iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** IV **Month of publication:** April 2018

DOI: http://doi.org/10.22214/ijraset.2018.4412

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

Asynchronous Circuits Design Using a Field Programmable Gate Array

Aya Nabeel Mowafy[1], Hatem Mohamed Zakaria[2]
[1]*Graduate student, Dept. of Electrical Engineering, Benha Faculty of Engineering, Benha University, Benha, Egypt.*
[2]*Assistant Professor, Dept. of Electrical Engineering, Benha Faculty of Engineering, Benha University, Benha, Egypt.*

Abstract: ***Synchronous design style is based on global timing assumptions determined by the clock. Handling with this assumption, especially in advanced technologies, is problematic as clock trees are gradually consuming more power and need more effort for managing. Asynchronous circuit is an efficient alternative solution for these problems. Therefore, developing robust and fast prototyping systems utilizing conventional Field-Programmable Gate Arrays (FPGAs) can help ease and overcome the difficulties caused by the complexity of asynchronous digital systems. This paper proposes a design flow and a FPGA template for implementing asynchronous components including different C-element style asynchronous controllers. A list of synthesized asynchronous benchmark circuits are documented and discussed in detail. The proposed design flow with FPGA-based realization approach is a very effective design methodology for rapid prototyping and functionality validation. The implemented asynchronous elements designs are tested by using the Virtex®-6 FPGA. This work could be useful for the early stage of performance estimation, circuits design training, and many other applications regarded asynchronous circuits.***
Keywords: ***Asynchronous, Muller, FPGA***

I. INTRODUCTION

The change of internal state in the synchronous sequential circuit depends on the synchronized clock pulses as shown in Fig.1. However, the change of internal state in asynchronous sequential circuits occurs by the change in the input variables and depending on the handshaking messages between multiple blocks [1]-[8].

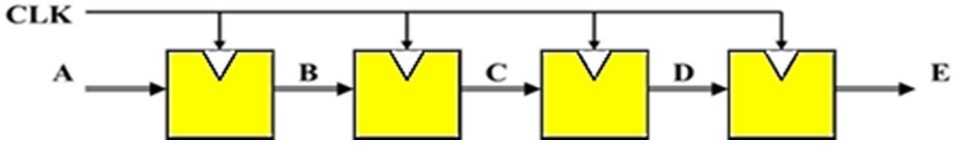


Fig. 1 Synchronous circuit block diagram

A. Synchronous Circuit Designs

The main advantage of the synchronous design is the simplicity of the design because it allows the system to operate at a discrete time. As a result, designers can ignore the problems associated with critical race conditions, unstable state, and feedback issues. Synchronous circuits are widely taught, understood and available components [8]. Therefore, designers only had to make sure that the clock period was large enough to guarantee that the instructions reach a stable state before starting a new state and this a simple way to deal with hazards. On the other hand, the synchronous design has many disadvantages. First, this design is wasting the effective processing time and the speed of the system will be reduced because the synchronous system always operates at the worst-case conditions [1]. Hence, the clock in the synchronous design runs at the speed of its slowest basic processing step. Therefore, any instructions that finish sooner than the next clock edge must wait before starting the next instruction [2]. Second, the synchronous system used a global clock which is connected to the majority of gates that consume a considerable amount of power even if that logic unit is not performing any useful work [3]. Third, due to using a global clock, all transitions occur on a clock edge that is set to a very precise frequency. This results in large current spikes that are concentrated in narrow spectral bands at the harmonics of the clock frequency, causing unwanted electrical noise and electromagnetic interference [2]-[4].

B. Asynchronous Circuit Designs

The asynchronous system has many important advantages. First, this design is high performance and effective processing time as the system is not synchronized to a fixed global clock so a given process can proceed immediately after it completes. Consequently, the system can achieve average-case performance instead of the worst-case performance which a typical synchronous system suffers

©IJRASET (UGC Approved Journal): All Rights are Reserved

from [5]. Second, the system is not synchronized to a fixed global clock (systems are naturally data dependent) causing only activating functional units consume power so asynchronous system reduces power consumption. Third, the system is not synchronized to a fixed global clock so this processor has low noise and reduces the electromagnetic interference. Fourth, one of the main advantages of the asynchronous designs is the small chip size because the global clock which is connected to each logic block will not be used [13]-[15]. Therefore, asynchronous designs are very useful for large-scale systems that depend on a huge number of transistors and this important for modern technology techniques. Unfortunately, asynchronous designs are more complex than traditional synchronous designs, needing more maintenance for checking delays between registers. In asynchronous systems, correctness also includes checking for critical race conditions, unstable states, in addition to stalled conditions that can lead to deadlock and hazards [14]. Therefore, special care must be taken during synthesis to reduce the possibility of function and logic hazards [2]-[6]. Today asynchronous circuits are naturally used in systems where the speed is important, such as signal processing circuits, fast arithmetic unit, and simple microprocessors [17].

This paperwork is organized as follows. Section II introduces a discussion about asynchronous design methodologies. Section III provides the types of asynchronous handshaking. The C-MULLER including the generalized MULLER-C element is proposed in Section IV. Simulation results of the design are reported in Section V. Finally, Section VI states the paper conclusions.

