that, most of the current FPGAs support synchronous read operation, but do not support asynchronous one. We are

978-0-7695-4569-1/11 $26.00 © 2011 IEEE
DOI 10.1109/ICNC.2011.20

thinking the following scenario to use our circuit rewriting algorithm:

- An asynchronous circuit designed by a non-expert, or quickly designed by an expert is given.
- Our circuit rewriting algorithm convert it into an equivalent synchronous circuit.
- The resulting synchronous circuit can be implemented in FPGAs.

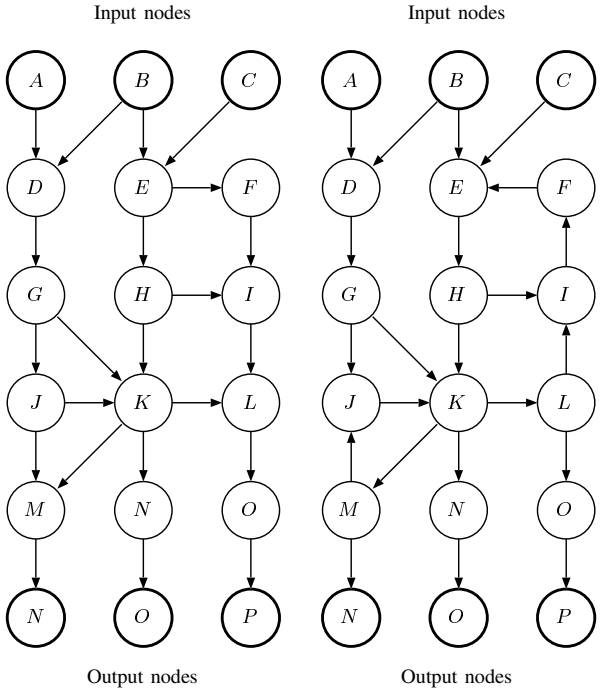
In other words, designers can design a circuit for FPGAs under the assumption of asynchronous read operation, which is simpler and easier than one with synchronous read operation.

For the reader's benefit, we will show a simple example illustrating that the circuit design is simpler if AROMs are available. Suppose that for an input X_0, we need to compute $X_n = X_{n-1} + f(X_{n-1})$ for every $n \geq 1$. We assume that the function f is computed using a ROM. More specifically, we use a ROM such that address i is storing a value of $f(i)$. Figure 1 (a) illustrates a circuit with an AROM to compute $X_1, X_2, \ldots$ for input X_0. An AROM is used to compute the value of $f(X_n)$ for a given X_n. It should be clear that this circuit outputs $X_1, X_2, \ldots$ in every clock cycle. Figure 1 (b) shows a circuit with an SROM. Since one clock cycle is necessary to read the value of $f(X_n)$ for input X_n, we need to insert a register to synchronize two inputs X_n and $f(X_n)$ of the adder as illustrated in the figure. This circuit outputs $X_1, X_2, \ldots$ in every two clock cycles. Hence, the circuit in Figure 1 (b) needs double clock cycles over the circuit in Figure 1 (a). Using our algorithm to the circuit in Figure 1 (a), we can obtain the circuit in Figure 1 (c) automatically. In the circuit with an SROM in Figure 1 (c), $X_1, X_2, \ldots$ is output in every clock cycle. Thus, the timings of the circuits in Figure 1 (a) and (c) are identical.

In our previous paper [8], we have presented a circuit rewriting approach for circuits represented by a directed acyclic graph (DAG), which has no directed cycle. Figure 2 (1) illustrates an example of a DAG. This graph has 3 input nodes and 3 output nodes, each of which corresponds to input ports and output ports of the circuit, respectively. The other internal nodes corresponds to circuit elements such as combinational circuits, registers, and ROMs. The presented circuit rewriting approach converts a circuit with combinational circuits, registers and AROMs represented by a DAG into an equivalent AROM-free circuit with combinational circuits, registers and SROMs.

However, the circuit rewriting approach presented in [8] has a strict restriction in terms of input circuits. It works only for a circuit whose underlying graph is a DAG, illustrated in Figure 2 (1). Although most of practical circuits have cycles, it can not handle such circuits.

The main contribution of this paper is to modify the circuit rewriting algorithm, presented in [8] to process practical circuits with cycles. More specifically, our new circuit rewriting algorithm can convert any circuit represented by a directed reachable graph (DRG), illustrated in Figure 2 (2). A directed reachable graph is a directed graph such that, for every internal node, there exists a directed path from an input node to an output node which includes it. Note that, one node and/or one directed path may appear twice or more in a directed path. For example, $(B, E, H, I, F, E, H, K, N, O)$ is a directed path. It should not have any difficulty to confirm that, every internal node in Figure 2 (2) is included. Clearly, a class of the DRG includes that of the DAG. Also, almost all practical circuits can be represented by a DRG. If there exists a node that is not in the directed path from an input node to an output node, the directed graph is not a DRG. However, the corresponding circuit element to such node makes no sense. Consequently, we can say that our new circuit rewriting approach can handle almost all practical circuits with combinational circuits, registers, and AROMs.



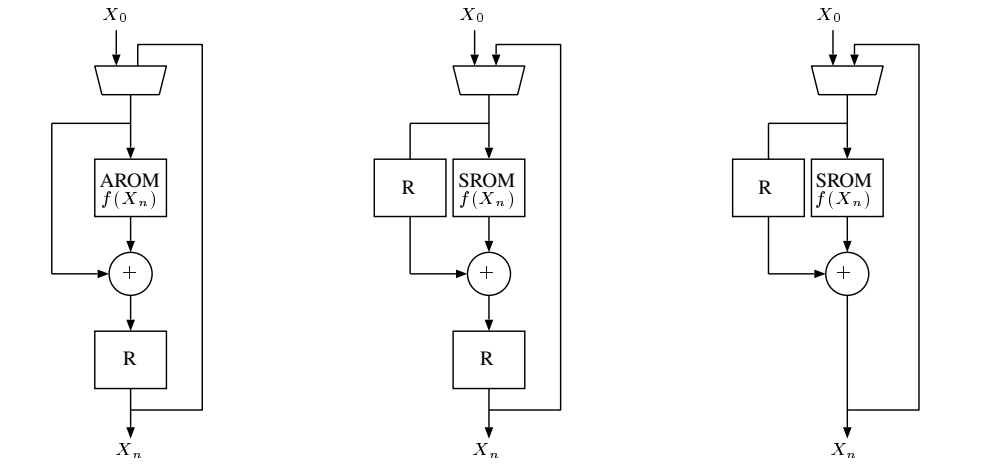
(1) Directed Acyclic Graph (DAG) (2) Directed Reachable Graph (DRG)

