COMPARATIVE STUDY OF DSP ARCHITECTURES FOR WIRELESS SENSOR NODES

SAI CHAITANYA A
Assistant professor, Dept. of ECE, RGUKT
Basar, Telangana-504107,India

SAIDULU RAPOLU
Assistant professor, Dept. of ECE, RGUKT
Basar, Telangana-504107,India

Abstract
Radio conversation exhibits the very best strength consumption in wi-fi sensor nodes. Given their restricted power supply from batteries or scavenging, those nodes have to exchange information communique for on-the-node computation. however keep an appropriate processing factors and algorithms the strength and structure can be organized and Sensor networks can controlled correctly. in this paper we focused on the contrast of design of Sensor nodes by way of using BK and KS adders. An green structure is proposed for developing a sign processing block for wi-fi sensor networks.

I. Introduction

wi-fi sensor network (WSN) packages variety from medical tracking to environmental sensing, commercial inspection, and navy surveillance. WSN nodes essentially include sensors, a radio, and a microcontroller blended with a constrained energy deliver, e.g., battery or strength tocomputation energy fee can range from one hundred to 3000 [1]. So data communication ought to be traded for on-the-node processing which in turn can convert the many sensor readings into a few beneficial facts values. The information-pushed nature of WSN packages requires a particular data processing approach. previously, we have proven how parallel prefix computations may be a not unusual denominator of many WSN facts processing algorithms [2]. The aim of this paper is to design an ultralow- power WSN digital sign processor by similarly exploiting this and different characteristics particular to WSNs. Scavenging. due to the fact that radio transmissions are very costly in phrases of power, they must be stored to a minimal as a way to amplify node lifetime.

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.

Data-Driven: WSN applications are all about sensing facts in an environment and translating this into beneficial statistics for the give up-consumer. So simply all WSN applications are characterised by means of local processing of the sensed information [3]

Many-to-Few: because radio transmissions are very luxurious in phrases of power, they must be kept to a minimum if you want to extend node lifetime. facts communication ought to be traded for on-the-node computation to shop strength, so many sensor readings may be decreased to three useful facts values.

software-precise: A "one-size-suits-all" solution does not exist considering that a general purpose processor is a long way too

power hungry for the sensor node's constrained power budget. ASICs, on the other hand, are extra power green however lack the power to facilitate many exceptional packages. apart from the above characteristics of WSNs, two key requirements for improving existing processing and manage architectures may be diagnosed.

reduce memory get admission to: modern micro-controllers (MCU) are based totally at the ideas of a divide-and-overcome method of ultra-speedy processors on the only hand and arbitrary complex applications then again [4]. but due to this customary approach, algorithms are deemed to spend up to 40–60% of the time in gaining access to memory [5], making it a bottleneck [6]. similarly, the lack of venture-particular operations ends in inefficient execution, which ends up in longer algorithms and big memory ebook maintaining.

integrate statistics drift and manipulate go with the flow concepts: To manipulate the statistics flow (to/from statistics reminiscence) and the education circulation (from software reminiscence) within the core useful unit, two approaches exist. under control go with the flow, the facts move is a result of the guidance circulate, while beneath information waft the preparation stream is a result of the facts stream. A conventional processor structure is a control glide gadget, with programs that execute sequentially as a stream of commands. In contrast, a records float application identifies the statistics dependencies, which permit the processor to extra or less select the order of execution. The latter technique has been extremely successful in specialized excessive-throughput applications, consisting of multimedia and graphics processing. This paper shows how a aggregate of both strategies can cause a huge improvement over conventional WSN statistics processing solutions.

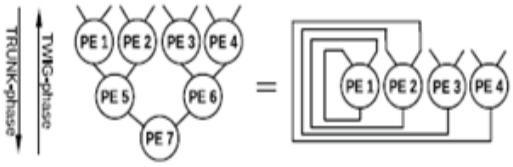


Fig1: A binary tree (left, 7 PEs) is functionally equivalent to the no foolded tree topology (right, 4 PEs) used in this architecture.