II. ASYNCHRONOUS DESIGN METHODOLOGIES

Asynchronous circuits communicate with handshaking technique. The communicating elements consist of a sender and a receiver part. The sender is the element that initiates the handshake sequence. The sender makes a request to the receiver to perform a certain task. Then the receiver will send an acknowledgment to the sender after it has finished performing the task. Fig.2 shows the handshaking communication [2]-[7]. The asynchronous models can be classified as:

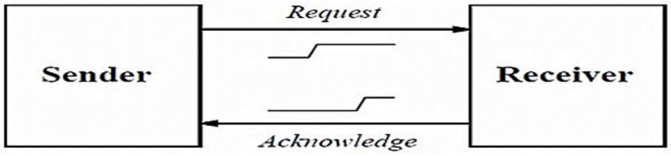


Fig. 2 Handshaking communication technique

A. Delay-Insensitive Timing Model

The delay-insensitive timing model is one of the strongest models of asynchronous designs. These circuits have the variations in both the speeds of the gates and the wire delays as shown in Fig.3 [12]. The gate delay of the logic blocks in addition to the propagation delays of the signal across the wires can be a random length [7]-[2]. Unfortunately, the delay- insensitive circuits are usually limited using in the large-scale designs because the number of logic blocks is great enough that required an acknowledgment from all forked destinations nodes and this would affect badly on the system and cripple all the system performance [7]. If the gate delay of logic blocks B and C is assumed to be of equal length the model is called quasi-delay-insensitive. It simplifies the design process by using the isochronic forks assumption [12]-[16].

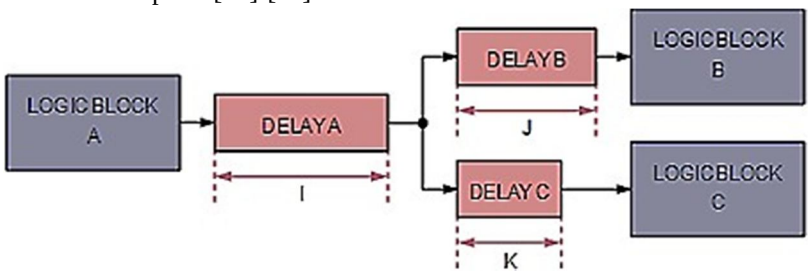


Fig. 3 Delay-insensitive timing model

B. Speed-Independent Timing Model

In the speed-independent timing model, all wire delays are supposed to be negligible as shown in Fig.4. This assumption that makes all wires delays are negligible will decrease the logic complexity [2]. Unfortunately, this assumption can't be generalized to all designs because some designs will need to add extra delays for ensuring the stability of the system. Therefore this model has limited usage.

©IJRASET (UGC Approved Journal): All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

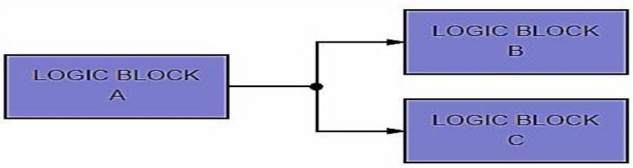


Fig. 4 Speed-independent timing model

C. Bounded-Delay Timing Model

In the bounded delay timing model, the delays in the logic elements in addition to the delays in the wires between them are generally supposed to be bounded and fixed. The main advantage of this model is that asynchronous circuits can be designed in the same way as the synchronous circuits design. For this reason, the change of input will occur after a certain amount of time and this will lead the system to achieve the stability. Unfortunately, this model depends on the worst-case delay and designing each logic block with its worst-case delay will impact the performance. Subsequently, this model is not used in large-scale designs [2]-[16].

III. TYPES OF ASYNCHRONOUS HANDSHAKING

In asynchronous circuits, the handshaking communication takes place between the logic blocks in the next steps. First, the sender will send a request message to the receiver. Second, as soon as the receiver has accepted this request, it will start to carry out the computation. Third, when the operation has ended, the receiver will send an acknowledgment signal to the sender. Finally, after the sender accepts the acknowledgment signal, it knows that the operation has finished and will start a new cycle [17]. Fig.5 shows the handshaking communication between multiple logic blocks.

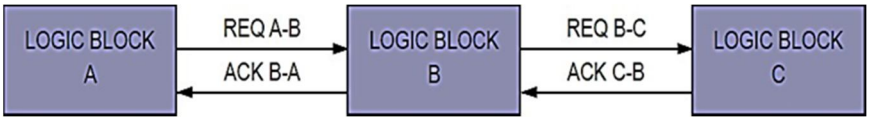


Fig. 5 The handshaking communication between multiple logic blocks [4]

A. The 4-phase handshaking

Fig.6 depicts a 4-phase handshaking communication between the multiple logic blocks of the Fig.5. Once the logic block (A) has valid data to output, it sends a request to logic block (B) by asserting the REQ A-B line. When logic block (B) receives the data, processes the data, and has valid data to output, it also sends a request to logic block (C) by asserting the REQ B-C line. Logic block (C) is the last block in the design so it can send back an acknowledgment to logic block (B) by asserting the ACK C-B line. In the same way, logic block B can propagate the acknowledgment back to logic block A by asserting the ACK B-A line. The return-to-zero (RTZ) phase is the final step in the 4-phase handshaking. Return-to-zero phase used to ensure that all control signals are de-asserted before starting the next cycle [1]-[2].