Fig. 2. A directed acyclic graph(DAG) and a directed reachable graph(DRG)

This paper is organized as follows: Section II briefly reviews the circuits and their equivalency. In Section III, we describe our rewriting algorithm, circuit graph and also explain the equivalency for our rewriting rules. For the reader's benefits, Section IV shows how our circuit rewriting algorithm works for circuit graphs. Section V presents the proof of the correctness of our rewriting algorithm. Finally Section VI concludes this work.

II. Circuits and Their Equivalence

Let us consider a synchronous sequential circuit that consists of input ports, output ports, combinational circuits (CCs), registers (Rs), Read Only Memories (ROMs), a global clock input (clock), and a global reset input (reset).



(a) A circuit with an AROM

(b) A circuit with an SROM by a non-expert

(c) The converted circuit with an SROM

Fig. 1. An example of circuits using an AROM and an SROM

A combinational circuit (CC) is a network of fundamental logic gates with no feedback. So, it can compute Boolean functions represented by Boolean formulas, such as $F = A \cdot \overline{B} + B \cdot C$ and $G = \overline{B \cdot C}$ as illustrated in Figure 3. Once inputs are given, the outputs are computed in small delay.

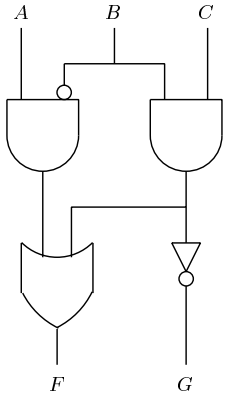


Fig. 3. An example of a combinational circuit (CC).

A register has a clock input and a reset input as illustrated in Figure 4. It can store fixed bits of data. If reset is 1, then the b-bit data is initialized by 0. If reset is 0, the stored data is updated by the value given to the input port d at every rising clock edge. The data stored in the register is always output from port q.

A ROM (Read Only Memory) has a (address) input d and a data output q as illustrated in Figure 4. It is storing 2^b words such as $M[0]$, $M[1]$, ..., $M[2^b - 1]$, where b is the number of address bits. We deal with two types of ROMs in terms of read operations as follows:

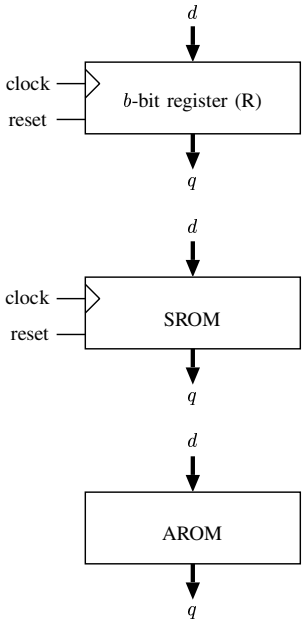


Fig. 4. A register (R), a synchronous ROM (SROM) and an asynchronous ROM (AROM).

- **Synchronous ROM (SROM)** An SROM has a clock input and a reset input. If reset is 1 then the stored value is initialized by 0. The read operation is performed at every rising clock edge when reset is 0. The output q is the value of $M[d]$ at the latest rising clock edge.
- **Asynchronous ROM (AROM)** An AROM has no clock input and no reset input. The value of $M[d]$ is continu-

ously output from port q.

The Figure 5 shows a timing diagram of reading operations of the R, SROM, AROM and NR (Negative Register). In the figure, time 0, 1, 2, ... correspond to rising edges of the periodic clock input. Initially global reset is 1 and it drops to 0 just before time 0. Data d_0, d_1, d_2, ... are given to the input port d. The value of output, q of R and SROM is 0 at time 0. Also, at time 1, 2, ... the values of output, q of R and SROM are d_0, d_1, d_2, ... and $M[d_0]$, $M[d_1]$, $M[d_2]$, ..., respectively. For the AROM, the data $M[d_0]$, $M[d_1]$, $M[d_1]$, ... are taken from the output port, q immediately at time 0, 1, 2, ..., respectively.

In current FPGAs, an SROM can be implemented in embedded block RAMs. However, an AROM is implemented in LUTs, which are very costly. Hence, we should use SROMs when we need a function of ROMs. On the other hand, AROM is easy to use, because we can get output data from the AROM immediately.

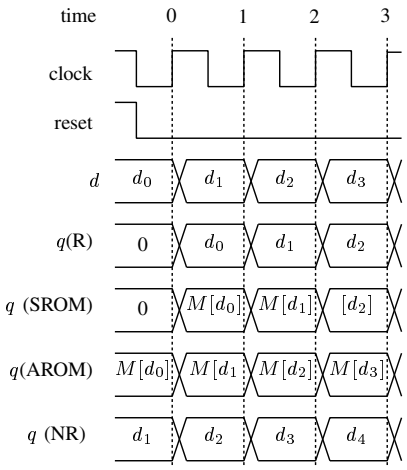


Fig. 5. A timing chart of a register (R), an SROM, an AROM and a negative register (NR).

