

A LOW POWER LOCALIZATION ARCHITECTURE AND SYSTEM FOR WIRELESS SENSOR NETWORKS

Tufan C. Karalar¹, Shunzo Yamashita², Michael Sheets¹, Jan Rabaey¹

¹University of California Berkeley, ²Hitachi Limited

{tufan, shunzo, msheets, jan}@eecs.berkeley.edu

ABSTRACT

Localization (a.k.a. locationing) is a central concern for ubiquitous self-configuring sensor networks. In this paper the implementation of a distributed, least-squares-based localization algorithm is presented. Low power and energy dissipation are key requirements for sensor networks. As part of the sensor network, the localization system must also conform to these requirements. An ultra-low-power and dedicated hardware implementation of the localization system is therefore presented. The cost of fixed-point implementation is also investigated. The design is implemented in a 0.13 μ CMOS process. It dissipates 1.7mW of active power and 0.122nJ/op of active energy with a silicon area of 0.55mm². The mean calculated location error due to fixed-point implementation is shown to be 6%.

1 INTRODUCTION

Ubiquitous, self-configuration sensor networks hold the potential of many new applications in monitoring and control. For example, climate control, intrusion detection, visitor guidance, and target tracking can be named as such. To enable these applications, low power and energy dissipation are essential. This is particularly true, for the localization system, which is a key (but complex) building block of sensor networks. This system ensures each sensor node to acquire its own position. Thereafter the position information can be associated with the sensor data as well as used for routing. Earlier results in the sensor network localization problem focuses on the algorithmic development [1]. However, to meet the low power and energy dissipation requirement, its implementation is equally important.

This paper presents an integrated, ultra-low power implementation of a localization system. To the authors' knowledge it is the first dedicated hardware solution for such a system. It is a part of a digital sensor node integrated circuit (IC) [2]. Taking up 0.55mm² area in a 0.13 μ CMOS process, it dissipates 1.7mW of active power and 0.122nJ per arithmetic operation of active energy. The fixed-point implementation yields 6% of position estimate degradation as compared to the floating-point (FP) calculations.

The rest of the paper is organized as follows. Section 2 introduces sensor networks and the localization problem, and highlights the algorithms considered. Section 3 details the design of the localization system. Section 4 investigates the cost of fixed-point implementation. Section 5 gives the physical implementation results. Finally, this paper is summarized in Section 5.

2 SENSOR NETWORKS AND LOCALIZATION

In a self-configuring sensor network, most sensor nodes are deployed without any presumed position. Only a few numbers of nodes are given a priori information about their positions with respect to a global coordinate system. These nodes are called anchor nodes or beacon nodes. The rest of the nodes then calculate their positions by using positions of the anchors and their distances to these anchors. This operation is termed as triangulation. Triangulation is the preferred method of localization in this work. However it should be noted that non-triangulation based localization techniques have also been proposed in literature [3], [4]. Localization may be repeated regularly to track modifications in the network. These changes are expected to occur in the order of minutes. Hence localization can be repeated every 1 to 10 minutes

Equally important is that the network is distributed and there is no other special or more capable node. Hence, the triangulation operation cannot be performed at a central location. As a result, each node needs to be able to carry out its localization tasks. These consist of two main functions: (1) computing the position by the least-squares (LS) solving algorithms and (2) computing the approximate "distances" to anchors using the Hop-TERRAIN algorithm.

2.1 Triangulation and Least-squares Solving Algorithms

Triangulation is a technique used to determine the unknown coordinates of a point using n reference points with known coordinates. To perform localization in 3D at least 4 reference points are needed. Also the distances between the unknown point and reference points are required [5].

Suppose (u_x, u_y, u_z) is the unknown coordinates and (x_i, y_i, z_i) is the coordinates of the i^{th} reference point for $i = 1, \dots, n$. Then, they are related by

$$\begin{bmatrix} (x_1 - u_x)^2 + (y_1 - u_y)^2 + (z_1 - u_z)^2 \\ \vdots \\ (x_n - u_x)^2 + (y_n - u_y)^2 + (z_n - u_z)^2 \end{bmatrix} = \begin{bmatrix} r_1^2 \\ \vdots \\ r_n^2 \end{bmatrix} \quad E1$$

where r_i is the distance between the i^{th} reference point and the point whose position is unknown. Subtracting the first equation from the rest, a linear system of $n-1$ equations is obtained. This can be written as