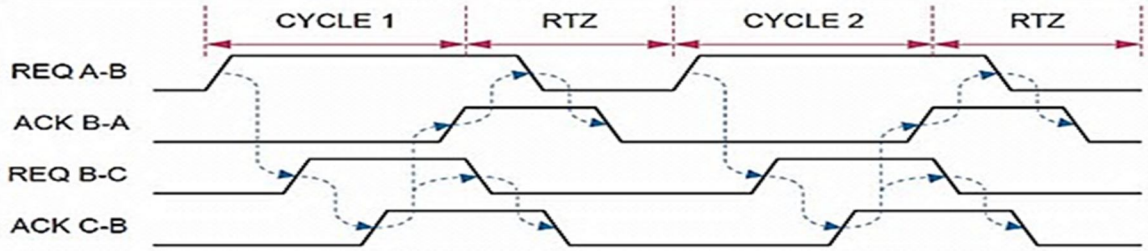


Fig. 6 4-Phase handshaking communication, [4]

B. The 2-phase Handshaking

Fig.7 shows the 2-phase handshaking communication between the multiple logic blocks for the Fig.5. The first part of the 2-phase handshaking is similar to the 4-phase handshaking. Though, the difference in the 2-phase handshaking is that when the acknowledgment signal is propagated back to the start of the pipeline, there is no needing to the return-to-zero phase. The next cycle simply starts by transitioning on the opposite edge of the previous cycle [6].

©IJRASET (UGC Approved Journal): All Rights are Reserved

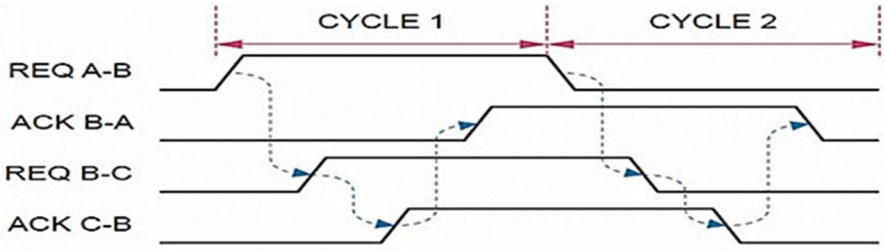


Fig. 7 2-Phase handshaking communication, [4]

C. The 4-phase Handshaking With fast Reset

The return-to-zero phase will impact overall the performance. Therefore, this design used to reduce the period of the return-to-zero phase and that occurs by passing back the final acknowledge to each logic block simultaneously as shown in Fig.8. Fig.9 shows the request and the acknowledgment sequence for the logic block configuration shown in Fig.8. At the initial part of the handshaking, both the original 4-phase handshaking and the modified 4- phase handshaking are the same. The final acknowledgment signal (ACK C-ALL) is passed back to all the blocks at the same time causing the request signals from each block to de-asserted at the same time. This simplification will reduce the wasted time due to reducing the period of the return-to-zero phase and this will lead to improving overall performance [2].

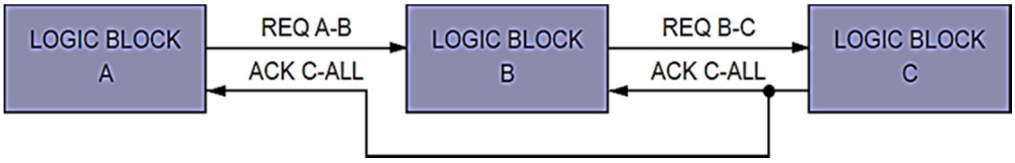


Fig. 8 Block diagram of the 4-phase handshaking with fast reset

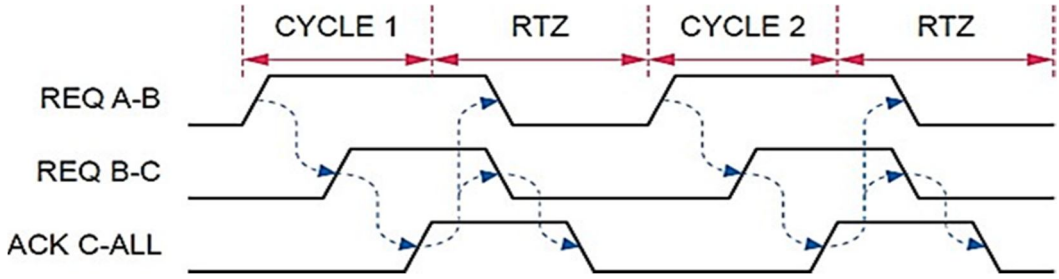


Fig. 9 4-Phase handshaking with fast reset communication

IV. THE C- MULLER

The Muller C-element is a special logic that used in asynchronous circuits for ensuring the stability of the system and assuring that the value is valid before starting a new cycle. So, when both the inputs values change to 1, it asserts the output, when both of the inputs change to 0, it de-asserts the output and if the two inputs are different in the value, it will retain the previous value [10]-[12]. Fig.10 shows the block diagram of Muller C-element.

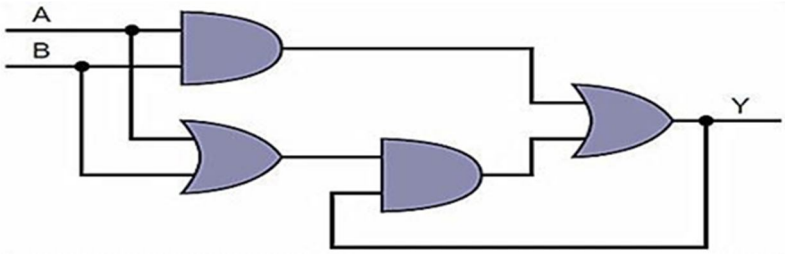


Fig. 10 Muller C-element

©IJRASET (UGC Approved Journal): All Rights are Reserved

Muller C-element can be generalized to be set or reset condition for making them independent of each other. Fig.11 shows the generalized Muller C-element. If all set conditions change to 1, the final output value Y is 1 (asserted). When one or more of the set conditions become 0, the feedback path will keep the output value to retain the previous state. If all the reset conditions change to 1, the final output value Y will be changed to 0 (de-asserted).

