International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 6: June 2015, pp 49-51. www.aetsjournal.com ISSN (Online) : 2394-6237

VLSI Design of Data Processing Architecture for Wireless Sensor Nodes

S.Kawya, S.Vignesh Kumar

***Abstract*—** Sensor network processors introduce an unprecedented level of compact and portable computing.Sensor processors have a wide variety of applications in medical monitoring, environmental sensing, industrial inspection, and military surveillance. Despite efforts to design suitable processors for these systems, there is no well-defined method to evaluate their performance and energy consumption. Most of the WSNs are energy scavenging.so it is very important to design low power consuming sensor networks.The goal of this paper is to design an ultralow- energy WSN digital signal processor by further exploiting the efficient data processing architecture. Using binary folded tree and using modified unit,the data processing is carried out. By using the processing elements effectively, the energy consumption can be decreased by 75% rather than using the normal binary tree. The design exploits the fact that many data processing algorithms for WSN applications can be described using parallel-prefix operations, introducing the much needed flexibility.

Keywords: Sensor network processors, medical monitoring, military surveillance, data processing algorithms.

I. Introduction

A wireless sensor networks distributed autonomous sensors to trackphysical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on. The WSN is built of "nodes" –from a few to several hundreds or even thousands, where each
node is connected to one (or sometimes several) sensors. Each such sensor network node has several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting.

II. CHARACTERISTICS OF WSNS AND RELATED REQUIREMENTS FOR PROCESSING

Several specific characteristics, unique to WSNs, need to be considered when designing a data processor architecture for WSNs.

S.Kawya is with Department of Applied Electronics , Mohamed Sathak Engineering College, Kilakarai. (Email : skawya92@gmail.com)
S.Vignesh Kumar is with Department of Elec. & Comm. Engg. Mohamed Sathak Engineering College, Kilakarai.(Email: svigneshap@gmail.com)

I. Data-Driven

WSN applications are all about sensing data in an environment and translating this into useful information for the end-user. So virtually all WSN application are characterized by local processing of the sensed data.

II. Many-to-Few

Since radio transmissions are very expensive in terms of energy, they must be kept to a minimum in order to extend node lifetime. Data communication must be traded for on-the-node computation to save energy, so many sensor readings can be reduced to a few useful data values.

III. Application-Specific

A “one-size-fits-all” solution does not exist since a general purpose processor is far too power hungry for the sensor node’s limited energy budget. ASICs, on the other hand, are more energy efficient but lack the flexibility to facilitate many different applications.

IV. Minimize Memory Access

Modern micro-controllers(MCU) are based on the principles of a divide-and-conquer strategy of ultra-fast processors on the one hand and arbitrary complex programs on the other hand . But due to this generic approach, algorithms are deemed to spend up to 40– 60% of the time in accessing memory, making it a bottleneck. In addition, the lack of task-specific operations leads to inefficient execution, which results in longer algorithms and significant memory book keeping.

V. Combine Data Flow and Control Flow Principles

To manage the data stream (to/from data memory) and the instruction stream (from program memory) in the core functional unit,two approaches exist. Under control flow, the data stream is a consequence of the instruction stream, while under data flow the instruction stream is a consequence of the data stream. to more or lesschoose the order of execution.

III. . PROPOSED APPROACH

In the existing system, they used binary tree for the data processing. the binary tree is folded to form a binary folded tree for the reuse of processing elements. In the digital design world, prefix operations are best known for their application in the class of carry look- ahead adders. The addition of two inputs A and B in this case consists of three stages. A bitwise propagate-generate(PG) logic stage, a group of logic stage, and a sum stage. The output of bitwise PG stage is given below.

Pi=Ai (xor) Bi ,Gi = Ai . Bi

Group PG logic stage, which implements the following expression.

(Pi,Gi)(Pi+1,Gi+1)= (Pi.Pi+1, Gi+Pi.Gi+1)

3.1 PREFIX ALGORITHM

Prefix operations can be calculated in a number of ways , but we chose the binary tree approach because its flow matches the desired on-the-node data aggregation. This can be visualized as a binary tree of processing elements (PEs) across which input data flows from the leaves to the root. This topology will form the fixed part of our approach, but in order to serve multiple applications, flexibility is also required.

To reduce the power consumption further,the PE's have to used more optimized.Through some architectural changes ,the power can be reduced further.

The tree-based data flow will, therefore, be executed on a data path of programmablePEs, which provides this flexibility together with the parallel prefix concept.

In prefix algorithm,two phases are used.

- Trunk phase
- Twig phase

3.1.1 Trunk phase

In the trunk phase the left value L is saved locally as Lsave and it is added to the right value R, which is passed on toward the root as shown in fig 3.1(a). This continues until the parallel prefix element 15 is found at the root. Note that each time, a store and calculate operation is executed.