$$\mathbf{A}\mathbf{u} = \mathbf{b} \quad E2$$

where

$$\mathbf{A} = \begin{bmatrix} (x_1 - x_2) & (y_1 - y_2) & (z_1 - z_2) \\ \vdots & \vdots & \vdots \\ (x_1 - x_n) & (y_1 - y_n) & (z_1 - z_n) \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

$$\mathbf{b} = 0.5 * \begin{bmatrix} r_2^2 - r_1^2 - x_2^2 + x_1^2 - y_2^2 + y_1^2 - z_2^2 + z_1^2 \\ \vdots \\ r_n^2 - r_1^2 - x_n^2 + x_1^2 - y_n^2 + y_1^2 - z_n^2 + z_1^2 \end{bmatrix}$$

This system of equations is over-determined if $n > 4$. Hence, the unknown position \mathbf{u} can be determined as the least squares solution to this system

$$\hat{\mathbf{u}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad E3$$

Methods to compute the least-squares solution of an over-determined linear system have been extensively studied and well published [6],[7]. In general, QR decomposition is applied to obtain an upper triangular matrix where back substitution with this triangular matrix yields the LS solution. For the QR decomposition, there are two standard algorithms: Householder reflections and Givens rotations.

$$\begin{array}{c} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ \vdots & \vdots & \vdots \\ x & x & x \end{bmatrix} \mathbf{u} = \begin{bmatrix} y \\ y \\ y \\ \vdots \\ y \end{bmatrix} \xrightarrow{i \rightarrow i} \begin{bmatrix} x & x & x \\ x & x & x \\ \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} \\ \vdots & \vdots & \vdots \\ 0 & \tilde{x} & \tilde{x} \end{bmatrix} \mathbf{u} = \begin{bmatrix} y \\ y \\ \tilde{y} \\ \tilde{y} \\ \vdots \\ \tilde{y} \end{bmatrix} \xrightarrow{i \rightarrow i+1} \\ \\ \begin{bmatrix} x & x & x \\ \hat{x} & \hat{x} & \hat{x} \\ 0 & \hat{x} & \hat{x} \\ 0 & \tilde{x} & \tilde{x} \\ \vdots & \vdots & \vdots \\ 0 & \tilde{x} & \tilde{x} \end{bmatrix} \mathbf{u} = \begin{bmatrix} y \\ \hat{y} \\ \hat{y} \\ \tilde{y} \\ \vdots \\ \tilde{y} \end{bmatrix} \xrightarrow{i+1 \rightarrow} \begin{bmatrix} \bar{x} & \bar{x} & \bar{x} \\ 0 & \bar{x} & \bar{x} \\ 0 & 0 & \bar{x} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \mathbf{u} = \begin{bmatrix} \bar{y} \\ \bar{y} \\ \bar{y} \\ \bar{y} \\ \vdots \\ \bar{y} \end{bmatrix} \end{array}$$

Figure 1 – Progress of QR decomposition using Givens rotations. The x and \tilde{x} represent non zero entries.

Algorithm using Householder reflections is an iterative approach. It zeros out the lower part of one column of the matrix at each iteration. For an m -by- n matrix, its cost is $2n^2m - 2/3n^2$ floating point (FP) operations (flops) [6].

However, this operation count does not take into account the hardware complexity of implementing each flop. During each iteration, Householder reflections require computing the squared norm of an m -dimensional vector and a division by this squared norm. These operations are usually deemed expensive for hardware implementation.

Algorithm using Givens rotations is also an iterative method. It zeros out one element of the matrix at a time instead of one column at a time (see Figure 1). It has twice the number of operations as the Householder reflections, that is $4n^2m - 4/3n^2$ flops[6]. Furthermore, floating-point operations are needed to compute the angle of a complex number and rotations with this angle. However, a CORDIC unit [8] suffices to implement these rotations in fixed-point and significantly simplifies the hardware implementation. As will be discussed more in Section 3.1, the Givens rotation based method is chosen for the QR decomposition.

2.2 Hop-TERRAIN Localization Algorithm

In this locating system, the number of hops to reach from an anchor to a node (or hop count for short) is used instead of the real Euclidean distance between the two nodes. Hence, the described above triangulation uses hop counts instead. This algorithm, which is known as Hop-TERRAIN, has initially been proposed by Savarese et.al. [10]. Independently Niculescu et.al also suggested a similar algorithm[11].