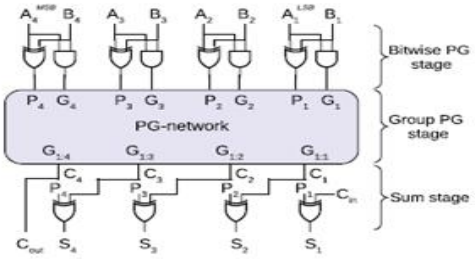


Fig 2: Addition With Propagate-Generate (PG) logic.

Existing Approach

WSN programs and On-The-Node facts Aggregation however the seemingly sizable nature of WSN programs, a hard and fast of simple constructing blocks for on-the-node processing can be diagnosed. commonplace on-the-node operations carried out on enter data accumulated without delay from the node's sensors or through in-the-network aggregation encompass filtering, fitting, sorting, and searching [7]. We published earlier [2] that these forms of algorithms may be expressed in phrases of parallel prefix operations as a not unusual denominator. Prefix operations may be calculated in some of approaches [8], however we selected the binary tree technique [9] because its drift suits the preferred on-the-node facts aggregation.

this can be visualized as a binary tree of processing factors (PEs) throughout which enter information flows from the leaves to the foundation (Fig. 1, left). This topology will form the constant a part of our method, but with a purpose to serve a couple of applications, flexibility is likewise required. The tree-based statistics go with the flow will, consequently, be executed on a statistics route of programmable PEs, which offers this adaptability together with the parallel prefix idea.

Parallel Prefix Operations in the virtual design global, prefix operations are fine known for their utility within the elegance of convey look-beforehand adders

[10]. The addition of two inputs *A* and *B* in this case consists of three stages (Fig. 2): a bitwise propagate-generate (PG) logic stage, a group PG logic stage, and a sum-stage. The outputs of the bitwise PG stage ($Pi = Ai \oplus Bi$ and $Gi = Ai \cdot Bi$) are fed as (Pi , Gi)-pairs to the group PG logic stage, which implements the following expression: $(Pi , Gi) \circ (Pi{+}1, Gi{+}1) = (Pi \cdot Pi{+}1, Gi + Pi \cdot Gi{+}1)$. It can be shown this-operator has an identify element $I = (1, 0)$ and is associative.

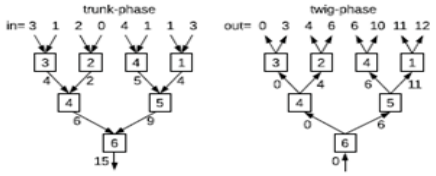


Fig 3: Example of prefix calculation with sum –operator using blelloch's generic approach in a trunk and twing phase

An instance application of the parallel-prefix operation with the sum operator (prefix-sum) is filtering an array so that all factors that do not meet sure criteria are filtered out. that is accomplished through first deriving a "maintain"-array, protecting "1" if an detail matches the standards and "0" if it ought to be disregarded. Calculating the prefix-sum of this array will go back the quantity in addition to the placement of the to-be-kept factors of the enter array. The end result array surely takes an element from the enter array if the corresponding maintain-array detail is "1" and copies it to the placement determined in the corresponding element of the prefix-sum-array. To further illustrate this, think the criterion is to simplest keep peculiar elements inside the array and throw away all even factors. This criterion can be formulated as keep$(x) = (x \bmod 2)$. The rest is calculated as follows:

input = [2, 3, 8, 7, 6, 2, 1, 5]
keep = [0, 1, 0, 1, 0, 0, 1, 1]
prefix = [0, 1, 1, 2, 2, 2, 3, 4]
result = [3, 7, 1, 5].

The keep-array provides the result of the criterion. Then the parallel-prefix with sum-operator is calculated, which results in the prefix-array. Its last element indicates how many elements are to be kept (i.e., 4). Whenever the keep-array holds a "1," the corresponding input-element is copied in the result-array at the index given by the corresponding prefix-element (i.e., 3 to position 1, 7 to position 2, etc.). This is a very generic approach that can be used in combination with more complex criteria as well.

