Implementation of an Asynchronous Micro-controller on the Commercial FPGA

Ziho Shin, Myeong-Hoon Oh, Hyukje Kwon, Hagyoung Kim, and Dongjae Kang

***Abstract*—An asynchronous circuit design methodology has been introduced as a novelty approach to future digital system design. Nevertheless, in order to implement the asynchronous circuit, there are various limitations. Especially, for the implementation on a commercialized field programmable gate array (FPGA), the vendors and design tools mainly support only synchronous circuit design. In this paper, we propose design techniques for implementing the asynchronous circuit on the commercial FPGA using the provided design tool. Then, with the proposed design techniques, we designed an asynchronous micro-controller based onaTIMSP430 instruction set architecture (ISA).We observe that the asynchronous core consumes lower power than the synchronous one. In addition, the asynchronous core also shows much more durability under conditions of unstable power supplies compared to the synchronous counterpart.**

***Index Terms*—Asynchronous circuit, AFSM, bounded delay, FPGA.**

I. Introduction

The technology of an internet of things which is called "IoT" is prevalent in these days [1], [2]. Using the IoT devices, mostly they can be placed in an unstable circumstance that is easily influenced by external factors such as shortage of power and variation of temperature. Thus, operational stability and low-power consumption have been recognized as critical features in the sphere of the processor for the end-nodes in the IoT environments. The processor for the IoT applications are mainly composed of a small processor such as 8-bitor16-bit micro-controllers with open hardware platform [3]-[5]. By virtue of the target applications are not required high-performance computation [6]. For these introduced processors, there were largely designed by traditional circuit design methodologies, which are a synchronous circuit design.

Whereasprevailing synchronous design methodologies have limitations on a large amount of clock network power consumption, unreliable performance from clock skew, and meta-stability problem from the multiple clock domains owing to a single global clock signal [7]. Those can contrast with significant characteristics for the field of IoT devices.

Asynchronous circuit design methodologies can change the design paradigm from the synchronous one. Since an asynchronous circuit does not have any globalized control signal inherently, this design methodology can be fundamentally free from the above-mentioned drawbacks. Instead of a global clock, the asynchronous circuit can guarantee its stability and robustness of functionality by performing handshake protocol for the localized synchronization. Therefore, the power consumption of the processor is reduced due to the elimination of the clock network, which accounts for 70% of the entire power source [8], in modern processors. And, thus, the asynchronous circuit can have low-power characteristics inherently [9]. Additionally, the asynchronous circuit has less emission of electro-magnetic noise (EMI) [9] feature because the asynchronous design methodologies do not employee globalized common and periodic control signal. These are reasons why [10]-[12] were introduced.

Yet, on the implementation point of view, designers could face diverse problems, when they try to implement the asynchronous circuit. Because commercial vendors and design methodologies are largely focused on synchronous circuit design. So as to the asynchronous circuit still has fascinating benefits, researches on design methodologies and implementation techniques are highly needed.

Thus, in this paper, we propose design techniques for the asynchronous circuit implementation on the commercial field programmable gate array (FPGA) and we designed an MSP430 core for the IoT applications. Then, the performance comparison with a synchronous type will address.

This paper is organized as follows: The next section talks about asynchronous circuit design and design issues on the commercial FPGA. Subsequently, feasible solutions for according to that of limitations will be introduced. And the following section introduces an architecture for the designed core. And Section V describes experimental results of the asynchronous circuit and its competitive design. Lastly, in the Section VI conclusion and future works will present.

II. Related Works

A. Delay Model and Signaling in Asynchronous Circuit

The asynchronous circuit design methodologies were introduced as an alternative approach for the synchronous one in the field of the digital system design.

Unlikely to the synchronous one, the asynchronous circuit does not utilize globalized control signal but it employee handshake protocol for the localized synchronization and on-demand processing [13]. The handshake protocol uses a request (Req) signal, which indicates data validity, and an

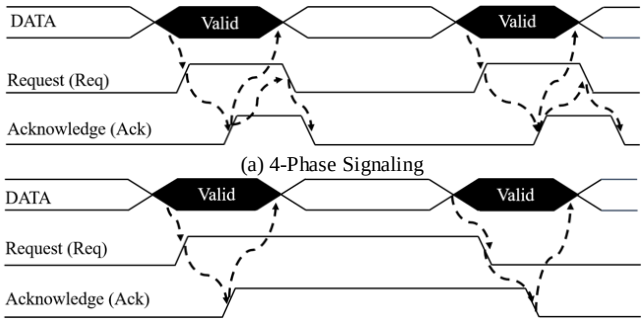
Manuscript received May 10, 2017; revised October 23, 2017. This work was supported by the ICT R&D program of MSIP/IITP. [2014-0-00050, Low-power and High-density Micro Server System Development for Cloud Infrastructure].