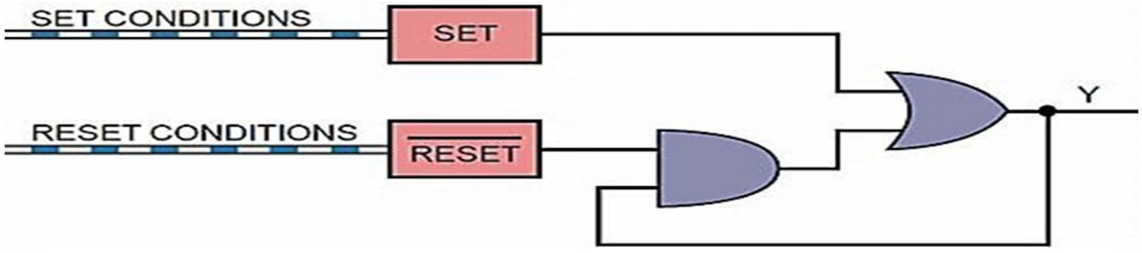


Fig. 11 The generalized Muller C-element

V. ASYNCHRONOUS CIRCUITS SIMULATION AND IMPLEMENTATION USING FPGA

In asynchronous circuits, synchronization is implemented by handshaking protocols which control the communications between the adjacent blocks [9]-[11]. From the structural point of view, asynchronous circuits can be classified into two main categories, Unconditional pipelines, and Conditional pipelines.

A. Unconditional Pipelines

Unconditional pipelines are composed of uncontrolled blocks (Registers, Forks, Join), as depicted in Fig.12 and Fig.13. In Fork, the input token is duplicated and distributed on multi-output channels. On the contrary, if multi-input tokens are processed and injected into a single output channel, the register is called Join.

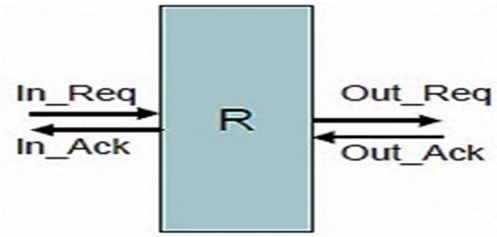


Fig. 12 Asynchronous register

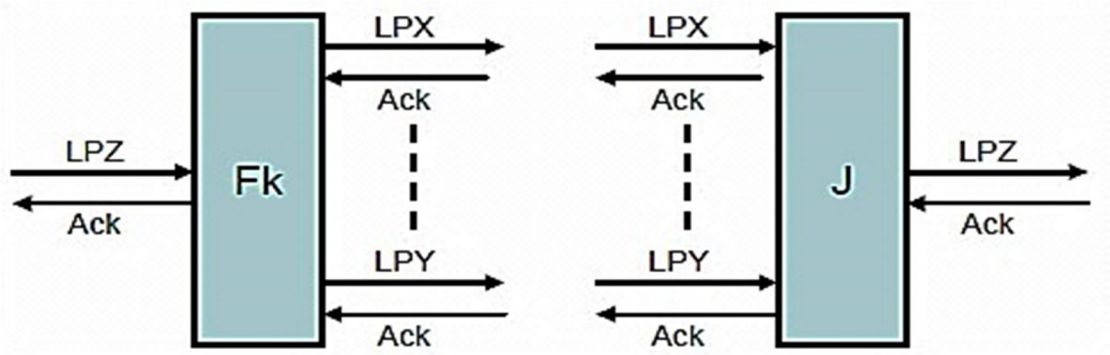


Fig. 13 Fork (FK) and join (J)

B. Conditional Pipelines

Conditional pipelines are basically composed of Splits and Merges as shown in Fig.14. Split injects the input token to a selected output channel (which is determined by the selection control). In contrast, a merge selects a single input token (selected by the Control) to be injected into its output channel. Most of the practical designs involve Splits/Merges in their structures.

©IJRASET (UGC Approved Journal): All Rights are Reserved

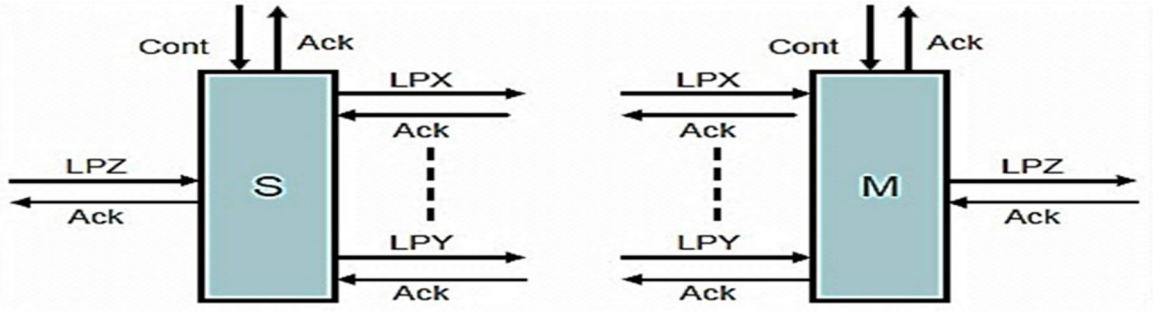


Fig. 14 Split (S) and merge (M)

The basic structures of asynchronous design are simulated using VHDL language. From Fig.15 to Fig.18 show examples of three inputs of C-Muller gates with their simulation results. The first gate describes when all three inputs are "0" value, the output will be "0". When all three inputs are "1" value, the output will be "1" and if three inputs are different values, the output retains its previous value. The second gate shows Muller-C with initialization signal is called "reset". Initially, the input reset will be "0" and this value will force the output value to be "0". Once the reset signal is "1", the output value will change according to the input values. The third gate shows Muller-C with initialization signal is called "set". Initially, the input set will be "1" and this value will force the output value to be "1". Once the set signal is "0", the output value will change according to the input values.

