편재형 센서네트워크 노드를 위한 저전력 비동기 MSP430 프로세서

A Low Power Asynchronous MSP430 Processor for Ubiquitous Sensor Network

*신치훈, **Delong Shang, ***오명훈, ***김영우, ***김성남, **Alex Yakovlev, ***김성운
ChiHoon Shin, Delong Shang, MyeongHoon Oh, YoungWoo Kim, SungNam Kim, Alex Yakovlev, SungWoon Kim

Abstract – This paper describes the design of an asynchronous implementation of a sensor network processor. The main purpose of this work is the reduction of power consumption in sensor network node processors and the research presented here tries to explore the suitability of asynchronous circuits for this purpose. The Handshake Solutions toolkit is used to implement an asynchronous version of a sensor processor. The design is made compact, trading area and leakage power savings with dynamic power costs, targeting the typical sparse operating characteristics of sensor node processors. It is then compared with a synchronous version of the same processor. Both versions are then compared with existing commercial processors in terms of power consumption.

Key Words : Sensor Network, Low Power, Micro Processor

I. INTRODUCTION

Wireless sensor networking is one of the most exciting technologies to emerge in recent years [6]. A sensor network (SN) device called sensor node can provide access to information anytime, anywhere by collecting, processing analyzing and disseminating data [7]. Duc to difficulties in maintenance, power efficiency is a prime concern in the sensor nodes [5]. However, the majority of current SN research is biased on software and communication, usually using off-the-shelf microprocessors which have high speed and power consumption greater than most SN applications require [1].

Asynchronous circuits are a potential solution to make specialized processor saving power, as the circuits completely remove the global clock signals which takes big proportion in energy budget (i.e. about 65% of the total power in the Alpha 21064). In addition to such considerations of dynamic power, asynchronous circuits may also be suitable for reducing leakage power as compact designs, with smaller area thus lower leakage, can be more easily derived.

In this paper, we will present the design of an asynchronous sensor processor based on the common understanding of SN applications as an exploration for the suitability of asynchronous circuits in low power sensor processors. We will pay special attention to the question of trading leakage gains with dynamic power costs.

II. ASYNCHRONOUS SN PROCESSOR IMPLEMENTATION

The MSP430 family of microprocessors from Texas Instruments has widely used for sensor networking applications. The micro-controller incorporates a flexible clock system which is designed specifically for battery-powered applications and uses six different operating modes to keeping the digital oscillator running to generate the clock but disabling the loop control to save power, to fully powered down [2]. This features looks attractive in the context of SN systems. In this work, we chose MSP430 as our ISA to connect our research with current SN researches based on MSP430 h/w platform.

Recently, a benchmark method for SN processors has been proposed [1]. This takes into account that for SN processors pure performance is less important, because SN applications are usually periodic with extremely slow duty cycle, as shown in Table I.

Applications	Duty Cycle (Hz)
Atmospheric temperature	0.017 - 1
Barometric pressure	0.017 - 1
Body temperature	0.1 - 1
Natural seismic vibration	0.2 - 100
gine temperature	100 - 150

TABLE 1. Sensor Network Application Sampling Rate

The duty cycle is defined as in equation (1),

$$DutyCycle = \frac{1}{RunTime + IdleTime} \quad (1)$$

In sensor based applications, when the idle time gets longer, the power consumption during this period becomes more significant. Normally, static energy such as leakage of circuits and some dynamic energy, like clock transition, affect the power consumption at idle time. Asynchronous circuits have the potential advantage of very low power consumption because they only dissipate when and where necessary and the global clock signals are completely removed [3]. Handshake Solutions offers one of the most successful toolkits of this type. This toolkit is called TiDE. It also includes Haste, used as a general-purpose

* University of Science and Technology
** Newcastle University, UK
*** ETRI Server Platform Research Team

programming language. In this work, we use the TiDE design flow to design our asynchronous MSP430 implementation [8].

For sensor processors, because of the low sampling frequency, a compact architecture is most appropriate. A compact architecture mostly is realized by sharing hardware resources to reduce the area. The reduction called functional coupling in area usually helps reduce static power consumption such as leakage power, but the function sharing will usually cause an increase of dynamic power. When the system duty cycle is slow and processing is sparse, system idle time tends to be greater than system active time. In SN applications the duty cycle is usually dominated by idle time. In this type of situation compact architectures can be good for reducing system power consumption [9], by trading dynamic power loss for leakage power gains.

Based on the specification of the MSP430, an architecture shown in Fig. 1 is used for implementing the core of our MSP430 versions.