Ziho Shin, Myeong-Hoon Oh, and Dongjae Kang are with Cloud Computing Research Group, Electronics and Telecommunication Research Institute (ETRI), Daejeon, Republic of Korea and Dept. of Computer S/W, University of Science and Technology (UST), Daejeon, Republic of Korea (e-mail: zshin@ ust.ac.kr , mhoonoh@etri.re.kr, djkang@etri.re.kr).

Hyukje Kwon and Hag Young Kim are with Cloud Computing Research Group, Electronics and Telecommunication Research Institute (ETRI), Daejeon, Republic of Korea (e-mail: heavenwing@etri.re.kr, h0kim@etri.re.kr).

DOI: 10.7763/IJCTE.2017.V9.1188

acknowledgment (Ack) signal, which represents the completion of data delivery. Pursuant to the handshake protocol, the asynchronous circuit can be divided into two categories which are shown in Fig. 1: 4-phase signaling (a) and 2-phase signaling (b).For the 4-phase signaling, it only uses rising edges of Req and Ack signals. So that 4-phase signaling needs return to zero phase. On the other hand, 2-phase signaling utilizes both rising and falling edges.



(a) 4-Phase Signaling

(b) 2-Phase Signaling

Fig. 1.The handshake protocol.

From the delay model perspective, the asynchronous circuit can also be divided into a bounded delays model, a delay insensitive model, and a speed independent model. For each delay model has its own distinctive modeling strategy for the delay of gates and wires. Which are represented in Table I [14].

TABLE I: DELAY MODELING

Delay Model	Wire	Gates
Bounded Delay	Bounded	Bounded
Delay Insensitive	Un-Bounded	Un-Bounded
Speed Independent	Zero-Delay	Un-Bounded

B. Implementation Methodologies in Asynchronous Circuit

Due to the relatively simple design methodology, which is the nature of synchronous circuit design, the commercialized computer aided design (CAD) tools mainly focused on synchronous circuit design scheme. Since then, the design archetype has not ever been shifted.

However, the user experience request novelty technology, which can provide multitudinous functions with the low-powered and high-reliable process, specifically in the sphere of the IoT. In order to meet these demands, design methodologies need to be changed which can solve botheration of traditional design methodologies.

The design process of the asynchronous circuit has thoroughly differed from the synchronous circuit design. Because the asynchronous circuit does not have any globalized control signal inherently, optimization and control path design strategy could not be same with synchronous circuit one [15]-[18].

A Petri-Net based signal transition graph (STG) representation [19] supports speed-independence model and for the synthesis of an STG-based controller, Petrify [20] has been widely used. Due to the STG is controlled by the transition of the input signal, the arbiter cells are required on the implementation stage, to select the appropriate control signal. This feature can increase the design complexity.

To design the delay insensitive model, the designers should use multi-bit data encoding scheme. By reason of, the Req signal from the handshake protocol is embedded into the multi-bit encoded data line. Thus, dual-rail or quad-rail encoding strategy could be utilized. A null convention logic (NCL) was introduced [21] as an implementation technique for the delay insensitive model. To get the delay insensitive characteristic from the NCL design, the designers have to use special cells, which are called NCL gates (Threshold gates (TH cells)).

The bounded delay model can be implemented by the asynchronous finite state machine (AFSM) [22] which is similar to Mealy machine styled FSM. To support the synthesis of the AFSM design, 3D [23] has been introduced. To implement the AFSM controller, matched delay cells should be inserted to the control signal. While the delay of wires and gates could not be modeled as zero/infinite value in this model.

In this paper, we target the bounded delay model with 4-phase handshake protocol. Since targeted delay model and handshake protocol analogous with synchronous one. This characteristic can reduce design complexity of the asynchronous circuit on the FPGA implementation. Subsequently, in order to optimize control logic, we utilize the 3D tool which can support directed ‘don’t care’ states and conditional branches [23].

Additionally, to verify our design techniques, we designed the MSP430 core [24], which is 16-bit processor known as low-powered, simple instruction set architecture (ISA) and open-compiling environments in the field of IoT.

C. Issues on FPGA Implementation in Asynchronous Circuit

The FPGAs have been recognized as relatively simple/easy and inexpensive implementation methodology compared to a full-custom design. However, the legacy FPGAs and their development tools are mainly focused on synchronous circuit design. Hence, [25], [26] were studied as the alternative techniques for implementing the asynchronous circuit on the FPGA. Nonetheless, introduced techniques were not targeted solutions for the bounded delay model neither they were not designed micro-controller through proposed techniques.

Thus, in this paper, we suggest useful techniques, which can overcome encountered design issues, for the implementation of bounded delay model on the commercialized FPGA (focused on Xilinx FPGA [27] with ISE).

Also, to verify proposed techniques, we designed and implemented16-bit micro-controller on the Xilinx FPGA.