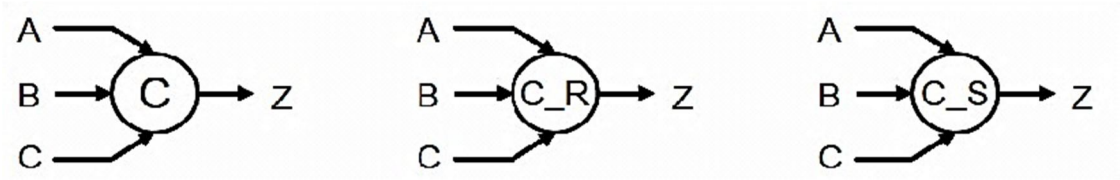


Fig. 15 Examples of 3 inputs C-Muller gates

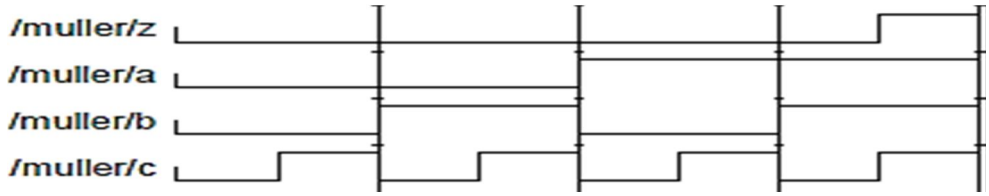


Fig. 16 Timing diagram of the C-Muller

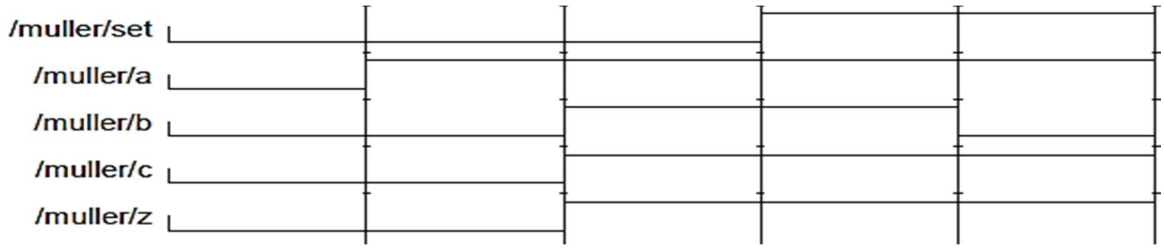


Fig. 17 Timing diagram of the C-Muller with set initialization

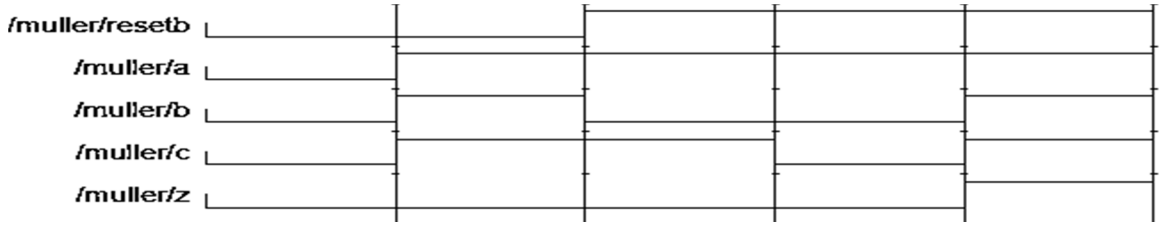


Fig. 18 Timing diagram of the C-Muller with reset initialization

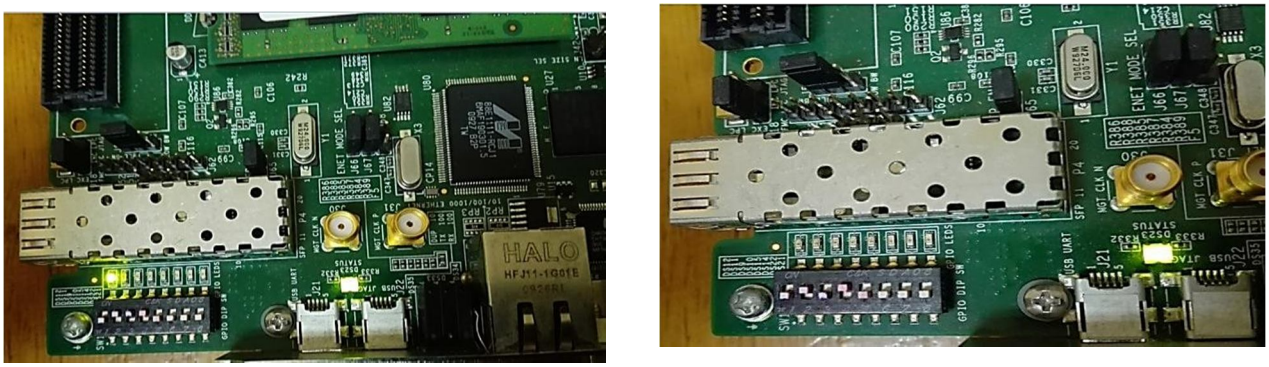
©IJRASET (UGC Approved Journal): All Rights are Reserved

Table1 shows the truth table of C-Muller with set initialization. Fig.19 displays the results of C-Muller with set initialization on FPGA. In a) case, the three inputs (SW2, SW3, and SW4) are 1 while the reset at SW1 is 0, so the output value at LED1 equal 1. In b) case, the two inputs (SW2 and SW4) are 1 while the reset at SW1 is 0, so the output value at LED1 equal 0.