We will describe a behavior of a circuit element using a sequence of output at every rising clock edge for the *periodic clock* (clock is inverted into a fixed frequency), and *initial reset* (initially, reset is 1 and drops to 0 before the first rising clock edge) as illustrated in Figure 5. The behavior of each circuit element is described by the output sequences as follows:

- **Combinational Circuit (CC)** For simplicity, we assume 3-input 2-output combinational circuit which is shown in Figure 3. There is no difficulty to extend the definition for general m-input n-output combinational circuit. We assume that, at time i ($i \geq 0$), a_i, b_i, and c_i are given to the 3 input ports A, B, and C. Let f and g be the two functions with three arguments that determine the value of output ports F and G. The output sequences of F and G are as follows:

  CC(F):$\langle f(a_0, b_0, c_0), f(a_1, b_1, c_1), f(a_2, b_2, c_2), \ldots \rangle$
  CC(G):$\langle g(a_0, b_0, c_0), g(a_1, b_1, c_1), g(a_2, b_2, c_2), \ldots \rangle$

- **Register (R)** Let d_i denote an input value given to an input port d at time i ($i \geq 0$). The output sequence is described as follows:

  R: $\langle 0, d_0, d_1, d_2, \ldots \rangle$

- **Synchronous and Asynchronous ROMs (SROMs and AROMs)** Let $M[j]$ denote the value stored in address j ($j \geq 0$) of the ROM. The output sequences of SROM and AROM are as follows:

  SROM: $\langle 0, M[d_0], M[d_1], M[d_2], \ldots \rangle$
  AROM: $\langle M[d_0], M[d_1], M[d_2], M[d_3], \ldots \rangle$

In this paper, we assume that a fully synchronous circuit has data inputs, data outputs, a global clock input, a global reset input, combinational circuits (CCs), registers (Rs), SROMs, AROMs, and their interconnects. The readers should refer to Figure 6 for illustrating an example of a fully synchronous circuit. The global clock and the global reset are directly connected to the clock input ports and the reset input ports of all Rs and SROMs. Also, we assume that a circuit has cycles.

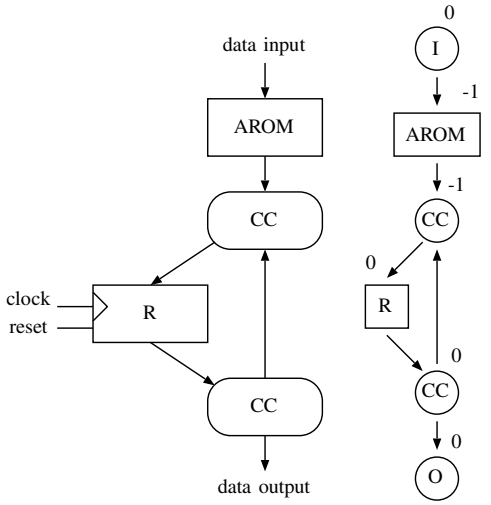


Fig. 6. An example of a fully synchronous circuit with cycle and the corresponding circuit graph with potentiality.

Let us define *equivalence* of two fully synchronous circuits for the periodic clock and initial reset. We say that two circuits X and Y are an *equivalent* if, for any input sequence, the output sequences are the same except for first several outputs. For the reader's benefit, we will show an example of the equivalence.

Let us consider a circuit R+AROM, that is, the output of R is connected to the input of AROM as illustrated in Figure 7. We also consider a circuit AROM+R, in which the output of AROM and the input of R are connected. For the periodic clock with initial reset, the output sequences of SROM, R+AROM, and AROM+R are as follows:

SROM: $\langle 0, M[d_0], M[d_1], M[d_2], \ldots \rangle$
R+AROM: $\langle M[0], M[d_0], M[d_1], M[d_2], \ldots \rangle$

AROM+R: $\langle 0, M[d_0], M[d_1], M[d_2], \ldots \rangle$

Since these three circuits have the same output in time 1, 2, ..., they are equivalent. Note that the outputs in time 0 are not equal. In this paper, we ignore first several clock cycles when we determine the equivalency of the circuits.

Suppose that a circuit X with AROMs is given. The main contribution of this paper is to show

- a necessary condition such that an AROM-free circuit, Y can be generated, which is equivalent to X, and
- an algorithm to derive Y if the necessary condition is satisfied.

We will introduce *a negative register* (NR), which is a nonexistent device used only for showing our algorithm to derive Y and related proofs. This is originally introduced in our previous paper [8]. Recall that, a regular register latches the input at the rising clock edge. *A negative register* latches a future input. The Figure 5 also shows a timing diagram of a negative register (NR). An NR latches the value of input *d* at the rising edge of two clock cycles later as illustrated in Figure 5. Thus, the NR has the following output sequence for a periodic clock with an initial reset is as follows:

NR: $\langle d_1, d_2, d_3, \ldots \rangle$.

In our algorithm to derive an AROM-free circuit Y, circuits with NRs will be used as interim results.

III. Circuit Graph and Rewriting Rules

We simply use a directed graph to denote the interconnections of a fully synchronous circuit. We call such graph *as a circuit graph.* A circuit graph consists of a set of nodes and a set of directed edges for connecting two nodes. Each node is labeled by either I (Input port), O (Output port), CC (Combinational Circuit), R (Register), NR (Negative Register), AROM, or SROM. A node with label I is connected with one or more outgoing edges. A node with label O is connected with exactly one incoming edge. A node with label CC has one or more incoming edges and one or more outgoing edges. A node with label R, NR, AROM, or SROM has one incoming and one outgoing edge. We also assume that a circuit graph is a directed reachable graph (DRG), such that for every internal node, there exists a directed path from an input node to an output node which includes it. Figure 2 (2) illustrates an example of a DRG.

Note that nodes with label I, R, NR, AROM, or SROM has only one outgoing edge. The readers may think that one outgoing edge is a too stringent restriction because it does not allow two or more fan-outs. However, we can implement multiple fan-outs by attaching a simple Combinational Circuit (CC) that just duplicates the input. For example, a CC with one input port A and two output ports F and G such that $F = A$ and $G = A$ is used to implement fan-out 2 as illustrated in Figure 8.