III. PROPOSED DESIGN TECHNIQUES

For the design of the bounded delay model, which is based on the AFSM controller, the 3D tool can be used for the purpose of logic level synthesis. Then, the designers need to perform a mapping process of the given gate level net-list from the 3D tool to the user specific technology library. However, gate mapping on the FPGA level differs from full-custom design. Because of the characteristic of the

FPGA, there are limitations on implementing gate level of design and meeting the variation of timing condition according to various FPGA types provided from the different FPGA vendors.

Furthermore, the synthesized result from the 3D tool has lots of feedback signals. Hence, if the hierarchy of gate configuration does not match to the timing of the feedback line, due to a race condition, the functionality of entire control module cannot be guaranteed [28].

To avoid the race problems and matching the timing conditions for each control module, the designers can utilize look-up table (LUT)-based coding style, which supports logical equation on the commercial FPGAs. Also, since commercialized FPGA tools usually remove the feedback lines on the step of the optimization process, the designers should give "*KEEP*" option [29] to each wire. So that the feedback lines are restricted not to be removed in the optimization process. The Fig. 2 shows the pseudo code of suggested LUT-based design technique for implementation of the AFSM controller.

```
(*keep = "TRUE"*) wire zzz00;
(*LUT_MAP = "yes"*)
assign zzz00 = ((nextIFID) | (IRwriteDone) | (zzz05) | ((IF1) & (!DecodeDone) &
               (!nextOF1OK) & (!zzz01)) | ((IF2) & (!DecodeDone) & (!nextOF2OK) &
               (!zzz02)) | ((IF2) & (!DecodeDone) & (!nextEXWB1OK) & (!zzz03)) |
               ((!MemoryReadAck) & (!DecodeDone) & (!nextOF1OK) & (!nextOF2OK) &
               (!nextEXWB1OK) & (zzz00))) & (!reset);
                                    ...
(* LUT_MAP="yes" *)
assign nextOF2ack_1 = ((nextOF2ack) & (zzz00) & (zzz01) &
                       (zzz02) & (zzz03));
orZSHIN nextOF2ack_pre_or ( // User definded LUT based OR gate
                            .a   (nextOF2req      ),
                            .b   (nextOF2ack_1    ),
                            .out (nextOF2ack_pre  )
                            );
andZSHIN nextOF2ack_and ( // User definded LUT based AND gate
                          .a   (nextOF2ack_pre ),
                          .b   (!reset         ),
                          .out (nextOF2ack     )
                          );
```

Fig. 2. Pseudo code of the AFSM controller.

The asynchronous circuit does not synchronize globally but locally synchronize. So as to synchronize with its neighbored modules, as shown in Fig. 3, the C-element is required to operate localized synchronization. For the design of C-element, there are two types: Based on RS-Latch style and generalized C-element (latch free style) [13] (Described in Fig. 4).

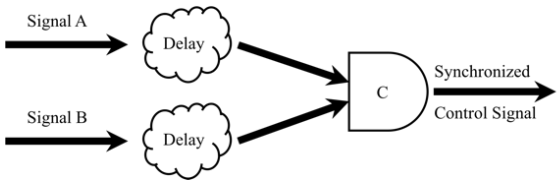
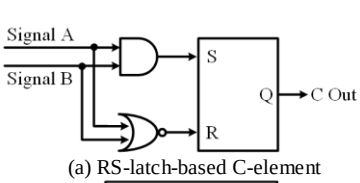
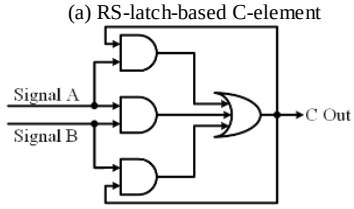


Fig. 3. The usage of the C-element.



(a) RS-latch-based C-element



(b) Generalized C-element

Fig. 4. Types of the C-elements.

Nevertheless, the commercial FPGA vendors do not support generalized C-element neither RS-latch styled C-element. In this paper, we utilize generalized C-element, in order to reduce power consumption of control logic as well as area occupation.

To implement the generalized C-element, the designers can directly initiate the LUT. Following Table II represents a truth table of the generalized C-element.

TABLE II: TRUTH TABLE OF THE GENERAL C-ELEMENT

A	B	Δ Out	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Thus, for implementing the generalized C-element, which has two input signal with low reset, the equation can be expressed as in (1). Therefore, to initialize 4 input LUT, the "INIT" code will be 16'h00E8.

$$Out = \{(A \,\&\, Out \,\&\, \overline{reset}) + (B \,\&\, Out \,\&\, \overline{reset}) + (A \,\&\, B \,\&\, \overline{reset})\} \quad (1)$$

The Fig. 5 describes written code for both RS-latch styled C-element and generalized C-element.

Moreover, to solve meta-stability conditions in the asynchronous circuit system, the arbiter cell (MUTEX) is required: When more than 2 control signals are asserted, the arbiter should make a decision for the priority of each control signal. To implement the MUTEX as the meta-stability filter, simultaneously excited control signals pass through the NAND gates with others feedback line. And the output of NAND gates sent to the AND gates [13].