TABLE I

Inputs				Output
SET (SW1)	A (SW2)	B (SW3)	C (SW4)	Z (LED1)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TRUTH TABLE OF C-MULLER WITH SET INITIALIZATION



Valid output (LED1=1) b) Invalid output (LED1=0)

Fig. 19 The implementation of C-Muller with set initialization on FPGA

Fig.20 shows the timing diagram of the register. Initially, the input reset will be "1" and this value will force the output values to be (Out_Req = "0" and In_Ack = "1"). When the reset signal is "0" and the input values are (In_Req = Out_Ack = "1"), then the values of outputs will be (Out_Req ="1" and In_Ack = "0"). While the reset signal is "0" and the input values are (In_Req = Out_Ack = "0"), then the values of outputs will be (Out_Req ="0" and In_Ack = "1"). If the reset value = "0" and the inputs are different values, then the output values will retain their previous values.

©IJRASET (UGC Approved Journal): All Rights are Reserved

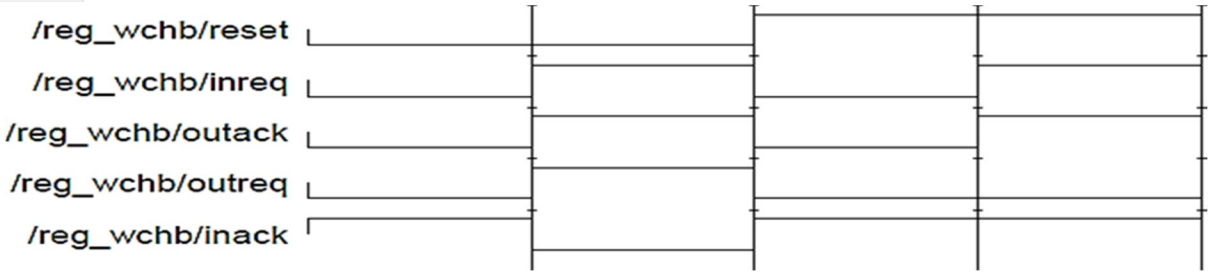


Fig. 20 Timing diagram of the asynchronous register

Fig.21 shows the timing diagram of the merge. Initially, the input reset will be "1" and this value will force the output values to be (In_Ack1 = "1", In_Ack2 = "1", Cont_Ack = "1" and Out_Req = "0"). When the reset = "0" and input values are (Out_Ack = "1", Cont_Val = "1" and In_Req1 = "1"), then the output values will be (Out_Req = "1" and In_Ack1 = "0"). If the reset = "0" and input values are (Out_Ack = "1", Cont_Val = "2" and In_Req2 = "1"), then the output values will be (Out_Req = "1" and In_Ack2 = "0"). When the reset = "0" and input values are (Cont_Val = "1" and In_Req1 = "0"), then the output values will be (Out_Req = "0", In_Ack1 = "1" and Cont_Ack = "0"). If the reset = "0" and input values are (Cont_Val = "2" and In_Req2 = "0"), then the output values will be (Out_Req = "0", In_Ack2 = "1" and Cont_Ack = "0"). Finally if the reset = "0" and input values are (Cont_Val = "0" and Out_Ack = "1") then the output will be (Cont_Ack = "1").

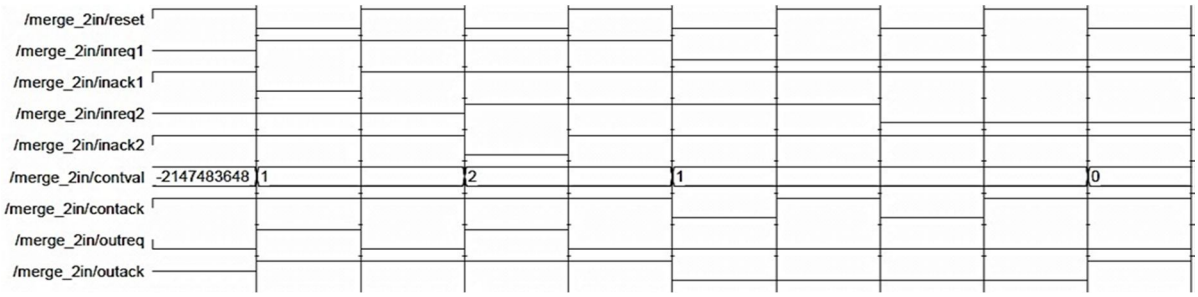


Fig. 21 Timing diagram of the asynchronous merge

Table 2 shows the truth table of asynchronous merge. Fig.22 displays the results of asynchronous merge on FPGA. In Fig.22-a, the default state when (SW1 = 1), so the output values are (LED1 = 1, LED2 = 0, LED3 = 1 and LED4 = 1). Fig.22-b, the reset at (SW1 = 0) and (SW2 = 1, SW3 = 1, SW4 = 1, SW5 = 0 and SW6 = 0), so the output values are (LED1 = 1, LED2 = 1, LED3 = 0 and LED4= 1). In Fig.22-c, the reset at (SW1 = 0) and (SW2 = 0, SW3 = 1, SW4 = 0, SW5 = 1 and SW6 = 1), so the output values are (LED1 = 1, LED2 = 1, LED3 = 1 and LED4 = 0).