For a given circuit X with AROMs, we will show an algorithm to derive an AROM-free and NR-free circuit, Y by rewriting circuits. We assume that X is given as a circuit graph. We will define rules to rewrite a circuit graph. The

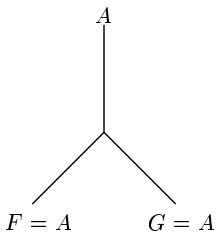


Fig. 8. A combinational circuit to implement fan-out 2 circuit.

readers should refer to Figure 9 for illustrating the rules, where P and S denote predecessor and successor nodes respectively. The nodes between predecessor and successor nodes are rewritten as follows:

Rule 0 AROM node is rewritten into SROM+NR.

Rule 1 Adjacent R and NR nodes are rewritten into NULL circuit, that is, they are removed.

Rule 2 R+SROM is rewritten into SROM+R.

Rule 3 If all the incoming edges of a CC node are connected to an R node, then all the Rs are removed to all the outgoing edges of the CC node.

Rule 4 NR+SROM is rewritten into SROM+NR.

Rule 5 If one of the incoming edges of a CC node is connected to an NR node, then the NR node is removed, an R node is added to all the other incoming edges, and the NR node is moved to all the outgoing edges of the CC node.

The readers should have no difficulty to confirm that, after applying one of the rewriting rules, an original circuit and the resulting circuit are equivalent. Please see [8] for the details regarding the equivalency.

We are now in position to describe the rewriting algorithm. Suppose that an input circuit graph has nodes with labels *I*, *O*, *R*, *AROM*, *SROM*, and *CC*. The following rewriting algorithm generates a circuit graph equivalent to the original circuit graph.

> *Find a minimum i such that Rule i can be applied to the current circuit graph. Rewrite the circuit graph using such Rule i. This rewriting procedure is repeated until no more rewriting is possible.*

IV. Behavior of our circuit rewriting algorithm

Let us observe the behavior of our circuit rewriting algorithm.

- First, Rule 0 is applied to all AROM nodes, and they are rewritten into SROM+NR. After that, Rule 0 is never applied.
- Rules 1 is applied and adjacent R and NR nodes are removed whenever possible.
- R nodes are moved toward the output nodes using Rules 2 and 3 whenever possible.
- NR nodes are moved toward the output nodes or are rotated in cycles using Rules 4 and 5.

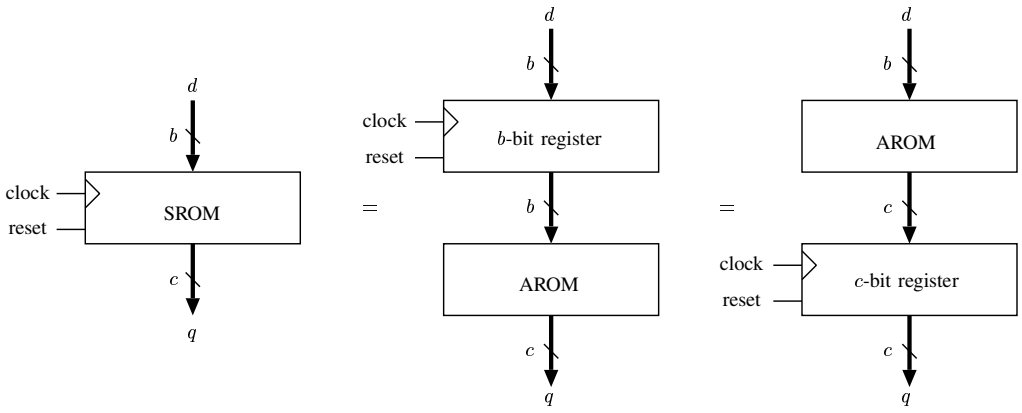


Fig. 7. SROM, R+AROM, and AROM+R.