To determine the hop counts at each node, the anchors periodically initiate broadcast messages that include the position of the anchor and a hop count equal to 0. Their immediate neighbors receiving this broadcast relay it to their neighbors with the hop count incremented by 1. Hence, the messages initiated by the anchors propagate throughout the network with increasing hop counts. This process is also called flooding of the network. Anchors periodically start new rounds of flooding to capture any additions and track possible position changes. So far, the discussion on the Hop-TERRAIN only highlights the issues of the algorithm implemented in this work. A detailed discussion and analysis of this algorithm along with its simulated performance can be found in [9], [10].

In addition, flooding messages communicate throughout the network using an additional type of packet called locationing packet. Upon their detection at the Data Link Layer these packets are passed on to the localization subsystem for decoding and any subsequent action. Also, the received flooding messages are relayed by incrementing the number of hops, creating a new packet and passing that to the Data Link Layer for transmission to immediate neighbors.

3 SYSTEM DESIGN

A simplified block diagram of the implemented localization system is given below in Figure 2. The system has a Least squares equation solver, which has its own modules, an anchor information storage list or anchor list for short, as well as receive (RX) and transmit (TX) sub blocks. In the rest of this section these sub blocks are described in detail. The arrows in the Figure 2 illustrate data flow between these blocks.

3.1 Least-squares (LS) Solver

The Least squares solver is the most computationally intensive block in the localization subsystem. Therefore to optimize system performance its design and optimization received considerable effort. Besides achieving low power, there are computation rates that the Least Squares sub block must be able to support.

The computation rate is predominantly determined by localization packet receptions. This is depends on the rate that anchors start floods. The flooding rate is programmable and can be as low as one flooding per tens of minutes. Assuming a conservative “one flooding round per minute”, the rate of updates that needs to be supported would be at most 16 LS calculations in one minute. This corresponds to a time allowance of 4 seconds for each calculation.

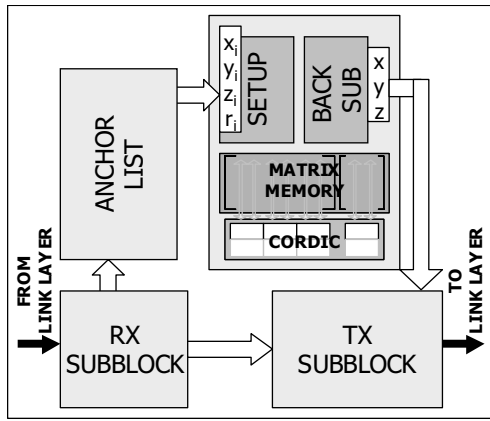


Figure 2 – Simplified localization system block diagram.

As will be seen later, this rate is low enough that all the computation is performed serially in time. In effect relaxed timing is traded for hardware complexity. One set of CORDIC blocks is used to decompose the matrix one element at a time and still meet the timing specification.

Figure 3 shows the block diagram of the LS solver block. The matrix to be decomposed is calculated using the data in the anchor list. That is, the matrix \mathbf{A} and vector \mathbf{b} of $E2$ are computed using the information anchor information from the anchor list. They are then stored on a matrix memory. Once the matrix to be decomposed is computed in the setup block the application of Givens Rotations commences.

The CORDIC rotators are used first to determine the angle of rotation and then to apply the Givens rotations to the matrix. The CORDIC block performs each rotation in 10 iterations. It is also implemented time sequentially and therefore, each CORDIC rotation takes 10 cycles. After 20 cycles, nullification of one element is completed. At each iteration, the Givens rotation that nullify entries of the matrix \mathbf{A} in $E2$ are also applied to the vector \mathbf{b} on the right hand side of the equality as shown below

$$\mathbf{Q}_i^H \mathbf{A} \mathbf{u} = \mathbf{Q}_i^H \mathbf{b} \quad E4$$

where \mathbf{Q}_i^H is the Givens matrix that nullify the entry of the matrix \mathbf{A} at the i^{th} step. Hence the equality is preserved at all

times.

In the worst case that all 16 anchors are used in the computation, then there are 39 (14+13+12) entries that need to be nullified. Hence, 780 cycles are needed to complete the QR decomposition.