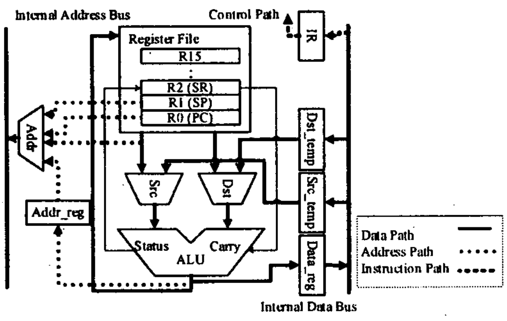

Fig.1. An datapath architecture for MSP430

There are data bus and address bus working as the interface between the core and the other part of the processor (ROM, RAM, timer, ADC, etc.). Inside the core, there are register files, instruction register (IR), address register (AREG), data register (DREG), source and destination temporary registers working as data storage.

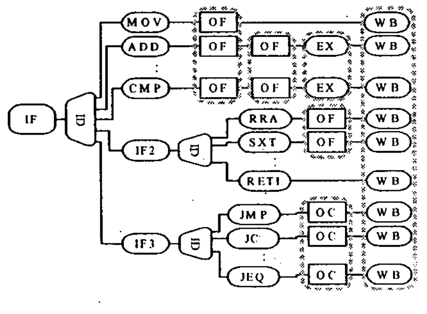

Fig.2. Operation allocation of instruction

To be compact, the other computations, such as address calculation, share the ALU unit. In addition, functional coupling is also applied to the instruction set. Fig. 2 shows the operation allocations of the instruction set. In general, after fetching an instruction and decoding, operations for a certain instruction are known and scheduled. For example, if the instruction is "add", fetching operands twice is required and then followed by 'add' (ex) and writing back. In practice, some operations can be executed based on shared components, such as operand fetching (OF), executing (EX), writing back (WB) and so on. Fig. 3 shows the block diagram of the operation allocation after functional coupling.

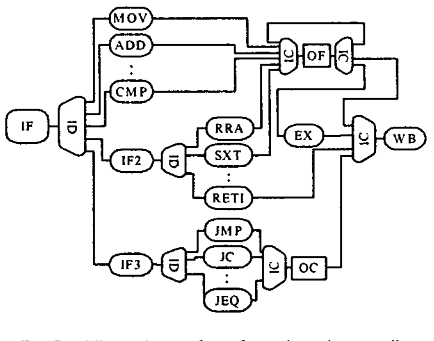

Fig.3. Allocation after functional coupling

Based on this compact architecture and the functional coupling techniques, an asynchronous MSP430 is specified in Haste, and input to the TiDE design flow. Our asynchronous MSP430 is implemented based on the TiDE design flow in UMC 130nm technology. As the design is started from the behavioral specification without converting complicated state machines, thus we can design an algorithm and synthesize it to actual circuit directly.

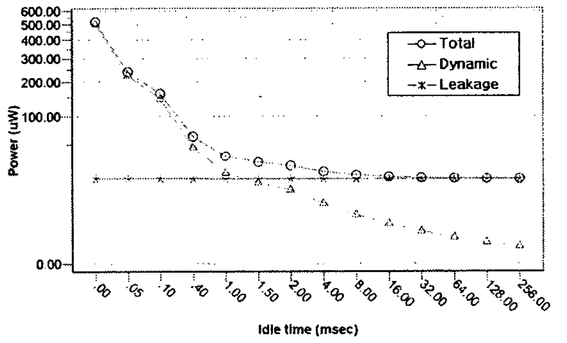

Fig. 4 Power behavior when idle time increases

As we can see in a result of power simulation shown in Fig. 4, leakage power becomes more significant than dynamic power when the idle time increases. This demonstrates the validity of our investigation of trading leakage power gains for dynamic power losses using compact designs.

IV. COMPARISON

In order to put the asynchronous design into perspective, we implemented a synchronous MSP430 on the same UMC 130nm technology, using the Synopsys toolkits. We started the design from a specification in Verilog. After compiling and mapping, a gate-level netlist and a corresponding sdf (Standard Delay Format) file are generated. After that we simulated the design based on the netlist and the sdf file. Comparisons were made between this and the asynchronous versions of MSP430, especially for power consumption characteristics.