3.1.2 Twig phase

The twig phase starts, during which data moves in the opposite direction, from the root to the leaves as shown in fig 3.1(b). Now the incoming value, beginning with the sum identity element 0 at the root, is passed to the left child, while it is also added to the previously saved Lsave and passed to the right child. In the end, the reduced prefix set is found at the leaves.

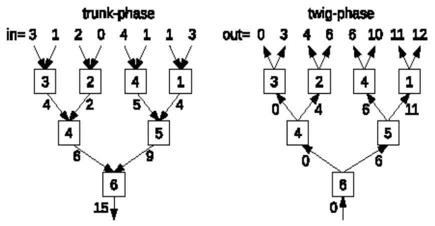


FIG 3.1 (a) Trunk phase 3.1 (b) Twig phase

3.2 FOLDED TREE

The idea presented here is to fold the tree back onto itself to maximally reuse the PEs. In doing so, p becomes proportional to n/2 and the area is cut in half. Note that also the interconnect is reduced.This newly proposed folded tree topology is depicted on the right, which is functionally equivalent to the binary tree on the left as shown in fig 3.2.

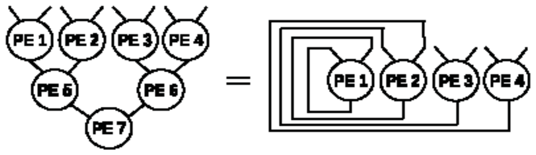
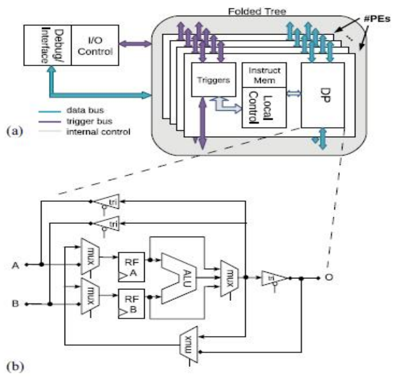


FIG 3.2 Folded binary tree

VI. BLOCK DIAGRAM



The processing elements consists of the following:

- Digital processor
- Instruction memory
- Triggers
- Local control

The DSP consists of the following components:

- RF buffer
- Arithmetic Logic Unit
- Multiplexers
- Triggers

V. PROGRAMMING USING THE FOLDED TREE

Now it will be shown how Blelloch's generic approach for an arbitrary parallel prefix operator can be programmed to run on the folded tree. As an example, the sum operator is used to implement a parallel-prefix sum operation on a 4-PE folded tree.

First, the trunk-phase is considered, a folded tree with four PEs is drawn of which PE3 and PE4 are hatched differently. The functional equivalent binary tree in the center again shows how data moves from leaves to root during the trunk-phase. It is annotated with the letters L and R to indicate the left and right input value of inputs A and B.

In accordance with Blelloch's approach, L is saved as Lsave F. Implications of using a folded tree (four4-PE folded tree shown at the top): some PEs must keep multiple Lsave's (center). Bottom: the trunk-phase program code of the prefix-sum algorithm on a 4-PE folded tree and the sum L+R is passed.

The first stage has all four PEs active. The second stage has two active PEs: PE3 and PE4. The third and last stage has only one active PE: PE4. More importantly, it can also be seen that

PE3 and PE4 have to store multiple Lsave values. PE4 must keep three: Lsave0 through Lsave2, while PE3 keeps two: Lsave0 and Lsave1. PE1 and PE2 each only keep one: Lsave0. This has implications toward the code implementation of the trunk phase on the folded tree as shown next.

The PE program for the prefix-sum trunk-phase is shown in fig 5.1(a).

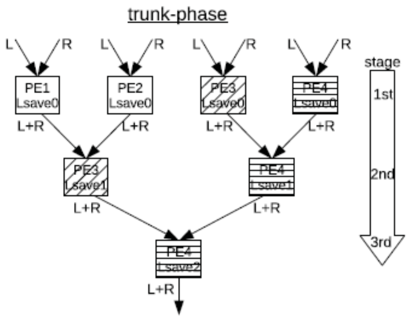


FIG 5.1(a) Trunk phase

The trunk-phase PE program here has three instructions, which are identical, apart from the different RF addresses that are used.

Now, the twig-phase is considered using Fig. 3.3(b). The tree operates in the opposite direction, so an incoming value (annotated as S) enters the PE through its O port. Following Blelloch's approach, S is passed to the left and the sum S + Lsave is passed to the right. The way the PEs are activated during the twig-phase again influences how the programming of the folded tree must happen.