Once an upper triangular matrix \mathbf{R} is obtained, the resulting 3-by-3 system is solved using back substitution. Back substitution uses a fixed-point serial divider [14]. Quotients are 10 bit wide. A divide operation takes 10 cycles and three divides take a total of 30 cycles. Another 15 cycles are also spent at the set up stage as the matrix memory is loaded. Therefore, in the longest case, approximately 825 (780 + 45) cycles are used during the LS computation step.

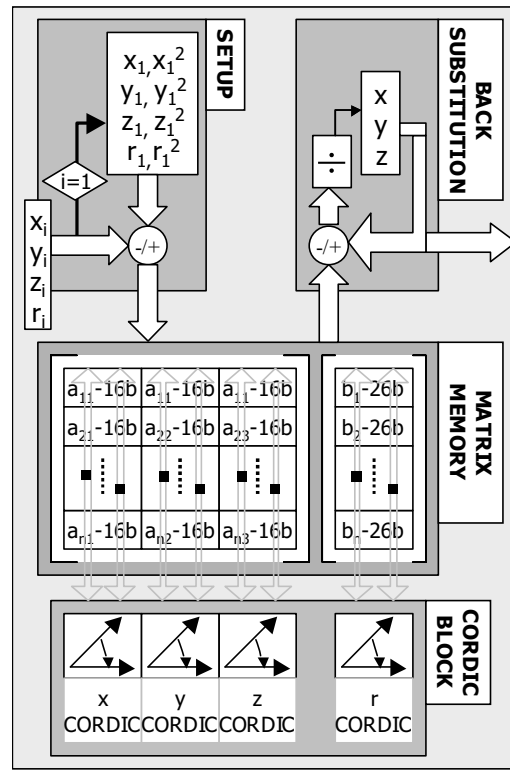


Figure 3 – LS solver block diagram

The Digital Sensor Node IC is clocked at 16MHz. At this rate, the 825 cycles take only about 52μs. Including the matrix multi-cycle divisions and other multi-cycle operations, the total time goes up to 70 μs. This computation time can easily satisfy the timing requirement discussed earlier.

The design of the localization system is performed in VHDL. Thereafter the design is synthesized, placed, routed and extracted. Following the parasitic extraction and back annotation, post layout simulations estimate 1.7mW average active power dissipation and 0.122nJ of energy consumption during the simulation. These simulations use the 16 MHz clock rate and a 1.1V supply voltage in active mode.

An equally important metric is the energy consumption per flop in the localization system. Using the energy number and

floating point operation count of 1000, which is calculated below, the energy per flop estimate is 0.122nJ/op.

It was stated in the previous section that QR decomposition with Givens rotations requires $4n^2m-4/3n^3$ flops. With $m = 15$ and $n = 4$, the expression above yields 880 flops. Including the setup of the matrix **A** and vector **b** of $E2$ as well as the final back substitutions, the operation count goes up to 1000 flops.

It should be noted however that the power and energy numbers quoted above are optimistic estimates since they do not consider the dissipation at any of the RX and TX sub blocks. However since the main contributors, the LS solver and the anchor list, are included in the simulations, the discrepancies would be negligible.

3.2 Anchor List

The second sub block of the locationing system is dedicated to maintaining the information regarding anchors known to the node. This subsystem includes a data storage that highly resembles an array of anchor objects (Object in the object oriented programming sense), where each anchor object includes the anchor coordinates and its hop distance. Figure 4 illustrates a block diagram of the implemented anchor list.

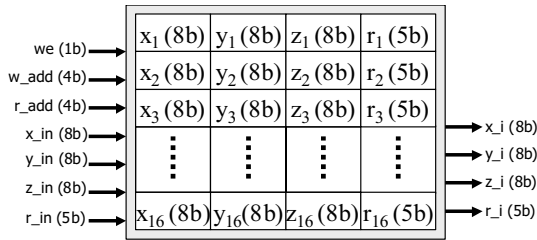


Figure 4 – Anchor List data storage.

For the particular implementation considered in this work, the maximum number of anchors is chosen to be 16. This number is selected because in a network of possibly hundreds of nodes it requires a manageable effort in preprogramming the anchor nodes. As a power of two it also allows efficient data addressing.