In general, the energy per instruction is used as the power metric. However this metric is useful only for an overall and inaccurate power assumption, since almost every approach has different ISA, architecture and performance. For accurate comparison, we use the metric EPB (Energy Per Bundle) which represents actual energy during a duty cycle of work [1], from which we derived a normalized energy per instruction figure shown in Table II. Additionally, we use one practical SN application as the test bench. The application - RLE_stream - emulates environmental monitoring which periodically collects data from outside, filters them through Schmitt trigger-based threshold detector and compresses them using Run Length Encoding (RLE).

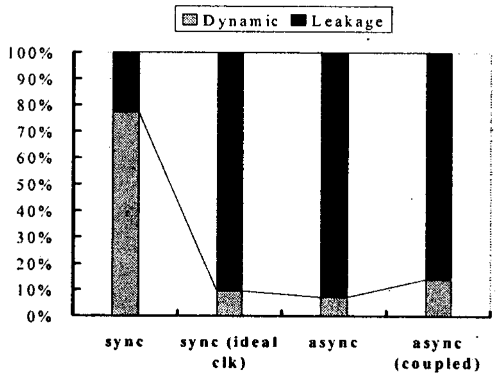

Fig. 5 Power simulation 1

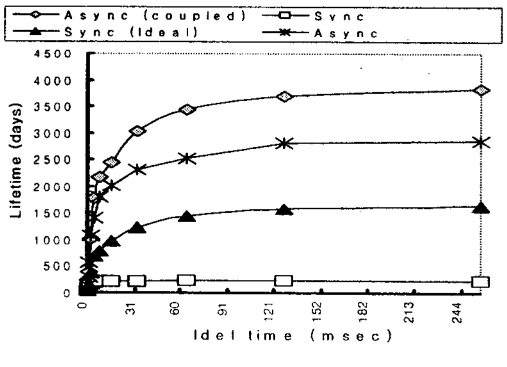

Fig. 6 Power simulation 2

The final power simulation results are shown in Fig. 5 and Fig. 6. We ran a test bench which is about 40us. In these two figures, Sync is our synchronous MSP430 which works based on a global clock signal. In implementation, a clock tree is used to balance the global clock signal. Sync idle is defined as during its idle time, the clock is gated to stop clock transition. Async is our asynchronous MSP430 without optimization, and async (functional coupled) with optimization. In Fig. 5, the idle time is 64ms, and in Fig. 6, the lifetime is for one of 1.5V 1000mA AAA size battery.

Through our coupling to derive a compact implementation, the area was reduced by almost 30%. On the other hand, dynamic power increased as much as static power decreased. Since the area is dominant power factor for SN, battery lifetime was increased by more than 30% when idle time is long enough.

V. CONCLUSIONS

A low power version of MSP430 for sensor applications was designed. Because of the specific requirements of the sensor processor, especially low duty cycles, a compact architecture was used to reduce leakage power in idle mode. The logic simulation shows that the design is correct and functions as expected. Compared to standard microprocessors, the specially designed asynchronous MSP430 for sensor applications shows good properties for low power dissipation, for example, the battery lifetime is increased by 30% by trading dynamic power for area and leakage power. Focusing on special applications, designing dedicated microprocessors is a new research topic, especially with asynchronous design technology.

ACKNOWLEDGEMENT

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-612-D00063)

REFERENCES

[1] Nazhandali, L., Minuth, M., and Austin, T.: SenseBench: Toward an Accurate Evaluation of Sensor Network Processors. ISCA 2006.
[2] TI, MSP430, http://focus.ti.com.
[3] Sparso, J., and Furber, S.,: Principles of Asynchronous Circuit Design --- A System Perspective. Kluwer Academic Publishers, 2002.
[4] Handshake Solution, Phillips: http://www.handshakesolutions.com/products_services/tools/Index.html.
[5] Lynch, Ciaran, and Reilly, F. O.: Processor Choice for Wireless Sensor Networks. In Proc.of REAlwsn 2005, June 20-21, 2005, Stockholm, Sweden.
[6] Tilak, S., Abu-Ghazaleh, N., and Heinzelman, W.: A Taxonomy of Wireless Micro-Sensor Netweok Models. ACM Mobile Computing and Communications Review (MC2R), Vol. 6, No. 2, April 2002.
[7] Tubaishat, M., and Madria, S.: Sensor Networks: An Overview. IEEE Potentials, 2003.
[8] Peeters, Ad, and Wit, Mark: Haste Manual. Handshake Solutions.
[9] Markovic, D., Stojanovic, V., Nikolic, B., and Horowitz, M. A.: Methods for true energyperformance optimization. IEEE Journal of solid-state circuits, Vol. 39, No. 8, August 2004.