Fig. 6 describes possible implementation method for the MUTEX cell. For all of the gates in the Fig. 6 configured by LUT equations (Likewise in the Fig. 5 (b)) with "KEEP" constraints to retain its functionality.

The bounded delay model is comparable with the synchronous circuit. However, in order to achieve asynchrony of the bounded delay model, the handshake signal from the asynchronous controller need to be matched to the respective data path module. Therefore, worst-case delay of data path requires to be calculated and the calculated delay should be inserted as a "Matched delay cell".

In the case of implementing the matched delay cell, formally, the designers employee chains of "INVERTER" or "AND" gates as shown in Fig. 7. However, on the commercial FPGAs, the delay cells do not provide. Also, if the designers make gate chain, the optimizer easily remove that logic. Owing to accomplish matching the delays of the respective data path, the designers need to make user specific delay cells by invoking the LUT gates [30] with "KEEP" constraints. After that, the designers require to measure the delay of written LUT gates manually on the post-route simulation level since, the timing of written LUT based delay cell would vary, depending on which FPGA model was used.

```
module GeneralC ( reset, A1, A2, Cout); // Generalized C-element
input reset, A1, A2;
output Cout;
        LUT4 #(.INIT(16'h00E8)) // Specify LUT Contents
        LUT4_inst (
        .O (Cout),
        .I0 (A1),
        .I1 (A2),
        .I3 (reset)
        );
endmodule
```

(a) Generalized C-element

```
module RSLatchC (reset, A1, A2, Cout); // RS-Latch styled C-element
input reset, A1, A2;
output Cout;
(*keep = "TRUE"*) wire LUTset;
(*keep = "TRUE"*) wire LUTreset;
(*keep = "TRUE"*) wire invertedReset;
(*keep = "TRUE"*) wire Cout_bar;

        LUT2 #(.INIT(4'h8)) // Specify LUT Contents
        LUTset_inst (
        .O (LUTset),
        .I0 (A1),
        .I1 (A2)
        );

        LUT1 #(. INIT(2'h1)) // Specify LUT Contents
        LUTinv_inst (
        .O (invertedReset),
        .I0 (reset)
        );

        LUT3 #(. INIT(8'h01)) // Specify LUT Contents
        LUTor_inst (
        .O (LUTreset),
        .I0 (A1),
        .I1 (A2),
        .I2 (invertedReset)
        );

        LUT2 #(. INIT(4'h1)) // Specify LUT Contents
        LUTnor1_inst (
        .O (Cout),
        .I0 (LUTreset),
        .I1 (Cout_bar)
        );

        LUT2 #(. INIT(4'h1)) // Specify LUT Contents
        LUTnor2_inst (
        .O (Cout_bar),
        .I0 (LUTset),
        .I1 (Cout)
        );

endmodule
```

(b)RS-Latch styled C-element

Fig. 5. Implementation of C-elements.

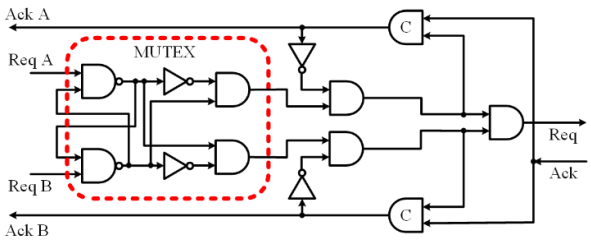


Fig. 6. Concurrent control signal handler (MUTEX) [13].

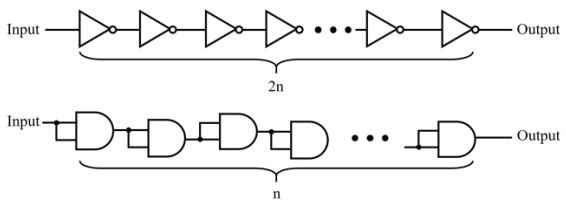


Fig. 7. Matched delay cells: Typical method.

To make a processor, some peripherals such as block memories and input/output ports are needed. Especially, the block memories are highly required since our processor architecture still uses traditional styled one (Harvard/Von Neumann Architectures) [31]. This is attributed to the fact that, the designers could write block memories, like ROM and RAM, own their purpose. However, using the user-defined memory is not an efficient way to implement block memories, because vendors already provide block devices as an IP or embedded one [27].

Consequently, to utilize embedded resources, the interfacing technique for the synchronous block between asynchronous one is decidedly demanded.

To interface among the asynchronous circuit and synchronous block, the most important thing is signal matching for each control signal. And, hence, handshake protocol of the asynchronous circuit should be mapped to the clock gating control signal for the synchronous block. Thus, according to the rising edges of handshake protocol, the clock signal can be gated. Further, for the 4-phase handshake signaling, the Ack signal needs to be generated from the interfacing module. Hence, when the interfacing unit detects Req signal, the Req signal passes through the LUT based matched delay element as described in Fig. 8.