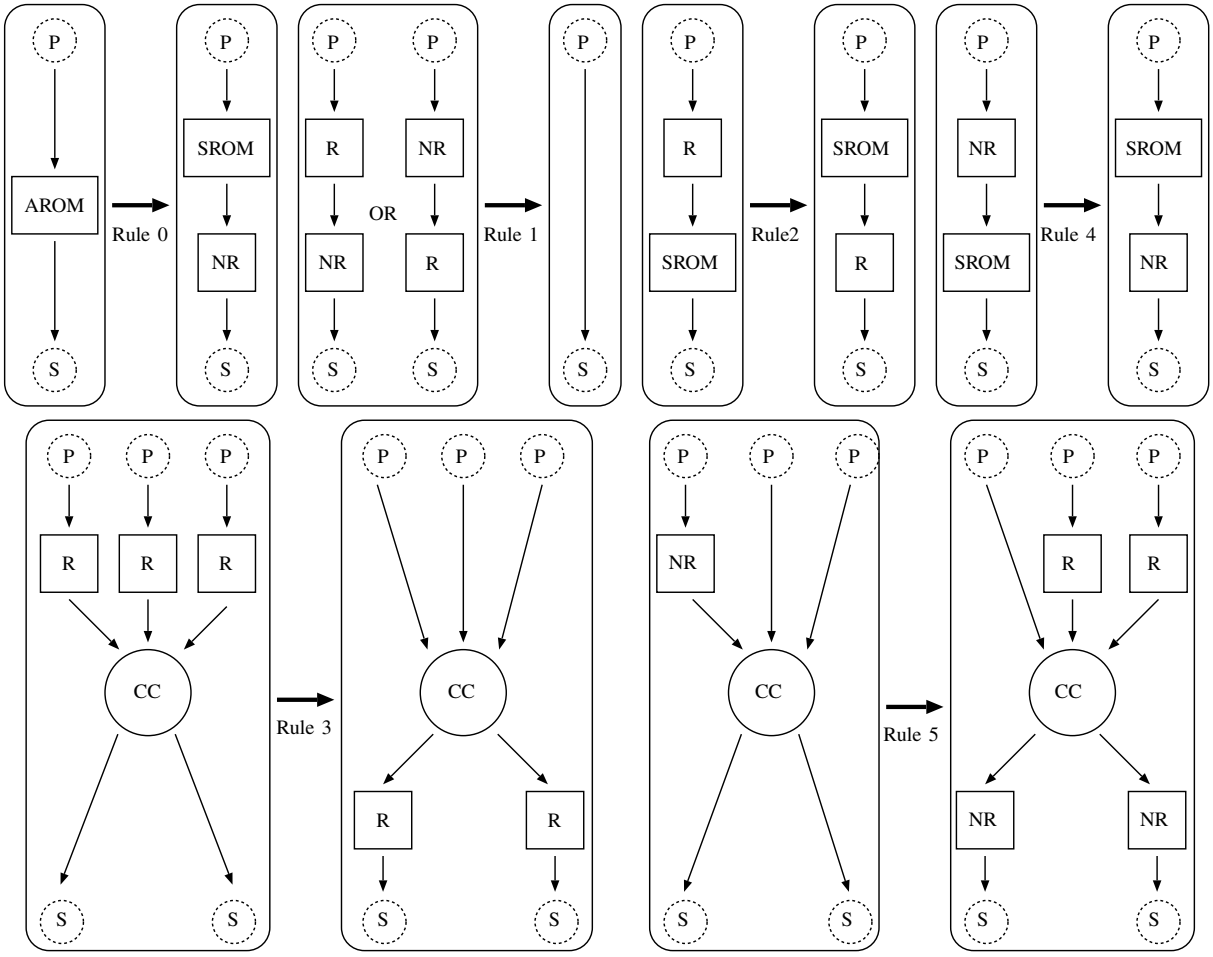


Fig. 9. Rules to rewrite a circuit graph.

Let us see how our circuit rewriting algorithm works using an example of a circuit in Figure 10, which shows the interim and resulting circuit graphs. First, Rule 0 is applied to the AROM, it is converted into SROM+NR. After that, Rule 3 is used to move the R, and two Rs are generated. Rule 5 is applied to move the NR and it is duplicated. Finally, adjacent

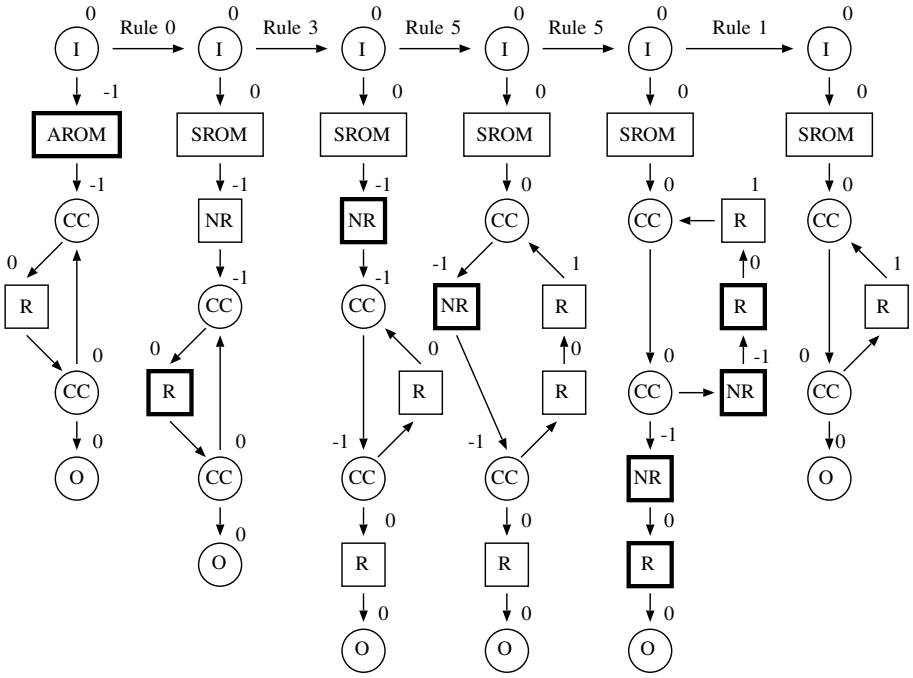


Fig. 10. Interim and resulting circuit graphs obtained by our rewriting algorithm for a circuit graph.

R and NR are removed by Rule 1.

Our circuit rewriting algorithm may not terminate for a circuit graph that has no way to convert an equivalent AROM-free circuit. Figure 11 shows an example of such circuit graph. It has a cycle with two AROMs and one R. Intuitively, one R is necessary to convert an AROM into an SROM. Thus, this circuit graph can not be converted into an equivalent AROM-free circuit. Let us see how our circuit rewriting algorithm works for the circuit graph in Figure 11. After applied Rule 0 and Rule 1, the interim circuit graph has an NR in the cycle. Rule 5 is applied to move the NR, and a new R is generated between the I node and the CC node. After that, the NR jumps over the SROM by Rule 4. Rule 5 is applied again, and a new NR is generated between the CC node and the O node. Again, the NR jumps over the SROM by Rule 4. The readers should have no difficulty to confirm that, while the NR is rotated in the cycle, one new R is generated between the I node and the CC node and one new NR is generated between the CC node and the O node. Rule 5 and Rule 4 can be repeated applied in the same way. In general, after Rule 5 and Rule 4 applied $2n$ times, new n R's and n NR's are generated, and our circuit rewriting algorithm never terminates.

For the purpose of clarifying the condition such that our rewriting algorithm can generate AROM-free and NR-free circuit graph, we define *the potentiality of the nodes* in a circuit graph. Suppose that a node v of a circuit graph has m (≥ 0) incoming edges such as $(u_1, v), (u_2, v), \ldots, (u_m, v)$. Let us define *the potentiality* $p(v)$ of a node v as follows:

- If v is I, then $p(v) = 0$.
- If v is O or SROM, then $p(v) = p(u_1)$.
- If v is AROM or NR then $p(v) = p(u_1) - 1$.
- If v is R then $p(v) = p(u_1) + 1$.
- If v is CC, then $p(v) = \min(p(u_1), p(u_2), \ldots, p(u_m))$.