TABLE II

Inputs						Output			
RESET (SW1)	CONTVAL1 (SW2)	OUTACK (SW3)	INREQ1 (SW4)	INREQ2 (SW5)	CONTVAL2 (SW6)	CONTACK (LED1)	OUTREQ (LED2)	INACK1 (LED3)	INACK2 (LED4)
1	0	0	0	0	0	1	0	1	1
0	1	1	1	0	0	1	1	0	1
0	0	1	0	1	1	1	1	1	0
0	1	0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0	1	1

TRUTH TABLE OF ASYNCHRONOUS MERGE

©IJRASET (UGC Approved Journal): All Rights are Reserved

(case1) b) (case2)

c) (case3)

Fig. 22 The implementations of asynchronous merge on FPGA

Fig.23 shows the timing diagram of the split. Initially, the input reset will be "1" and this value will force the output values to be (In_Ack = "1", Cont_Ack = "1" and Out_Req1 = Out_Rrq2= "0"). When the reset = "0" and input values are (In_Req = "1", Cont_Ack = "1", Cont_Val = "1" and out_ack1 = "1") then the output values will be (Out_Req1 = "1" and In_Ack = "0"). If the reset = "0" and input values are (In_Req = "1", Cont_Ack = "1", Cont_Val = "1" and out_ack2 = "1"), then the output values will be (Out_Req2 = "1" and In_Ack = "0"). If the reset = "0" and input values are (In_Req = "0", Cont_Ack = "1", Cont_Val = "1" and out_ack1 = "0") then the output values will be (Out_Req1 = "0", In_Ack = "1" and Cont_Ack = "0"). If the reset = "0" and input values are (In_Req = "1", Cont_Ack = "1", Cont_Val = "2" and out_ack2 = "0"), then the output values will be (Out_Req2 = "1", In_Ack = "0" and Cont_Ack = "0"). Finally if the reset = "0" and input values are (Cont_Val = "0" and Out_Ack1 = Out_Ack2 = "1") then the output value will be (Cont_Ack = "1").