The generated Ack signal must be matched to the accordance of a response time of the synchronous block. To be specific, if the delayed signal does not match to timing constraints (Set-up/Hold time regulations of latches/Flip-Flops) of the synchronous block, the asynchronous circuit cannot be guaranteed the validity of fetched data.

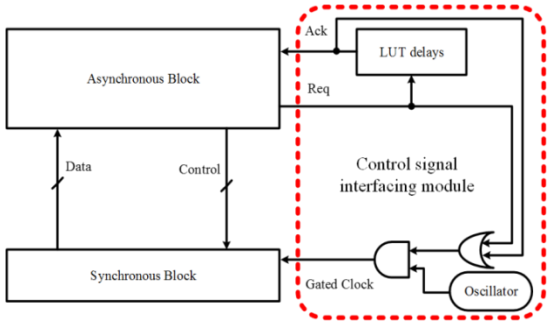


Fig. 8. Basic concept of interfacing module.

IV. Design of Asynchronous MSP 430 on FPGA

The MSP430 core executes 27 reduced instruction set computer (RISC)-type instructions and it supports 7 addressing modes. Theoretically, every instruction can use all the addressing modes without any restriction. Thus, instead of RISC styled design, a complex instruction set computer (CISC) styled design is more suitable for the MSP 430 core [32] in order to support various addressing mode and various opcode sizes for each instruction.

In this paper, we designed the MSP430 core in two different types:

1) Asynchronous MSP430: The control path designed as the bounded delay with 4-phase handshake protocol-based AFSM controller (AMSP430).
2) Synchronous MSP430: The control path designed as an FSM-based controller (SMSP430).

These cores are implemented on the commercial Xilinx Spartan-3 FPGA (xc3s400-5tq144) [33]. The data path is designed as Fig. 9 [32] and it is shared by each AMSP430 and SMSP430.

The AFSM controller for the AMSP430 is synthesized with the 3D tool for the elimination of excessive restrictions

on concurrent events. Afterward, the synthesized gate-level logic equations mapped into the LUT gates through the proposed design techniques.

Afterward, through reviewing the technology schematic, which is provided by Xilinx ISE, we verified that our proposed interfacing logic between the synchronous block and the asynchronous block is well glued. The Fig. 10 shows the technology schematic of designed system.

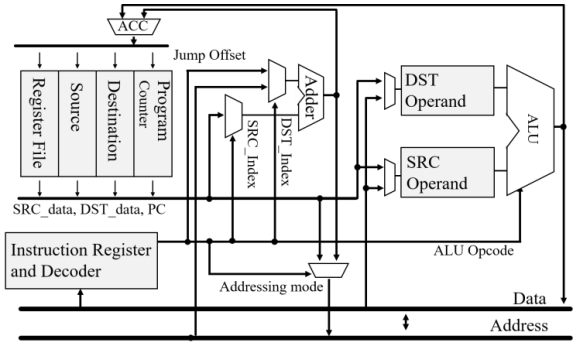


Fig. 9. Data path architecture [32].

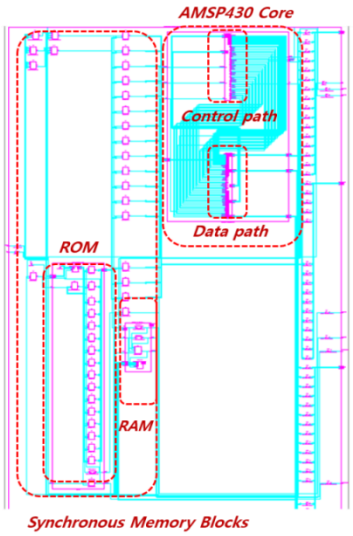


Fig. 10. Technology schematic of designed system.