From the definition, the potentiality of a node can be determined if the potentiality of all predecessor nodes are determined. Unfortunately, as we will show next, we may not determine the potentiality of every node by the above definition, if a circuit graph has a cycle.

Let us discuss the potentiality for a circuit graph with a cycle using three circuits in Figure 12. Let the potentiality $p(a)$ of the CC node a be k. From the definition of the potentiality, we can write the equations of potentiality for Figure 12 (1) as follows:

$p(a) = k$, $p(b) = \min(p(a), p(e))$, $p(c) = p(b) + 1$,
$p(d) = p(c)$, $p(e) = p(c) + 1$, and $p(f) = p(d)$.

From these equations, we have, $p(e) = p(c)+1 = p(b)+2$ and thus, $p(b) = \min(k, p(b) + 2)$. Hence, we can determine the value of $p(b)$ such that $p(b) = k$. Further, we can determine the potentiality of the other nodes as follows: $p(c) = p(d) = p(f) = k + 1$, and $p(e) = k + 2$. Intuitively, the equation $p(b) = \min(k, p(b) + 2)$ means that the cycle is a *positive cycle* because the cycle $b - c - d - e$ increases the potentiality by $+2$.

We can do the same discussion for Figure 12 (2) as follows:

$p(a) = k$, $p(b) = \min(p(a), p(e))$, $p(c) = p(b) + 1$,
$p(d) = p(c)$, $p(e) = p(c) - 1$, and $p(f) = p(d)$.

From these equations, we have, $p(b) = \min(k, p(b))$. Regardless the value of $p(b)$, this equation is satisfied. If this is the

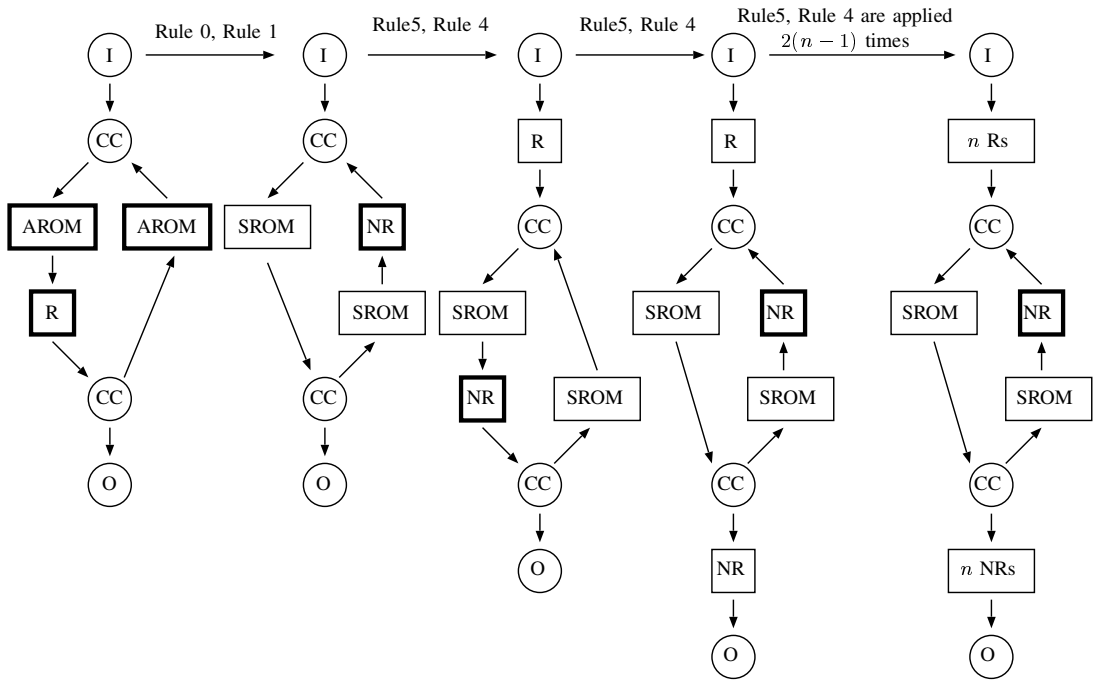


Fig. 11. Example of a circuit for which our rewriting algorithm does not terminate.

case, we assume that $p(b) = k$. We can then determine the potentiality of the other nodes as follows: $p(c) = p(d) = p(f) = k + 1$, and $p(e) = k$. Similarly, from the equation $p(b) = \min(k, p(b))$, we can think that the cycle is a *zero cycle*.

Figure 12 (3) shows an example of a *negative cycle*. We have the equations as follows:

$p(a) = k$, $p(b) = \min(p(a), p(e))$, $p(c) = p(b) - 1$, $p(d) = p(c)$, $p(e) = p(c) - 1$, and $p(f) = p(d)$.

From these equations, we have, $p(b) = \min(k, p(b) - 2)$. If $p(b) \neq k$ then $p(b) = p(b) - 2$. Hence $p(b) = k$ must be satisfied. If this is the case, $p(b) = \min(k, k - 2) = k - 2$, a contradiction. Therefore, $p(b) = \min(k, p(b) - 2)$ has no solution.

From this observation, we define *the potentiality of a cycle* as follows: Let $v_0, v_1, \ldots, v_m (= v_0)$ be a cycle such that there is a directed edge (v_i, v_{i+1}) $(0 \leq i \leq m - 1)$. We define the potentiality $p'(v_i)$ of node v_i $(1 \leq i \leq m)$ with respect to the cycle starting v_0 as follows:

- $p'(v_0) = 0$.
- If v_{i+1} is CC or SROM, then $p'(v_{i+1}) = p'(v_i)$ $(0 \leq i \leq m - 1)$.
- If v_{i+1} is AROM or NR then $p'(v_{i+1}) = p'(v_i) - 1$ $(0 \leq i \leq m - 1)$.
- If v_{i+1} is R then $p'(v_{i+1}) = p'(v_i) + 1$ $(0 \leq i \leq m - 1)$.

We say that *the potentiality of the cycle* is $p'(v_m)$. For example, the potentialities of the cycles in Figure 12 (1), (2), and (3) are 2, 0, and -2, respectively.

