Design of Low-Power Asynchronous MSP430 Processor Core Using AFSM Based Controllers

Myeong-Hoon Oh[1], Chihoon Shin[2], and Seongwoon Kim[1]
[1] Electronics and Telecommunications Research Institute (ETRI)
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, Korea
[2] School of Engineering, University of Science and Technology (UST)
113 Gwahangno, Yuseong-gu, Daejeon, 305-333, Korea
E-mail: [1]mhoonoh@etri.re.kr, [2]chsin@etri.re.kr

Abstract: As an asynchronous design method, TiDE tool chain supports the robust design flow at a high level with an asynchronous specialized language, HASTE. However, there is a limitation at an optimized step, since it is difficult for designers to know and manage inner synthesis operations. With a button-up manner based on the synthesis method with asynchronous finite state machine (AFSM), we design an asynchronous MSP430 core which is widely used in the sensor nodes.

The designed asynchronous MSP430 core was compared with an already designed asynchronous MSP430 core which employed the TiDE tool flow and a clock based synchronous MSP430 core at 0.13 um CMOS technology. The maximum performance of the TiDE tool version was only 40.6 % of the synchronous or the AFSM version. The AFSM version saves the energy consumption of the synchronous version and the TiDE tool version by about 31.9 % and 28.8 %, respectively.

1. Introduction

An asynchronous circuit design is based on a data transfer mechanism among neighboring modules with handshake protocols and their synchronizations are localized in corresponding modules. Therefore, the timing closure is not a critical design issue any more in asynchronous designs. This is one of attractive advantages of such a self-timed design over synchronous designs where power consumption for clock tree balance is forming a grater part of the total amount of power dissipation [1].

Also, asynchronous circuits enable on-demand or event-driven styled operations which have good energy efficiency. Designers do not need to use a special technique, such as a clock gating, that prevents clock propagation forcibly around a whole system.

The asynchronous design itself does not mean a low-power design but there are a lot of successful stories in that area due to its natural characteristics mentioned above [2, 3, 4, 5].

Meanwhile, the market of a sensor network is rapidly growing and a low-power feature becomes an essential requirement of a sensor network research field. Microcontrollers in sensor nodes need an energy-efficient specification as well.

The objective of this paper is to design a compatible version of an MSP430 processor core [6] which is widely employed in sensor nodes by using an asynchronous design method and to verify the energy-efficiency of the design.

2. Asynchronous Design

A signaling and encoding scheme for the handshake protocol is based on the 4-phase bundled data assumption [7]. This scheme can use the same data path logics that are employed to synchronous designs and manage the timing of signals in a control path through matched delay elements to each sub-block in a data path.

With the same signaling and encoding scheme, the MSP430 processor core has been designed by the TiDE tool [5] in [8]. The TiDE is easy and reliable to use because, like conventional CAD tools, it supports a consistent design flow from a high level language-based description method which is named HASTE to synthesized gate level verilog netlists. However, all designers can do is only to describe their asynchronous circuits using HASTE and it is considerably difficult to join in the rest of the design flow. Accordingly, how much the circuit is optimized depends on only description ways of HASTE and this definitely is a design limitation.

To eliminate the limitation, we apply a button-up styled design flow, in other words, a graph-based method. Designers can describe behaviors of all control circuits in a system from the low level stage with graph model such as Petri net or asynchronous finite state machine (AFSM). The specification using such graph models can be implemented by specialized synthesis tools like Petrify [12] or 3D [9].

Among several graph- based methods, we apply the design of the control path to AFSM based method because its operation is similar to synchronous FSM and synthesized circuit is smaller than the Peri-net based method if there is not many current behaviors in the system. The most of the control operation in the MSP430 core is serialized.