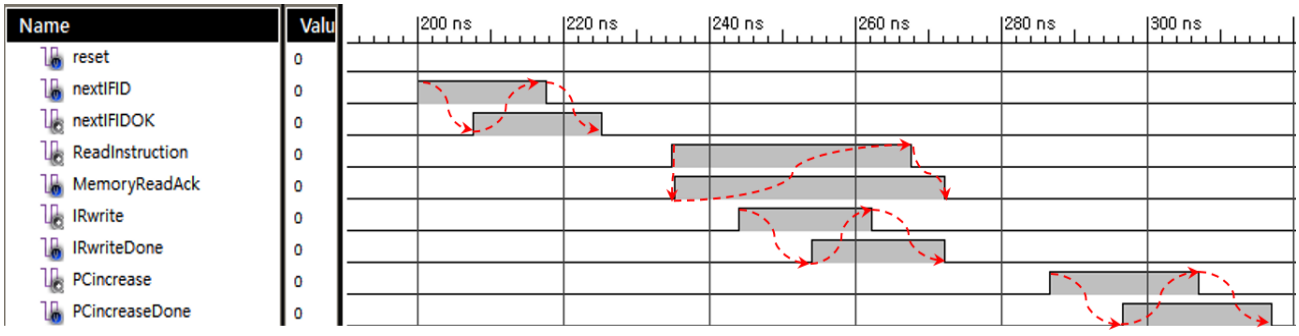
V. Experimental Result

In this paper, we designed two different versions of the MSP430 core and these are implemented on the commercialized Xilinx Spartan-3 FPGA. For the description language, we utilize Verilog HDL. As a designing and timing simulation tools, Xilinx ISE and ISim were used. And, to keep the architectural design of both cores, we did not optimize the control path and data path on the synthesis level of design [29]. Lastly, in order to verify its functionality, we perform timing simulation for each control module on each step of the design process through Xilinx ISim. Fig. 10 shows the waveforms for partial parts of instruction fetch and decode module and synchronous block ROM interfacing module simulations.

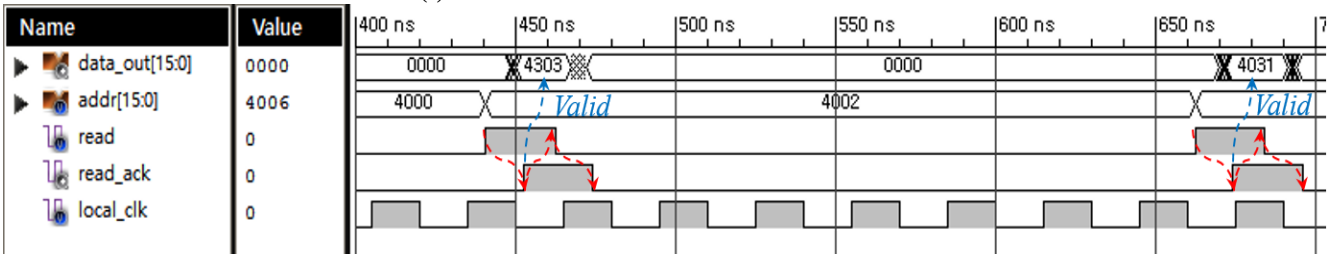
As shown in Fig. 11 (a), when next IFID (Req) signal is asserted, which indicates Req signal for fetching the next instruction, to express Ack, the next IFIDOK signal is excited. Subsequently, ReadInstruction (Req) is generated for accessing the memory interface. Consequently, as an Ack signal, MemoryReadAckis delivered. Afterward, for writing the instruction to the instruction register, IRwrite (Req) signal rises. And to deliver the Ack, the IRwirteDone is asserted. As a consequence, in order to increase program counter (PC), PCincrease (Req) and PCincreaseDone (Ack) is followed. Likewise, other control signals are delivered.

Therefore, through the timing simulation, we confirmed that the designed controller executes the 4-phase handshake protocol and its output is verified with required AFSM control signals.

The Fig. 11 (b), describes the simulation of interfacing module for the synchronous block ROM. When the read (Req) and 16-bit address are delivered to the interfacing module, for the read signal, it passes through the matched delay cells, which are configured as LUT-based one. Then according to the delay time, the Ack signal is generated. In this point, Ack signal should assert, when output data becomes valid. And for the address, inputted address is translated to memory address then, related to the Req and Ack signals, the respected data is delivered.



(a) Waveform of the instruction fetch and decode module



(b) Waveform of the interface module with synchronous block rom

Fig. 11. Post-route level timing simulation.

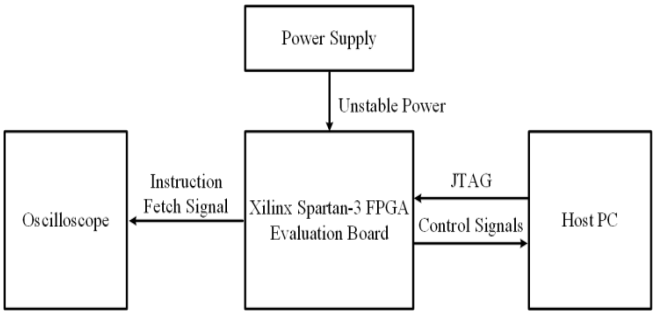


Fig. 12. Block diagram of experimentation environment.

To check its functionality in the real system, we implement the entire system on to the Spartan-3 FPGA and it runs benchmark programs [34], when every module is passed the timing simulation.

As described in Fig. 12. The host PC was used for downloading the bitstream file to target FPGA through the JTAG interface as well as timing simulation for before implementing the designed core. To check the functionality of designed core, we run the emulator [35] as the reference core on the host PC and it executes same benchmark program [34]. The host PC monitors output signals of designed core. (e.g., PC, address, control signals).

Additionally, to check the performance of the each designed core, we measure the cycle of instruction fetch signal through the oscilloscope. Additionally, to clarify its durability under the unstable voltage circumstances, we apply an unstable voltage to designed cores.