C. Folded Tree

However, a straightforward binary tree implementation of Blelloch's approach as shown in Fig. 3 costs a significant amount of area as n inputs require $p = n - 1$ PEs. To reduce area and power, pipelining can be traded for throughput [8]. With a classic binary tree, as soon as a layer of PEs finishes processing, the results are passed on

and new calculations can already recommence independently. 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. On the other hand, throughput decreases by a factor of log2*(n)* but since the sample rate of different physical phenomena relevant for WSNs does not exceed 100 kHz [12], this leaves enough room for this tradeoff to be made. This newly proposed folded tree topology is depicted in Fig. 1 on the right, which is functionally equivalent to the binary tree on the left.

II. PROGRAMMING AND USING THE FOLDED TREE

Now it is going to be proven how Blelloch's generic method for an arbitrary parallel prefix operator can be programmed to run at the folded tree. for instance, the sum-operator is used to implement a parallel-prefix sum operation on a 4-PE folded tree. First, the trunk-phase is considered. on the top of Fig. four, a folded tree with four PEs is drawn of which PE3 and PE4 are hatched in a different way. The purposeful equal binary tree in the center again indicates how information moves from leaves to root throughout the trunk-phase. it's miles annotated with the letters L and R to indicate the left and right input price of inputs A and B. according with Blelloch's technique, L is saved as Lsave Fig. 4. Implications of using a folded tree (four4-PE folded tree shown at the pinnacle): a few PEs should keep multiple Lsave's (center). bottom: the trunk-phase software code of the prefix-sum set of rules on a 4-PE folded tree. and the sum L+R is surpassed. note that those annotations are not international, that means that annotations with the equal call do now not always proportion the identical real value. to look exactly how the folded tree functionally will become a binary tree, all nodes of the binary tree (center of Fig. four) are assigned numbers that correspond to the PE (1 via four), for you to act like that node at that level. As can be visible, PE1 and PE2 are handiest used as soon as, PE3 is used two times and PE4 is used 3 times. This corresponds to a decreasing number of lively PEs even as progressing from degree to degree. the primary level has all 4 PEs lively. the second one level has lively PEs: PE3 and PE4. The 0.33 and closing degree has simplest one lively PE: PE4. greater importantly, it can additionally be seen that PE3 and PE4 ought to shop more than one Lsave values. PE4 ought to maintain 3: Lsave0 thru Lsave2, at the same time as PE3 continues two: Lsave0 and Lsave1. PE1 and PE2 each most effective maintain one: Lsave0. This has implications closer to the code implementation of the trunkphase on the folded tree as shown next. The PE program for the prefix-sum trunk-segment is given at the lowest of Fig. 4. the outline column shows how information is stored or movements, while the actual operation is given in the last column. The write/read register files (RF) columns show how incoming data is saved/retrieved in local RF, e.g., *X*@0*bY* means *X* is saved at address 0*bY* , while 0*bY*@*X* loads the value at 0*bY* into *X*..

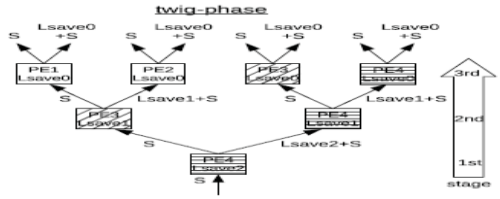
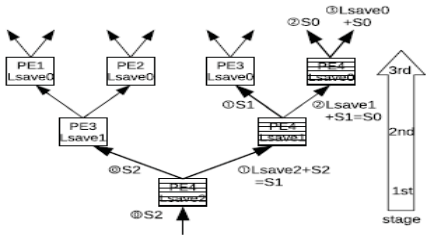