3. Architecture of Asynchronous MSP430 Core

The MSP430 processor core [10] has a 16-bit instruction set and total 7 address modes. The number of instructions is not large but all instructions can have all address modes. In addition, the number of memory access in every single instruction cycle varies zero to five concerning address modes. So the pipeline based architecture which assumes that utilization of data path in all instructions is fairly distributed is not suitable for our design. We select a CISC style architecture where completion time of each instruction

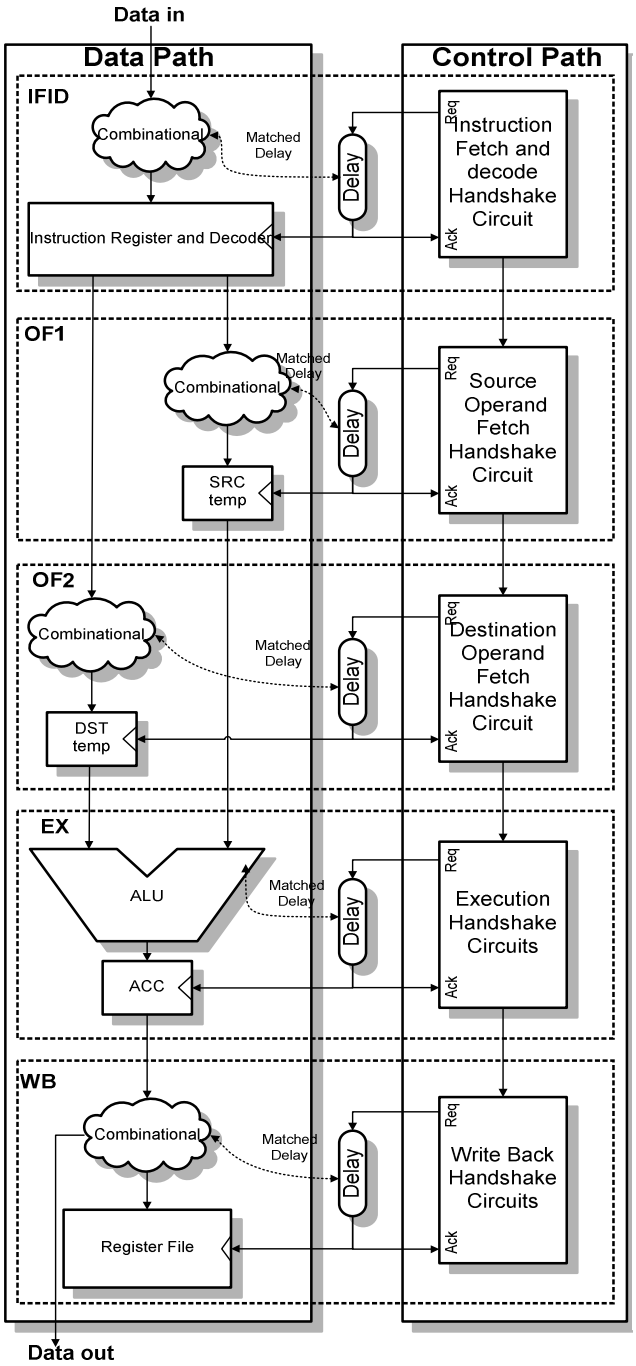

Figure 1. Block diagram of the asynchronous MSP430 core

is variable according to the types of instruction and address modes. Figure 1 shows the abstracted block diagram of the asynchronous version of the MSP430 core.

The architecture is comprised of five sub-modules and each sub-module has a pair of the data path and the control path. For convenience, sub-modules are named IFID, OF1, OF2, EX, and WB which means a instruction fetch and decode, a source operand fetch, a destination operand fetch, an execution, and a write back, respectively. All instructions do not fully pass through every module and data flow changes according to instruction types and address modes.

In the control path, there are control logics which describe the sequences of local clocks and control signals needed in