We have the following theorem.

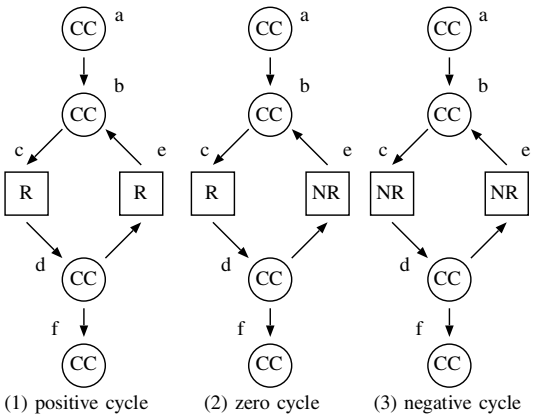


Fig. 12. The potentiality for circuits with a cycle

Theorem 1: Our rewriting algorithm generates an AROM-free and NR-free circuit graph, equivalent to the original circuit graph, if all O nodes and all cycles of a circuit graph have non-negative potentiality.

In other words, we can determine a fully synchronous circuit that can be converted into an AROM-free circuit by evaluating the potentiality of all O nodes and all cycles of the corresponding circuit graph. Also, the potentiality of all O nodes and all cycles are non-negative, our rewriting algorithm generates an AROM-free and NR-free circuit graph, and the corresponding fully synchronous circuit is AROM-free and

equivalent to the original fully synchronous circuit.

V. Proof of Theorem 1

The main purpose of this section is to show a proof of Theorem 1. We will show several lemmas for a proof of Theorem 1.

First, let us observe how the potentiality of nodes is changed by our rewriting algorithm. We focus the potentiality of successor nodes. Let P and S denote the predecessor and successor nodes for Rules 0, 1, 2 and 4. Also, let P_1, P_2, P_3, and S_1, S_2 be the three predecessor and two successor nodes in Rules 3 and 5. We compute the potentiality of each successor node both before and after applying the rules as follows.

Rule 0 $p(S) = p(P) - 1$.
Rule 1 $p(S) = p(P)$.
Rule 2 $p(S) = p(P) + 1$.
Rule 3 $p(S_1) = p(S_2) = \min(p(P_1)+1, p(P_2)+1, p(P_3)+1) = \min(p(P_1), p(P_2), p(P_3)) + 1$.
Rule 4 $p(S) = p(P) - 1$.
Rule 5 $p(S_1) = p(S_2) = \min(p(P_1) - 1, p(P_2), p(P_3)) = \min(p(P_1), p(P_2) + 1, p(P_3) + 1) - 1$.

Thus, the potentiality of every successor node is never changed by applying the rules. In every rule, O nodes can only be successor nodes. Thus, we have,

Lemma 2: The potentiality of every O node of the resulting circuit graph is the same as that of the corresponding O node of the original circuit graph.

Similarly, we can prove the following lemma:

Lemma 3: The potentiality of every cycle of the resulting circuit graph is the same as that of the corresponding cycle of the original circuit graph.

In a circuit graph, let *a segment* be a directed path u_1, u_2, ..., u_m such that, u_1 and u_m are either I, O, SROM, or CC, and u_2, ..., u_{m-1} are either R or NR. Note that, if $m = 2$ then it represents a null segment with u_1, u_2.

We have the following lemma:

Lemma 4: Once our circuit rewriting algorithm uses either Rule 4 or Rule 5 to move an NR node, it never applies Rule 2 and Rule 3 to move an R node.

Proof: If either Rule 4 or Rule 5 is applied an interim circuit, both Rule 2 and Rule 3 cannot be applied to it. If this is the case, all Rs are either (1) in the segment of Rs ending at an O node, or (2) in the segment of Rs ending at an CC node and another incoming edge of the CC node is not connected to R (Figure 13). To apply Rule 2 and Rule 3 later, the non-R node in Figure 13 must be an R node. However, to be an R node, Rule 2 and Rule 3 must be used. Thus, both Rule 2 and Rule 3 are never applied. ■

We will prove that all NRs in a cycle with non-negative potentiality will be removed by our rewriting algorithm.

Lemma 5: Suppose that all cycles in a circuit graph have non-negative potentiality, and Rule 0 are repeatedly applied to remove all AROMs. If a cycle has m NRs, it also has at least m Rs. If either Rule 2 or Rule 3 is applied, the Rs are moved and adjacent R and NR may be removed by Rule 1. If either Rule 4 and Rule 5 is applied, the NRs are moved. Note that, from Lemma 4, the Rs are never moved, once either Rule 4 and Rule 5 is applied. In other words, the NRs are moved along the cycle, while Rs are never moved. Thus, at some point, all NRs in the cycle will be removed by Rule 1.

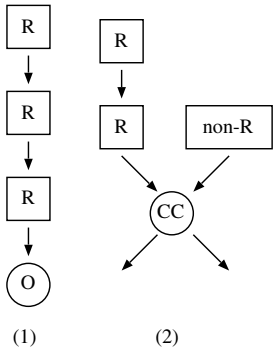


Fig. 13. Illustration for the proof of Lemma 4

Note that, if there exists a cycle with negative potentiality, our circuit rewriting algorithm does not terminate. As illustrated in Figure 11, an NR moves along the cycle and Rs and NRs are repeatedly generated. It should be clear that, there exists no way to generate an equivalent AROM-free circuit for such circuit.

When our rewriting algorithm terminates and the resulting circuit graph is obtained, we have the following lemma:

Lemma 6: Let u be an NR node and (u, v) be its outgoing edge in the resulting circuit graph. Node v must be either NR or O node. Also, all NR nodes must be in segments ending at O node.