3.3 RX and TX Sub-blocks

In addition to the sub-block that maintains the anchor list, there are two more sub-blocks worth mentioning. First of them handles the reception and decoding of localization packets from the Data Link Layer. The second sub-block handles the creation of the localization packets and their delivery to the Data Link Layer. The position of these sub blocks is depicted in Figure 2.

The TX sub-block also handles the small tasks such as incrementing the hop count before relaying a flooding message or setting it to zero if the node is an anchor.

3.4 First Alternative Implementation: General purpose Microprocessor (μ P)

One can argue that with such relaxed timing requirements the localization system can be realized using a general-purpose microprocessor (μ P), which would ideally be embedded on the

Digital Sensor Node IC. The general-purpose computation alternatives were dismissed early on due to their expected power consumption. However this decision still needs to be quantified.

The power dissipation of general purpose μ P are usually published characteristics of these components. The energy consumption however depends on the length of operation. Hence, the duration to compute LS solution on such devices should be computed.

The μ P implementation of the localization is better equipped to compute the LS solution via Householder reflections. It is known from previous sections that 40 flops are needed for the QR decomposition. However additional clock cycles are needed for retrieving the anchor data from the memory into the data cache, setting up the matrix and back substituting the triangular matrix. Also multi-cycle division and square root operations also add more cycles to the operation.

However many embedded μ P do not have floating point hardware support [15]. These chips would execute fixed-point software to emulate these floating-point operations.

As an example, the embedded μ P in [15] dissipates 450mW at 600MHz. In this processor the LS computation takes 820 clock cycles or 1.368 μ s to finish the computation. Therefore the energy consumption becomes 547nJ for 560 flops. This yields a 0.977nJ/flop

Next consider a 1MHz, 16-b microcontroller (μ C) that is targeted for low power applications.[20] Reported active power dissipation is 0.9mW. Some multi cycle computations should be expected due to 16b hardware used to perform computations on the 19 and 26 bit data shown in Figure 3. Including all expected overheads QR decomposition with Givens rotations requires 2500 cycles (or 2.5ms) to complete. Over this period the consumed energy is 2.25 μ J and the effective energy/flop metric yields 2.25nJ/flop

For a low power system with floating point support, consider the chip set proposed in [16] and [17]. It consists of a processor and a floating-point accelerator, where the first dissipates 800mW [17] and later dissipates 152mW [16]. The clock rate is 250MHz. The computation time is calculated to take up 686 clock cycles or 2.7 μ s. The total energy consumption is 2.57 μ J for 560 flops and the energy per operation is 4.59nJ/flop.

	Time(μ s)	Power	Energy	Energy/flop
This work	71 μ s	1.7mW	0.122nJ	0.12nJ/flop
Embed'd μ P[15]	1.37 μ s	450mW	547nJ	0.98nJ/flop
Embed'd μ C[20]	2.5ms	0.9mW	2.25 μ J	2.25 J/flop
μ P w/ FP accel. [16][17]	2.7 μ s	952mW	2.57 μ J	4.59nJ/flop
DSP w/ fp support[18]	4.13 μ s	825mW	3.41 μ J	6.01nJ/flop

Table 1 Estimation of energy/flop metric for various implementations.

Finally a commercially available digital signal processor (DSP) with on chip floating point addition and multiplication support [18] is considered. The chip is clocked at 150MHz hence completes the computations in 620 cycles or 4.13 μ s. The reported power dissipation is 825mW [19] and the energy dissipation is computed 3.41 μ J for 560 flops. Therefore for a

DSP implementation energy/flop metric becomes 6.01nJ/flop.

As can be seen in Table 1 these alternative implementations consume an order of magnitude higher energy than our dedicated hardware implementation.

3.5 Second alternative implementation: Dedicated, parallel, pipelined, systolic implementation

Dedicated digital signal processor systems and circuits solving QR factorization problems have previously been reported in the literature. They are used for the adaptive nulling in multiple antenna receivers [12] and adaptive signal processing in implementing the square-root Recursive LS (or RLS) algorithms using QR factorization [13]. In both cases the final solutions involved fully parallel and pipelined solutions known as systolic arrays.

However it should be noted that the inputs to both systems are continuous stream of input samples. The flow of samples can fill up a computation pipeline and allow pipelining the data flow. Therefore, the use of systolic arrays, which implement a pipelined form of parallel computation, is an appropriate choice. In contrast, distributed Sensor network localization problem requires infrequent LS computations. A pipeline would never be completely full and the operation would be inefficient. Hence a parallel implementation would be inappropriate for this case.