The performance in the typical condition has measured both cores, around 18MHz. According to the supplied voltage, the performance of each core is degraded at the condition of low-voltage. In detail, for the synchronous core, the instruction fetching cycle is dramatically dropped. On the other hand, the asynchronous one keeps its performance until 3.2V and it degrades at the 3.0V.

This experimentation stands for the asynchronous core can execute the application on the low-voltage condition and it can guarantee its functionality until 3.0V. However, the functionality of the synchronous core cannot be guaranteed under unstable supply voltage condition. While the instruction fetch signal from the synchronous controller (FSM) is functioning, the data path is malfunctioned. The reason is that the clock network becomes unreliable under the unstable voltage circumstance.

Withal, to check the power consumptions of each core in precisely, we perform power analysis through the Xilinx ISE and ISim. The power simulation results show that the SMSP430 consumes 84.2mW. To be specific, since SMSP430 has clock network, the dynamic power shows 28.7 %from the entire power consumption. On the other hand, in the case of AMSP430, owing to the nature of the asynchronous circuit it can remove the globalized clock network, which accounts for around 30% of the total power consumption in our case, fundamentally. Therefore, AMSP430 consumes lower power than SMSP430.

VI. Conclusion and Future Work

In this paper, we propose the design techniques for the bounded delay model with the 4-phase signaling-based asynchronous circuit on the commercial FPGAs. Then, we designed the 16-bit micro-controller, which targeted to the IoT application, through the proposed techniques.

The experimentation and simulation results show that designed asynchronous MSP430 consumes 41% lower power than the synchronous MSP430.In addition, the asynchronous core exhibits fault tolerance in the unstable power condition comparing to the synchronous one.

Henceforth, we will optimize the asynchronous MSP430 core and attach peripherals for the real applications. Also, to provide congeniality with MSP430-GCC compiler, we will analyze the compiled binary program in the near future.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.
[2] Verizon, *State of the Market: Internet of Things 2016 Report*, Verizon, 2016.
[3] J. Uthus and O. Strom, *MCU Architectures for Compute-Intensive Embedded Applications*, ATMEL White Paper, 2009.
[4] Arduino. (2017). [Online]. Available: https://www.arduino.cc/
[5] Raspberrypi. (2017). [Online]. Available: https://www.raspberrypi.org/
[6] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen. "A survey on internet of things from industrial market perspective," *IEEE Access*, vol. 2, pp. 1660-1679, 2014.
[7] Y. I. Ismail, "Interconnect design and limitations in nanoscale technologies," in *Proc. 2008 IEEE International Symposium on Circuits and Systems*, 2008, pp. 780-783.
[8] C. J. Anderson, *et al.*, "Physical Design of a Fourth-Generation POWER GHz Microprocessor," in *Proc. ISSCC2001*, 2001, pp. 232-233.
[9] J. McCardle and D. Chester, "Measuring an asynchronous processor's power and noise," in *Proc. Synopsys User Group Conf.*, 2001, pp. 66-70.
[10] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor, and G. Stegmann, "An asynchronous low-power 80C51 microcontroller," in *Proc. 4th ASYNC*, 1988, pp. 96-107.
[11] M. E. Dean, "STRiP: A self timed RISC processor," Ph.D. dissertation, Stanford University, CA, 1992.
[12] S. B. Furber, J. D. Garside, S. Temple, J. Liu, P. Day, and N. C. Paver, "AMULET2e: An asynchronous embedded controller," in *Proc. IEEE*, Aug. 2002, pp. 243-256.
[13] J. Sparso and S. Furber, *Principles of Asynchronous Circuit Design – A* Systems *Perspective*, New York, U.S,: Springer, 2001.
[14] A. Davis and S. M. Nowick, "An introduction to asynchronous circuit design," *The Encyclopedia of Computer Science and Technology*, vol. 38, pp. 1-58, Sep. 1997.
[15] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in *Proc. 3rd Caltech Conf. on VLSI*, 1983, pp. 87-116.
[16] G. De Micheli, "Synchronous logic synthesis: Algorithms for cycle-time minimization," *IEEE Tr. On CAD*, vol. 10, pp. 63-73, 1991.
[17] P. H. Ho. "Industrial Clock Synthesis," *ISPD*, 2009.
[18] E. G. Friedman, "Clock distribution network in synchronous digital integrated circuits," in *Proc. IEEE*, May 2001, pp. 665-692.
[19] A. Yakovlev, L. Lavagno, and A. Sangiovanni-Vincentelli, "A unified signal transition graph model for asynchronous control circuit synthesis," in *Proc. ICCAD92*, 1992, pp. 104-111.
[20] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavgno, and A. Yakovelv, "PETRIFY: A tool for manipulating concurrent specifications and synthesis of asynchronous controller," *IEICE Tr. on Information and System*, vol. E80-D, no. 3, pp. 315-325, Mar. 1997.
[21] K. M. Fant, *Logically Determined Design Clockless System Design with Null Convention Logic*, New York, U.S: John Wiley & Sons, Inc., 2005.
[22] C. J. Myers, *Asynchronous Circuit Design*, New York, U. S: John Wiley & Sons, 2001.
[23] K. Y. Yun, "Synthesis of asynchronous controllers for heterogeneous systems," Ph.D dissertations, Stanford University, CA, 1994.
[24] MSP430x2xx *Family User's Guide*, SLAU144J, Texas Instrument, 2013.

