Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Dual-rail asynchronous logic multi-level implementation

Igor Lemberski [a],*, Petr Fišer [b]

[a] Baltic International Academy, Lomonosova 1/24, Riga, LV-1019, Latvia
[b] Czech Technical University in Prague, FIT, Department of Digital Design, Prague, Czech Republic

ARTICLE INFO

Article history:
Received 9 August 2012
Received in revised form
17 November 2012
Accepted 19 February 2013
Available online 27 February 2013

Keywords:
Asynchronous logic
Decomposition
Multi-level implementation
Boolean network
Node

ABSTRACT

A synthesis flow oriented on producing the delay-insensitive dual-rail asynchronous logic is proposed. Within this flow, the existing synchronous logic synthesis tools are exploited to design technology independent single-rail synchronous Boolean network of complex (AND-OR) nodes. Next, the transformation into a dual-rail Boolean network is done. Each node is minimized under the formulated constraint to ensure hazard-free implementation. Then the technology dependent mapping procedure is applied. The MCNC and ISCAS benchmark sets are processed and the area overhead with respect to the synchronous implementation is evaluated. The implementations of the asynchronous logic obtained using the proposed (with AND-OR nodes) and the state-of-the-art (nodes are designed based on DIMS, direct logic and NCL) network structures are compared. A method, where nodes are designed as simple (NAND, NOR, etc.) gates is chosen for a detailed comparison. In our approach, the number of completion detection logic inputs is reduced significantly, since the number of nodes that should be supplied with the completion detection is less than in the case of the network structure that is based on simple gates. As a result, the improvement in sense of the total complexity and performance is obtained.

1. Introduction

Asynchronous logic attracts an increasing interest of designers because asynchronous (delay-insensitive—DI) circuits are extremely robust. This means, the design is able to adapt to variations of manufacturing process parameters, gate and wire delays, temperature changes, noise, etc. [1]. The correct function is guaranteed, only the operational speed changes adaptively. Furthermore, a DI paradigm is very similar to the synchronous one and generally, the DI design process follows the same steps as in synchronous logic design. As a result, the developed DI design flow can be easily incorporated into the design industry, since the tools and design processes are familiar to designers. The DI design process can be easier implemented, since a minimal delay analysis is required to ensure the circuit correct behavior. DI paradigm has additional advantages in designing complex circuits including substantially reduced crosstalk between analog and digital circuits, ease of multi-rate circuits cooperation and facilitation of component reuse [15].

The general disadvantages of DI asynchronous circuits with regard to the synchronous ones are high area and huge power consumption overheads, although the thermal distribution is uniform across the chip.

We propose a synthesis flow of multi-level DI dual-rail implementation. It is based on exploiting synchronous logic synthesis tools to produce a single-rail Boolean network of a two-level (AND-OR) nodes, and further transformation of the network into a dual-rail one. Based on results [18], each node is designed as a hazard-free structure. Finally, the technology-dependent mapping procedure is applied. For the comparison, several state-of-the-art methods are considered, where nodes are designed based on DIMS [13], direct logic [11], and NCL [15]. For the detailed comparison, the method [17], where each single-rail Boolean network node is designed as a simple gate (NAND, NOR, etc.) was chosen. We believe that this method is the closest one to our approach. Although [17] is supposed for designing some other class of circuits, it is clear that the method can easily be adapted for DI logic synthesis. Indeed, the Boolean network [17] is designed as a dual-rail hazard-free logic. The indication of the new input state and internal stability can be done using the completion detection (CD) logic proposed in this paper.

The main disadvantage of the method [17] is a large number of nodes that should be supplied with the CD—the completion detection must be provided for each simple gate. It is not the case of our approach, where the number of nodes of the synthesized Boolean network is significantly less than in [17]. Therefore, in our approach, the CD logic complexity is reduced,

* Corresponding author. Tel.: +371 67 10 06 26; fax: +371 67 24 12 72.
E-mail addresses: Igor.Lemberski@bsa.edu.lv (I. Lemberski), fiserp@fit.cvut.cz (P. Fišer).

0167-9260/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2013.02.002

although the functional logic implementation complexity may be slightly increased (to ensure hazard-free implementation of the AND-OR nodes). As a result, the improvement in sense of the total complexity and performance is obtained. The other approaches to CD optimization can be found in literature. Namely, in [30], the optimization method based on evaluation of the gates relative timing was proposed. In [31], the method in a cost-aware manner was described.

The rest of the paper is organized as follows. In Section 2, the review of the related works is given. The information and notations regarding dual-rail logic is presented in Section 3. Also, details of DI logic behavior rules that are based on Seitz's strong and weak constraints [10] are described. Section 4 is devoted to the description of the model with modified weak constraints. Next, the node minimization constraint to ensure hazard-free implementation is formulated and the structure of the dual-rail network is proposed. Examples illustrating the state-of-the-art and our approaches are given. It is shown that our approach produces networks with significantly less number of signals the CD logic is supplied with. Section 5 describes the technology-independent and technology-dependent synthesis procedure. Experimental results are given in Section 6. Statements summarizing the results conclude the paper.