4 PERFORMANCE COST DUE TO FIXED-POINT IMPLEMENTATION

The design of the localization system was performed in VHDL using fixed-point arithmetic. The position information is represented as 8-bit signed numbers for each coordinate. The hop count is an unsigned 5-bit number. With 8 bit coordinates, an indoor network locality with 100m edges would have 0.5m of resolution. Considering that one hop can correspond to up to 10m of real distance [5], 0.5m of coordinate resolution error would be smaller than error caused by hop counts used as crude distances.

The degradation from a floating point least squares solution to a fixed point least squares solution is 6% mean position error, with a maximum error of 14%. The errors are due to rounding errors during shift and divide operations.

The floating point LS solutions were computed with MATLAB while the fixed-point results were obtained by cycle-true and bit-true HDL simulations. To avoid high error due to fixed-point implementation, when necessary, the word lengths (or bit widths) were increased and outputs were scaled down. In the remaining of this section, this will be explained in more detail.

Contents of the matrix to be decomposed (\mathbf{A} in $E2$) are sums or differences of the 8-bit position inputs. Therefore they need to be represented by 9 bits. Similarly, the elements of the result vector (\mathbf{b} of $E2$) require 19 bits for each entry. This can be seen as follows. Squared 8-bit numbers become 16-bits numbers and sum of six 16-bit numbers require 19 bits to avoid overflow.

There is also data expansion possibility due to CORDIC units. After one rotation an output can increase to $\cdot 2$ of its

value. Such a case occurs when a vector with both coordinates equal to the highest k-bit number (2^{k-1}) is rotated such that one of its coordinates becomes zero. In this case the other coordinate becomes equal to the magnitude of the vector.

Since the matrix being decomposed has 3 columns, any of its entries can be rotated at most 6 times. Hence the value of the Matrix entries can increase by $\cdot 2^6 = 8$ times. This corresponds to a 3 bit wider output than the input.

In addition, the 10-stage CORDIC involves 10 additions in total. These additions can require $\text{ceil}(\log_2 10) = 4$ more bits. At the end of each rotation the CORDIC results are normalized so that this need for 4 bits would remain internal to the CORDIC rotation and not accumulate to any subsequent rotations.

In summary, 3 bits are needed for result overflow avoidance and 4 bits are needed for internal CORDIC rotation overflow avoidance. Therefore a 7-bit increase is needed during the QR decomposition stage. This requires the matrix entries be sign extended from 9 to 16 and the vector entries to be extended from 19 to 26 bits. These are also the word lengths used in the fixed point implementation.

In the back substitution, divisions are performed to yield 10-bit quotients and these 10 bit outputs are saturated down to 8 bits.

The mean position error increases significantly in cases that the 10-bit quotient needs to be saturated during conversion to an 8-bit number. During the simulations these case were encountered less than 5% LS solutions computed. The mean position error in such situations rose up to 27%.

Physically, a result that needs to be saturated implies a node falling outside the grid intended for the network and this node is assumed to be on one edge of the network. Such cases can be prevented with careful anchor topology planning. Especially positioning the anchor nodes uniformly on the periphery of the network is such a solution proves to be helpful also in other aspects of localization problems [10]. Such an approach would remedy calculated location errors due to saturated results.

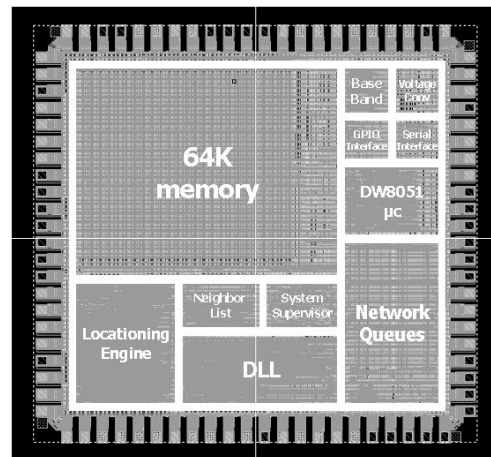


Figure 5 Layout of the Digital Sensor Node IC. Localization system is on the lower left.