[25] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "An asynchronous FPGA based on dual/single-rail hybrid architecture," in *Proc. ERSA'12*, vol. 139, 2012.
[26] M. M. Kim and P. Beckett, "Design techniques for NCL-based asynchronous circuits on commercial FPGA," in *Proc. 2014 17th Euromicro Conference on Digital System Design*, 2014, pp. 451-481.
[27] *Spartan-3 Generation FPGA User Guide*, UG331 (v.1.8), Xilinx, 2011.
[28] M. H. Oh, Y. W. Kim, S. Kwak, C. H. Shin, and S. N. Kim, "Architectural design issues in a clockless 32 bit processor using an asynchronous HDL," *ETRI Journal*, vol. 35, no. 3, pp. 480-490, Jan. 2013.
[29] Constraints *Guide*, UG625 (v. 14.5), Xilinx, 2013.
[30] J. G. Lee and M. H. Oh, "Asynchronous circuit designs on an FPGA for targeting a power/energy efficient SoC," *IEICE Trans. Electron.*, vol. E97-C, pp. 253-263, Apr. 2014.
[31] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Francisco, C.A, U.S,: Morgan Kaufmann Publishers Inc., 2011.
[32] M. H. Oh, C. Shin, and S. Kim, "Design of Low-Power Asynchronous MSP430 Processor Core Using AFSM Based Controllers," in *Proc. 23rd ITC-CSCC*, 2008, pp. 1109-1112.
[33] Spartan-*3 FPGA Family Data Sheet*, DS099, Xilinx, 2013.
[34] Z. Shin, M. H. Oh, J. G. Lee, H. Y. Kim, and Y. W. Kim. (Mar. 2017). Design of a Clockless MSP430 Core Using Mixed Asynchronous Design Flow. *IEICE Electronics Express*. [Online] 14(8). pp. 20170162. Available: http://doi.org/10.1587/elex.14.20170162
[35] Rudolf Geosits. MSP430-Emulator. *GitHub*. Available: https://github.com/RudolfGeosits/MSP430-Emulator

Ziho Shin received B.E. in electrical and electronics engineering from University of Ulsan in 2015 Ulsan, Korea. He was worked as VRE operation intern goodwill San Francisco|SanMateo|Marine Counties HQ in 2014 at San Francisco, California, USA. And he joined Automatic Control LAB in University of Ulsan, Ulsan, South Korea as a research assistant from 2012 to 2013. He is currently enrolled in master course student in computer software, Department of University of Science and Technology (UST) in Electronics and Telecommunication Research Institute (ETRI) Campus, Daejeon, Korea. His research interest focuses on computer architecture, asynchronous circuit design and high-speed system fabric interconnection design.

Myeong-Hoon Oh received his Ph.D in information and communications engineering from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea in 2005. He has been with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea since 2005 as a senior and principal researcher. From 2006, he has been an associate professor in University of Science and Technology (UST), Daejoen, Korea. His current research focuses on digital circuit design, embedded system, cloud computing infrastructure, and cloud computing standardization. He also has been an editor for cloud computing in ITU-T SG13 since 2012.

Hyukje Kwon received his MS degrees in electronics engineering in Chonbuk National Univerisity, Korea in 1997. And in 2008, he received PhD degrees in the same university. In 2012, he joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. His current research interest focuses on micro-server system and high-speed system fabric interconnection design.

Hag Young Kim received B.S and M.S in electronics engineering from Kyungpook National University, Daegu, Korea, in 1983, and 1985, respectively, and Ph.D in computer engineering from Chungnam National University, Daejeon, Korea, in 2003. In 1988, he joined ETRI, Daejeon, Korea. His current research interests are micro-server and high performance computing system architecture.

Dongjae Kang received B.S and M.S in computer science from Inha University, Incheon, Korea, in 1999 and 2001 respectively and Ph.D in computer and information engineering from Inha University, Incheon, Korea in 2010. He is a principal researcher at cloud computing research group in ETRI since 2001 and is a professor at computer software department in University of Science and Technology (UST) since 2009. His main research part is Cloud computing and virtualization. Another interesting parts includes system software, open source software (OSS) and system management software. Since 2008, his research focused on multi-Cloud technologies including server virtualization, Cloud management platform, Cloud service brokerage and Cloud federation.