Fig 4: folded tree

that is why PE4 (active for 3 stages) needs three instructions (lines zero - 2), PE3 (lively for 2 tiers) desires commands (traces 0 - 1) and PE1 and PE2 (lively in first degree most effective) want one preparation (line 0). This essentially manner that the folding of the tree is traded for the unrolling of this system code. Now, the twig-section is considered using Fig. 5. The tree operates inside the opposite course, so an incoming value (annotated as S) enters the PE via its O port [see Fig. 4(top)]. Following Blelloch's approach, S is exceeded to the left and the sum S + Lsave is passed to the right. note that here as well none of those annotations are global. The manner the PEs are activated all through the twig-phase once more affects how the programming of the folded tree should occur. To explain this, Fig. 6 suggests every degree of the twig-section (as shown in Fig. 5) one at a time to better see how every PE is activated in the course of the twig-phase and for what number of stages. The annotations on the graph wires (turned around numbers) relate to the coaching lines of the program code proven in Fig. 7, which will additionally be discussed. Fig. 6 (pinnacle) suggests that PE4 is active all through all 3 levels of the twig-segment. First, an incoming cost (in this case the identity detail S2) is passed to the left. Then it's miles added to the formerly (from the trunk-segment) saved Lsave2 cost and surpassed to the proper. PE4-coaching 1 will each bypass the sum Lsave2 + S2 = S1 to the proper (= itself) and bypass this S1 additionally the left in the direction of PE3. The same applies for the following preparation

2 . The last coaching three passes the sum Lsave0+S0. looking at the PE3 interest [Fig. 6 (center)], it's miles handiest energetic in the 2d and third degree of the twig-phase. it is certainly most effective brought on after the primary degree when PE4 passes S2 to the left. the primary PE3-instruction zero passes S2 to PE1, and instruction 1 adds this to the saved Lsave1, passing this sum T1 to PE2. The identical manner is repeated for the incoming S1 from PE4 to PE3, that is surpassed to its left (coaching 2), even as the sum Lsave0+S1 is passed to its proper (instruction three). In fact, pairs of instructions can be identified, that showcase the same conduct in terms of its outputs: the preparation-pair 0 and 1 and the preparation-pair 2 and three . matters are special however. First, the used sign in addresses (e.g., to keep Lsave values) are one-of-a-kind. second, the first pair shops incoming values S0 and S1 from PE4, even as the second pair does now not shop anything. these variations due to the folding, once more result in unrolled application code for PE3. last, PE1 and PE2 interest are shown at the lowest of Fig. 6. They each execute two commands. First, the incoming value is surpassed to the left, observed by using passing the sum of this value with Lsave0 to the proper. this system code for each is proven within the backside two tables of Fig. 7.



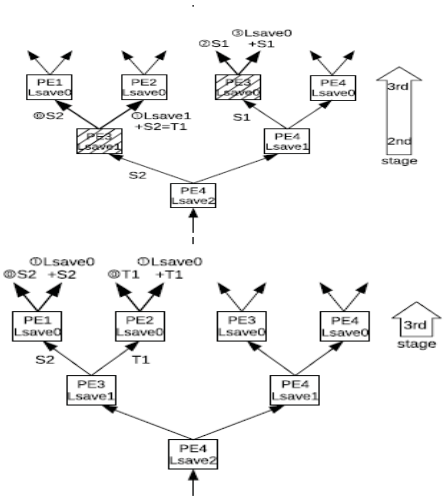


Fig. 5. Activity of different PEs during the stages of the twig-phase for a 4-PE folded tree: PE4 (top), PE3 (center), PE1 and PE2 (bottom).