Proof: If v is an R, SROM, or CC node then Rules 1, 4, or 5 can be applied. Since no more rule can be applied to the resulting circuit graph, v must be either NR or O nodes. Since the successor of NR nodes must be NR or O nodes, all NR nodes must be in segments ending at O node. ■

A simple directed path is a directed path if it has no repeated nodes. For example, in Figure 2 (2), (B, E, H, K, N, O) is a simple directed path, but $(B, E, H, I, F, E, H, K, N, O)$ is not. We say that nodes are *regular* if it is on a simple directed path from an input node to an output node. Note that nodes on a cycle in a DRG can be a non-regular node. For example, nodes F and I are non-regular nodes.

From Lemma 6, we will prove that all regular SROM and CC nodes in the resulting circuit graph have zero potentiality.

Lemma 7: All regular SROM and CC nodes in the resulting circuit graph have non-negative potentiality.

Proof: Since the resulting graph is AROM-free, nodes follows NR nodes can have negative potentiality. Since no segment ending at SROM or CC have NR nodes, their potentiality must be non-negative. ■

Similarly, we have the following lemma.

Lemma 8: All regular SROM and CC nodes in a simple directed path from an input node to an output node in the

resulting circuit graph have non-positive potentiality.

Proof: We assume that the resulting circuit graph has a positive potentiality SROM or CC node in a simple directed path from an input node to an output node, and show a contradiction. Let v be a first SROM or CC node with negative potentiality, that is, all SROM and CC nodes in all directed paths incoming to v have non-positive potentiality and SROM or CC node v has positive potentiality.

Case 1 v is an SROM node

Let (u, v) denote the incoming edge. If u is either R or NR, then Rule 2 or Rule 4 can be applied. Since no more rule can be applied to the resulting circuit graph, it must be either I, SROM, or CC. If this is the case, $p(u) = 0$ and thus, $p(v) = 0$, a contradiction.

Case 2 v is a CC node

Let $(u_1, v), (u_2, v), \ldots, (u_k, v)$ $(k \geq 1)$ denote the incoming edges. From Lemma 6, none of $u_1, u_2, \ldots, u_k$ is an NR node. If all of them are R nodes, then Rule 3 can be applied. Thus, at least one of them is not an R node. It follows that at least one of them is either I, SROM, or CC node. From the assumption, the potentiality of such node is non-positive, Hence, the potentiality of v is non-positive, a contradiction.

■

We are now in position to show the proof of Theorem 1. From Lemma 7 and 8, all SROM and CC nodes in a simple directed path from an input node to an output node of the resulting circuit graph have zero potentiality. Hence, if the potentiality of one of the O nodes in the resulting circuit graph is negative, a segment ending at O node in the resulting graph should have NR from Lemma 6. Similarly, if the potentiality of all the O nodes is non-negative, no segment ending at an output node has NR in the resulting circuit graph. From Lemma 2, the potentiality of O nodes does not change by our rewriting algorithm. Thus, from Lemma 5, if all output nodes and all cycles of a circuit graph have negative potentiality our rewriting algorithm generates the resulting circuit graph with NR nodes. This completes the proof of Theorem 1.

As we have discussed, our circuit rewriting algorithm does not terminate for a circuit graph with a negative cycle. We can modify our circuit rewriting algorithm that always terminates as follows: First, we compute the potentiality of every cycles. If one of them is negative, we do not execute our circuit rewriting algorithm. Since it is impossible to generate an equivalent AROM-free circuit if this is the case, it is not reasonable to execute our circuit rewriting algorithm.

VI. Conclusions

In this paper, we have presented a rewriting algorithm and six rewriting rules to obtain the equivalent circuits with Synchronous ROMs (SROMs) for the circuits with Asynchronous ROMs (AROMs) including cycles. Using our rewriting algorithm, any sequential circuit with AROMs represented by a directed reachable graph (DRG) can be converted into an equivalent fully synchronous sequential circuit with no AROMs to support the current FPGA architecture. It is not trivial to convert the sequential circuits (cycles included) with AROMs into the equivalent fully synchronous circuits with no AROMs for supporting the modern FPGA. However, our algorithm can do it automatically.

References

[1] J. Bordim, Y. Ito, and K. Nakano. Accelerating the CKY parsing using FPGAs. *IEICE Transactions on Information and Systems*, E86-D(5):803–810, 2003.

[2] J. Bordim, Y. Ito, and K. Nakano. Instance-specific solutions to accelerate the CKY parsing for large context-free grammars. *International Journal on Foundations of Computer Science*, 15(2):403–416, 2004.

[3] S. S. C. Design of an FPGA logic element for implementing asynchronous NULL convention logic circuits. *IEEE Transactions on very large scale integration (VLSI) system*, 15(6):672–683, June 2007.

[4] Y. Ito and K. Nakano. A hardware-software cooperative approach for the exhaustive verification of the Collatz cojecture. In *Proc. of International Symposium on Parallel and Distributed Processing with Applications*, pages 63–70, 2009.

[5] S. Jens. Asynchronous circuit design, a tutorial. http://webee.technion.ac.il/courses/048878/book.pdf.

[6] L. Lavagno and A. Sangiovanni-Vincentelli. *Algorithms for synthesis and testing of asynchronous circuits*. Kluwer Academic, 1993.

[7] R. Manohar. Reconfigurable asynchronous logic. In *Proc. of IEEE Custom Integated Circuits Conference*, pages 13–20, 2006.

[8] M. N. I. Mondal, K. Nakano, and Y. Ito. A rewriting algorithm to generate arom-free fully synchronous circuits. In *Proc. of International Conference on Networking and Computing*, pages 148–157, November 2010.

[9] K. Nakano and Y. Yamagishi. Hardware n choose k counters with applications to the partial exhaustive search. *IEICE Transactions on Information and Systems*, E88-D(7), 2005.

[10] I. Shota, K. Yoshiya, H. Masanori, and K. Michitaka. An asynchronous field-programmable VLSI using LEDR/4-phase-dual-rail protocol converters. In *The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, Monte Carlo Resort, Las Vegas, Nevada, USA, July 2009.

[11] J. Teifel and R. Manohar. An asynchronous dataflow FPGA architecture. *IEEE Transaction on Computers*, 53(11):1376–1392, 2004.