To explain this, each stage of the twig-phase are shown separately to better see how each PE is activated during the twig-phase and for how many stages. First, an incoming value (in this case the identity element S2) is passed to the left. Then it is added to the previously (from the trunk-phase) stored Lsave2 value and passed to the right. PE4-instruction 1 will both pass the sum Lsave2 + S2 = S1 to the right (= itself) and pass this S1 also the left toward PE3.

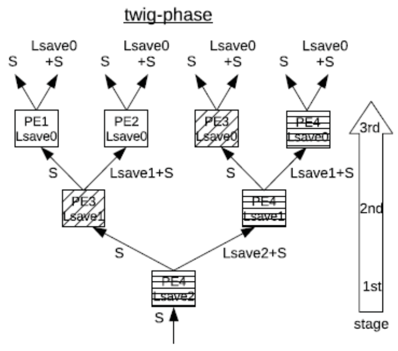


FIG 3.3(b) Twig phase

In this way the PEs are working. Thus the datas get processed in DPs and the outputs are produced.

VI. RESULTS AND CONCLUSIONS

This paper presented the folded tree architecture of a digital signal processor for WSN applications. The design exploits the fact that many data processing algorithms for WSN applications can be described using parallel-prefix operations, introducing the much needed flexibility.

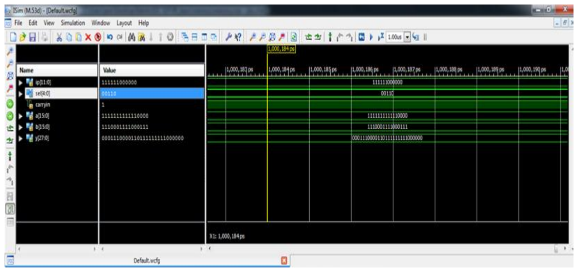
Energy is saved because of the following:

1) limiting the data set by pre-processing with parallel-prefix operations;

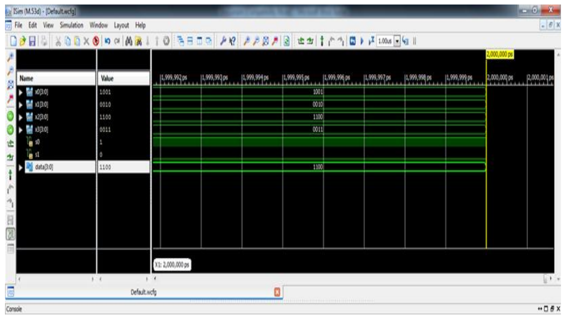
2) the reuse of the binary tree as afolded tree; and

3) the combination of data flow and control flow elements to introduce a local distributed memory, which removes the memory bottleneck while retaining sufficient flexibility.

The simulation results are shown in below.


ISim (M.53d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
1,000,184 ps
X1: 1,000,184 ps
Default.wcfg

Name	Value
ip[11:0]	111111000000
sel[4:0]	00110
carryin	1
a[15:0]	1111111111110000
b[15:0]	1110001111000111
y[27:0]	0001110000110111111111000000


ISim (M.53d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
2,000,000 ps
X1: 2,000,000 ps
Default.wcfg
Console

Name	Value
x0[3:0]	1001
x1[3:0]	0010
x2[3:0]	1100
x3[3:0]	0011
s0	1
s1	0
data[3:0]	1100

Future enhancement

The future Scope of this project is at the end of architecture router is included. It is used to reduce the delay as well as congestion. Also the novel architectures could be found for the complex data processing.

REFERENCES

[1] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless micro sensor networks," IEEE Signal Process. Mag., vol. 19, no. 2, pp. 40–50, Mar. 2002.
[2] C. Walravens and W. Dehaene, "Design of a low-energy data processing architecture for wsn nodes," in Proc. Design, Automat. Test Eur. Conf.Exhibit., Mar. 2012, pp. 570–573.
[3] H. Karl and A. Willig, Protocols and Architectures for Wireless Sensor Networks, 1st ed. New York: Wiley, 2005.
[4] J. Hennessy and D. Patterson, Computer Architecture A Quantitative Approach, 4th ed. San Mateo, CA: Morgan Kaufmann, 2007.
[5] S. Mysore, B. Agrawal, F. T. Chong, and T. Sherwood, "Exploring the processor and ISA design for wireless sensor network applications," in Proc. 21th Int. Conf. Very-Large-Scale Integr. (VLSI) Design, 2008, pp. 59–64.
[6] J. Backus, "Can programming be liberated from the von neumann style?" in Proc. ACM Turing Award Lect., 1977, pp. 1–29.
[7] L. Nazhandali, M. Minuth, and T. Austin, "SenseBench: Toward an accurate evaluation of sensor network processors," in Proc. IEEE WorkloadCharacterizat. Symp., Oct. 2005, pp. 197–203.