III. PROPOSED WORK

on this paper the wi-fi sensor community is constructed and designed the use of parallel prefix adders mainly KS and BK adders. A assessment is formulated for these distinct architectures that are particular with their performance and low energy talents. the usage of these architecture the folding and unfolding are applied at the trans-receiver aspect. individually each method act as encryption and decryption

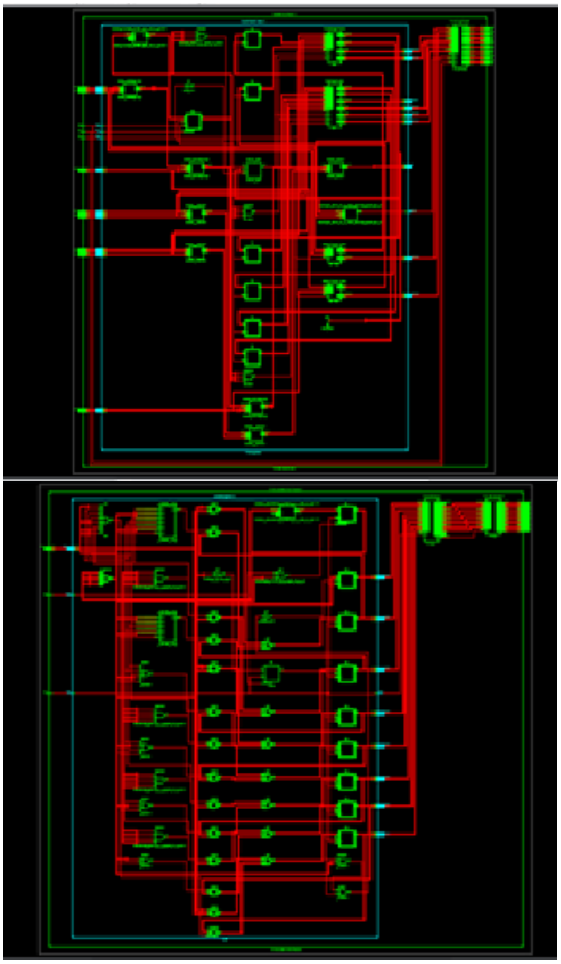


Fig 6: RTL schematic for fold & Unfold of WSN using BK

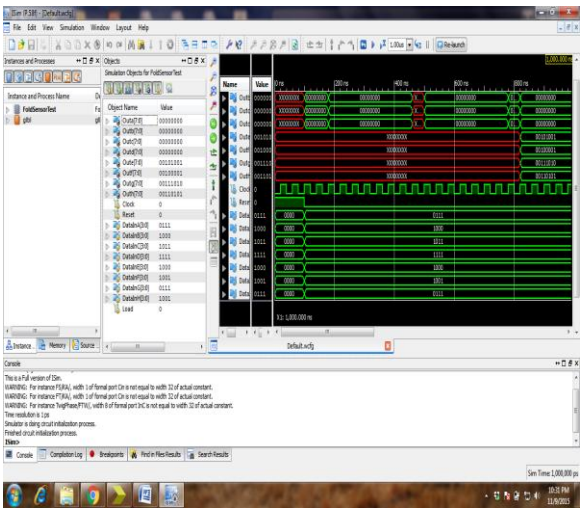


Fig 7: Simulation result of Folding of WSN using BK adder

Comparison table for BK and KS

Criteria	BK	KS
No LUT for Folding	196	176
No. LUT for Unfolding	152	147
Power	73mW	73mW

CONCLUSION

A DSP structure is designed and carried out the usage of parallel prefix adders with the aid of BK and KS adders. the use of this a folding and unfolding technique is carried out for the facts among the nodes of the sensors networks. A contrast table is drawn between the BK and KS adders wherein variety of LUT and strength are as compared and via deducing KS based totally architecture is taken into consideration be region efficient and BK is excessive performance base for Sensor community systems. The architecture is taken into consideration as according to the applications of the Sensor networking device.

REFERENCES

[1] *V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energyaware wireless microsensor 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, "Sense Bench: Toward an accurate evaluation of sensor network processors," in Proc. IEEE Workload Characterizat. Symp., Oct. 2005, pp. 197–203.*

[8] *P. Sanders and J. Träff, "Parallel prefix (scan) algorithms for MPI," in Proc. Recent Adv. Parallel Virtual Mach. Message Pass. Interf., 2006, pp. 49–57.*