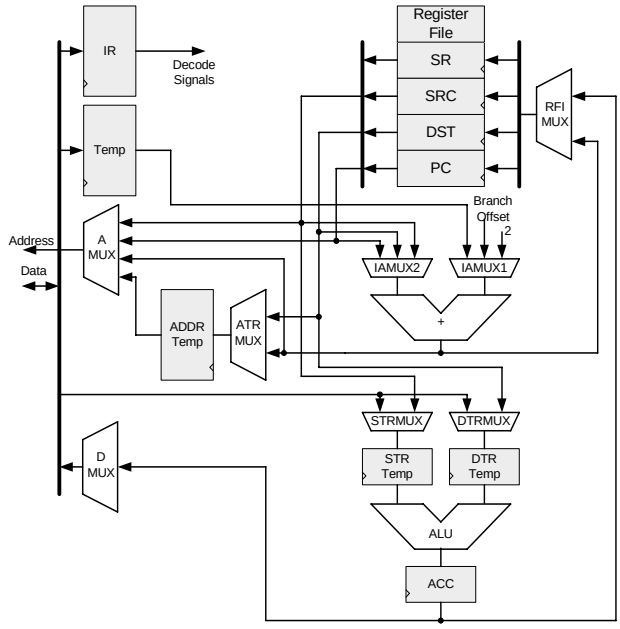

Figure 2. Datapath of MSP430 core

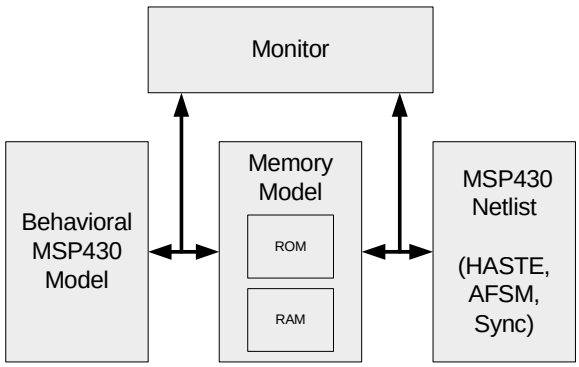

Figure 3. Simulation environment

the data path. The control logics are based on AFSM and regulate the timing of the data path by using each matched delay element. Figure 2 shows a block diagram of the asyncrhonous MSP430 core.

4. Design and Simulation Result

Each AFSM based control logic was implemented by directly mapping an equation from 3D logic synthesis tool [9] to AND-OR styled circuitry in a gate level. For comparison with other versions, we reused design outcomes of the asynchronous version with the TiDE tool and the synchronous version in [8]. Three versions were modeled with the same UMC 0.13 um libraries and their functionalities are verified by benchmark programs specialized in sensor network applications [11] in a simulation environment of Figure 3.

Table 1 summarizes the performance of three versions as a metric of a maximum equivalent clock frequency. Note that an asynchronous circuit does not have a global clock so it is reasonable to convert its performance to a clock

Table 1. Maximum performance of each version

Version	Max. equivalent clock frequency (MHz)
sync	89.2
AFSM	89.1
HASTE	36.2

(sync: synchronous version,
AFSM: AFSM version,
HASTE: TiDE tool version)

frequency of synchronous one which has the same completion time of asynchronous one.

The performance of the AFSM version is nearly similar to the synchronous version since they use the identical data path. Besides, both timing requirements of a global clock (in case of the synchronous version) and delay elements (in case of the AFSM version) meet a critical path delay.

The maximum equivalent clock frequency of the TiDE tool version is only 40.6 % of the synchronous or the AFSM version. The reason is that the reflection of designer's intention is restricted to only HASTE code generation, while a separate optimization for control paths and data paths can be performed at the synchronous or AFSM version.

Power consumption of the three versions depending on variable equivalent clock frequencies is depicted in Figure 4. As the equivalent clock frequency increases, dissipated power also continues to grow. Since we cannot access and adjust internal delay elements in the TiDE version, its power amount is fixed and measured to about 400 uW.

With the same equivalent clock frequencies, consumed energy metrics by unit of "uW/MHz," are described in Figure 5. As shown in Table 2, the AFSM version saves the energy consumption of the synchronous version and the TiDE tool version by about 31.9 % and 28.8 %, respectively.

In the simulation we assume a global clock in the synchronous version is ideal so the power dissipation of a