2. Related work

The asynchronous logic is classified depending on the mode of interaction with the environment [27]. In the input–output mode, the environment is allowed to change the input state once a new output state is produced. There is no assumption about internal signals and the environment is allowed to change the input state before the circuit is stabilized in response to the previous input state.

In the fundamental mode (assumed in this paper, too), the logic operates based on the following discipline: the environment changes the input state once the output state has changed in response to the current input state and each gate inside the circuit is stable. Both design methodologies assume either bounded (a maximal value is known) or unbounded (a maximal value is unknown) gate and wire delays.

In case of the fundamental mode with bounded delays, the moment when the environment may change the input state is estimated based on the worst case propagation delay [3]. Within this model, only one input signal can be changed at a time. In [4], a generalized fundamental mode was proposed, where multiple input changes are allowed during a narrow time interval. For such a mode, a method of hazard-free two-level implementation was published [5]. A multi-level hazard-not-increasing transformation is applied to optimize the implementation [6]. Methods of hazard-free technology mapping were proposed in [7,8].

In case of the unbounded delays, the asynchronous logic should be capable of

(1) recognizing the moment when a new input state (generated by the environment) appears on the inputs and the moment when the circuit generates a new output state in response to the input one and
(2) notifying the environment of new input and output states. After receiving the notification, the environment can generate the next input state.

To solve this problem, m-of-n codes of length n are used for states encoding, where each valid state is represented by ones in m positions and zeroes in the rest of $(n–m)$ ones [2]. Among them, 1-of-2 (or dual-rail) as well as 1-of-4 encodings have been of special interest. The other 1-of-n encodings are rather expensive, since their implementation requires more wires than the dual-rail one. In this paper, the dual-rail state encoding is used.

A four-phase behavior discipline is supposed: to change an input state, the environment should reset it first (change to so called spacer state). The output state resets too, as a result. After that the environment sets a new input state. It implies a new output state. The behavior rule is based on Seitz's strong or weak constraints [9,10]. Under the strong constraints, each output changes its state only when all inputs have changed their state. Under the weak constraints, some outputs are permitted to change their state when some (not all) inputs have changed their state. In the case of strong constraints, output signals also serve as the completion detection ones and indicate the moment when both internal and output signals become stable. In case of weak constraints, output signals may also serve as the completion detection, if they are able to indicate the state of all input signals. Otherwise, an additional completion detection block is required to ensure a proper indication [11]. In [12], the distribution of the completion detection between the outputs is proposed to minimize the implementation cost. Also, the completion detection must indicate the moment when internal signals become stable.

The dual-rail implementation under the four-phase discipline is based on Delay-Insensitive Minterm Synthesis (DIMS) technique [13]. Within it, a function is implemented as a two-level structure with C-elements on the first level and an OR gate on the second one. The DIMS cost is very high, since the minimization of the number of product terms is not allowed. Therefore, 2^k minterms (where k is the number of C-element inputs) must be generated to implement each function's positive and negative forms, where each minterm is implemented using a k-input C-element. Finally, the C-element is more complex than a simple gate.

The implementation of the two-level C-OR logic as a single CMOS gate (Direct Logic) significantly reduces the area [11].

A similar approach is based on using threshold functions (Null Convention Logic-NCL). NCL circuits are designed based on 27 library gates that are capable of implementing any function of four or less inputs [15]. In the case of dual-rail logic, each literal is considered as a separate variable. Therefore, not any single-rail function of more than two inputs can be implemented in NCL; the feasibility of a function implementation depends on the number of literals in its dual-rail representation.

Multi-level implementations of the dual-rail asynchronous logic were proposed in [14] and [17]. These methods are based on the initial circuit decomposition into simple (AND, OR, NOR, NAND, etc.) two-input gates. Further, each gate is mapped into DIMS [13] or implemented by a threshold gate [14]. It results in not only high complexity of the circuit implementation (in sense of the area), but also in a low performance, since each simple gate (single level structure) is implemented as a two-level (AND-OR, C-OR) structure. In [17], each simple gate is doubled to implement dual-rail logic. Compared to the synchronous implementation, the circuit cost doubles.

Desynchronization [16] is a modern paradigm that is based on adopting synchronicity to the asynchronous logic design. If bounded delays are supposed, matched delays are introduced for the synchronization purpose. In case of unbounded delays, extra completion detection logic should be present to indicate the circuit stability. A network of local controllers is designed to provide proper local synchronization signals, resulting in an additional area penalty. Finally, the circuit is equipped with output latches and a new output state is available only once a latch signal enables.

Our approach is based on the combination and extension of methods [15,17]. Namely, we propose the Boolean network multi-level implementation, where complex gates of general nature