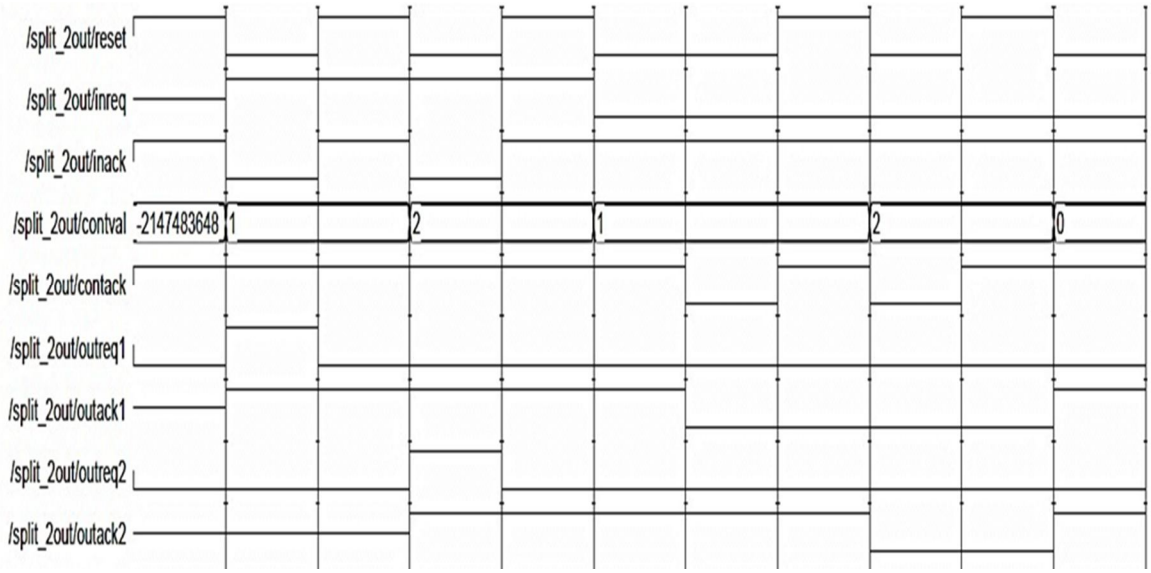


Fig. 23 Timing diagram of the asynchronous split

©IJRASET (UGC Approved Journal): All Rights are Reserved

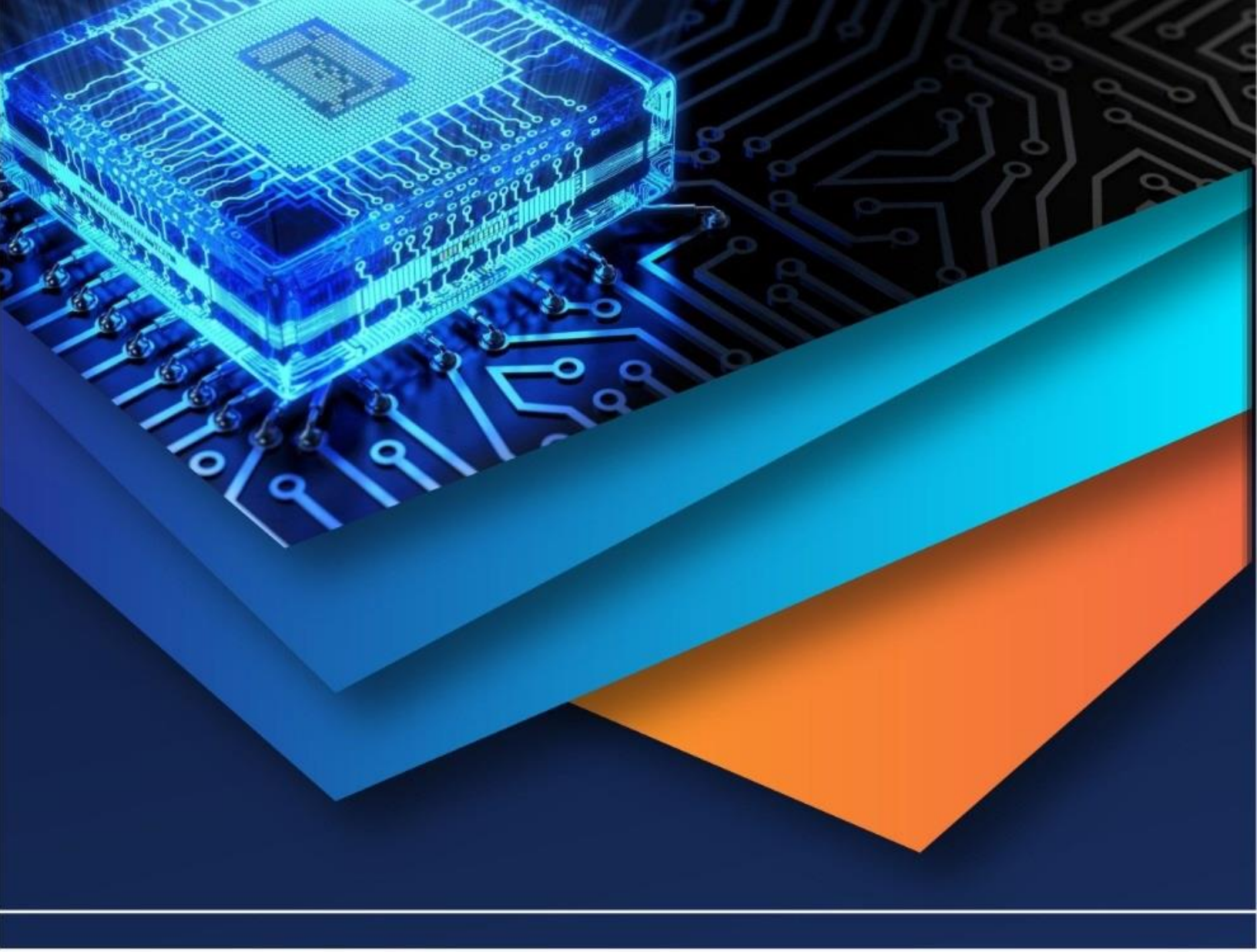
VI. CONCLUSIONS

Synchronous systems have several restrictions due to the increased complexity, clock skew problems, high power consumption, growing in electromagnetic interference, dealing with the worst-case instructions causing low speed and low performance. Therefore, this paper is interested in asynchronous designs to overcome the problems of synchronous designs and overcome the impacts of the clock. Asynchronous designs are very useful in the applications that the speed of the system is very important. In these systems, the operations responded quickly without the need of waiting for the clock pulse. So, asynchronous designs are used in applications that the input signals need to change at any time and run with an independent manner of the internal clock. This paper discusses the simulation and the implementation of different asynchronous components using the C-Muller element. The proposed implemented asynchronous element designs are tested and validated by using the Virtex®-6 Field-Programmable gate arrays.

REFERENCES

[1] M. T. Moreira and N. L. Vilar, "Proposal of an Exploration of Asynchronous Circuits Templates and Their Applications," Faculdade de Informática Pucrs, Brazil, March, 2014.
[2] Robert Hoshino," Investigating the Feasibility of Asynchronous Design Methodologies in Large-Scale Microprocessor Systems," Queen's University Kingston, Ontario, Canada, December 2007.
[3] S. Furber, "Kicking out the Clock," Integrated System Design, EE Design, May 2000.
[4] D. Fritz, "Synchronous Interconnect is Hitting the Wall," Silistix Ltd., San Jose, CA, October 2006.
[5] R. Webb, "Asynchronous MIPS Processors: Educational Simulations," The Faculty of California Polytechnic State University, July 2010.
[6] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design," The hardware/Software Interface, Morgan Kaufmann, 2005.
[7] Jens Sparsø, "Asynchronous Circuit Design," Technical University of Denmark, 2006.
[8] Jo L. Hennessy, David A. Patterson, "Computer Architecture A Quantitative Approach," Third Edition, 2006.
[9] S. Nowick, M. Singh, "High-Performance Asynchronous Pipelines: An Overview," IEEE Design & Test of Computers, 28(5), pp. 8-22, 2011.
[10] C. Myers, "Asynchronous Circuit Design," New York, John Wiley & Sons, Inc., 422p, 2001.
[11] P. Beerel, R. Ozdag, M. Ferretti, "A Designer's Guide to Asynchronous VLSI," Cambridge University Press, 2010.
[12] J. Sparso and S. Furber, "Principles of Asynchronous Circuit Design: A Systems Perspective", Dordrecht, Netherlands: Kluwer Academic Publishers, 2001.
[13] Dr. Dilip Kumar and Kirat Pal Singh, "Design of High performance MIPS-32 Pipeline Processor", 2012.
[14] 1Kirat Pal Singh and 2Shivani Parmar, "Vhdl Implementation of a MIPS-32 Pipeline Processor," International Journal of Applied Engineering Research, ISSN 0973-4562 vol. 7 no.11, 2012.
[15] James R.Larus and Daniel J.Sorin, "Computer organization and design", university of California, Berkeley, Stanford University, 2005.
[16] Jens SparsØ, Steve Furber, "Principles of Asynchronous Circuit Design. A System Perspective". Kluwer Academic Publishers, 2001.
[17] Muzaffer Can, " Pipelined Design Approach to Microprocessor Architectures a Partial Implementation: MIPS™ Pipelined Architecture on FPGA," December 2005.

©IJRASET (UGC Approved Journal): All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)