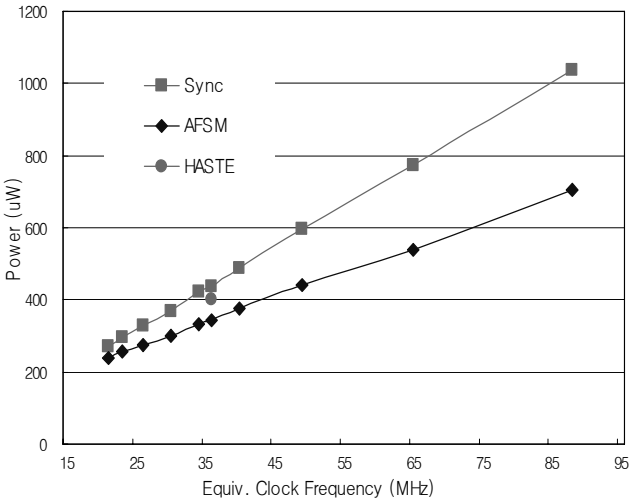

Figure 4. Power consumption of each version

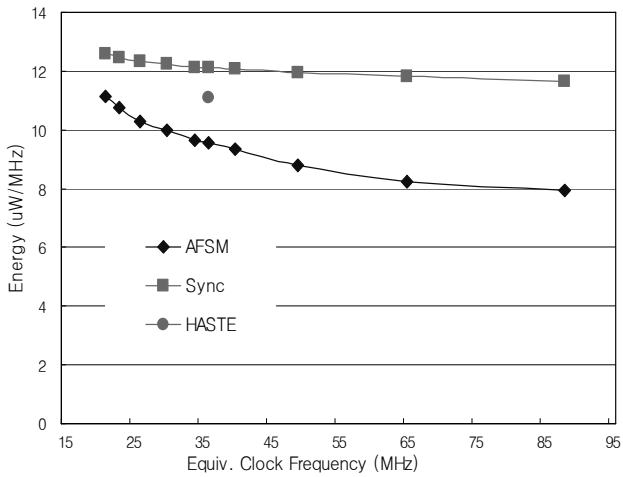

Figure 5. Energy consumption of each version

Table 2. The minimum energy consumption of each version

Version	Energy (uW/MHz)
sync	11.6
AFSM	7.9
HASTE	11.1

clock tree network is zero. Considering an increasing percentage of the power for clock distribution in P & R stages or real fabricated chips [1], calculated reduction ratio of the energy consumption between the AFSM version and the synchronous version can be much more improved.

Conclusion

An asynchronous MSP430 processor core was designed with a control path synthesis flow based on AFSM.

Comparing with another asynchronous version using TiDE tool flow and a synchronous version at 0.13 um CMOS technology, it is concluded that the designed AFSM version can save the energy consumption of each version by about 31.9 % and 28.8 %, respectively.

With the real environment with a fabricated chip where percentage of the power for clock distribution in a total power is increasing, calculated reduction ratio of the energy consumption between the AFSM version and the synchronous version can be much more improved.

References

[1] J. Pangjun and S. S. Sapatnekar, "Low-power clock distribution using multiple voltages and reduced swings," *IEEE Trans. VLSI Systems*, Vol. 10, No. 2, pp. 309-318, June 2002.

[2] J. Kessels and R. Marston, "Designing asynchronous standby circuits for a low-power pager," *Proc. of the IEEE*, Vol. 87, No. 2, pp 257-267, Feb. 1999.

[3] L. S. Nielsen and J. Sparso, "Designing asynchronous

circuits for low power: an IFIR filter bank for a digital hearing aid," *Proc. of the IEEE*, Vol. 87, No. 2, pp 268-281, Feb. 1999.

[4] H. van Gageldonk, and et. al, "An asynchronous low-power 80c51 microcontroller," *Proc. int. Symp. Advanced Research in Asynchronous Circuits and Systems*, pp. 96-107, 1998.

[5] http://www.handshakesolutions.com

[6] C. Hynch and F. O'Reilly, "Processor choice for wireless sensor networks," *Proc. REALWSN 2005*, 2005.

[7] S. B. Fuber and P. Day, "Four-phase micropipeline latch control circuits," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, Vol. 4, No. 2, pp. 247-253, June 1996.

[8] D. Shang, and et. al, "Asynchronous functional coupling for low power sensor network processors," *Lecture Note in Computer Science*, Vol. 4644/2007, pp. 53-63, Aug. 2007.

[9] K. Y. Yun, "Synthesis of Asynchronous Controllers for Heterogeneous Systems," Ph.D. thesis, Stanford University, Aug. 1994.

[10] MSP430x1xx Family User's Guide, Texas Instrument, 2006.

[11] L. Nazhandali, M. Minuth, and T. Austin, "SenseBench: toward an accurate evaluation of sensor network processor," *Proc. of the IEEE Int. Symp. Workload Characterization*, pp. 197-203, Oct. 2005.

[12] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavgno, and A. Uakovelv, "PETRIFY: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Trans. Information and Systems*, Vol. E80-D, No. 3, pp. 315-325, March 1997.