4.1 Physical Implementation

The Digital Sensor Node IC, one of whose sub blocks is the localization system, has been implemented on silicon using a 0.13μ CMOS manufacturing process. The chip has dimensions of 3mm by 3mm.

The Localization system is located on the lower left corner of the Digital Sensor Node IC and occupies an area of 0.55mm^2 . The final layout is shown in Figure 5 and the localization system physical implementation results are summarized in Table 2. Note that power estimates include all parasitic effects.

5 CONCLUSIONS

An integrated and low power localization system implementation has been presented. The system serves as part of a distributed self-configuring, ad-hoc sensor network node. It calculates the sensor node position via triangulation. Triangulation is performed by computing a Least Squares solution.

Parameter	Value
Manuf. Process	0.13 μ CMOS
Area	0.55mm ²
Dimensions	707 μ x 787 μ
Gate Count	30k
Register Count	3k
Power	1.7mW
Clock Frequency	16MHz

Table 2 Localization System Physical Implementation results

Various alternative implementations were considered. A QR decomposition using Givens Rotations was selected for the LS solution algorithm. The final design exhibits 1.7mW of active power dissipation. This implies an order of magnitude active power dissipation savings with respect to a General-purpose microprocessor or DSP implementation. This shows the low power dissipation goal of the implementation has been achieved. In addition the system occupies 0.55mm^2 of silicon area and the fixed-point implementation causes a negligible degradation in the accuracy of the final location outputs.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of the industrial members of Berkeley Wireless Research Center. Bhusan Gupta and Ben Coates of ST Berkeley Labs were extremely helpful during the tape out of the Digital IC. Ada S.Y. Poon is acknowledged for her suggestions in writing this publication. This work was funded by NSF under XYZ on a chip project.

6 REFERENCES

[1] K. Langendoen, N. Reijers, "Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison"

Computer Networks (Elsevier), special issue on Wireless Sensor Networks, November 2003.

[2] J. Rabaey et al "PicoRadio supports ad-hoc ultra-power wireless networking," *IEEE Computer*, vol. 33, no. 7, pp. 42 – 48, July 2000.

[3] L. Doherty, K. Pister, and L. El Ghaoui, "Convex Position Estimation in Wireless Sensor Networks," *IEEE Infocom*, Anchorage, AK, April 2001.

[4] Savvides, H Park, and M. Srivastava, "The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems", *First ACM Inter. Workshop on Sensor. Networks and App.*, Atlanta, Septmber, 2002.

[5] C. Savarese "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks," MS Thesis, University of California, Berkeley, May 2002.

[6] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[7] A Bjorck, *Numerical Methods For Linear Least Squares Problems*, SIAM, Philadelphia, 1996.

[8] J. E. Volder, "The CORDIC trigonometric computing technique", *IRE Trans. Electron Comput.*, vol. EC-8, pp.330 – 334, 1959.

[9] C. Savarese, J. Rabaey, J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks" *Int. Conf. on Acoustics, Speech, and Signal Proc. (ICASSP)*, pp 2037 – 2040, Salt Lake City, UT, May 2001

[10] C. Savarese, K. Langendoen, J. Rabaey, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks", *USENIX Technical Annual Conference*, Monterey, CA, 2002.

[11] D. Niculescu and B. Nath, "Ad-hoc Positioning system," *GlobeCom Conf.*, November 2001.

[12] C.M. Rader, "VLSI Systolic Arrays for Adaptive Nulling", *IEEE Signal Processing Magazine*, vol: 13, Issue: 4, July 1996,pp. 29 – 49.

[13] H. Leung, H.; S. Haykin, "Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation", *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, vol: 37 ,Issue: 5, May 1989.

[14] E.A Swartzlander ed., , IEEE Computer Society Press, 1990.

[15] L.T. Clark et al. "An embedded 32-b microprocessor core for low-power and high-performance applications" *Journal of Solid-State Circuits*, vol: 36, Issue: 11, Year: Nov 2001, pp. 1599 – 1608.

[16] J. Moon et al., "An area-efficient standard-cell floating-point unit design for a processing-in-memory system" *European Solid-State Circuits Conference*, Lisbon, Portugal, Sept. 2003

[17] J. Draper et al., "Implementation of a 256-bit Wide Word Processor for the Data intensive Architecture (DIVA) Processing-In-Memory(PIM) Chip", *European Solid-State Circuits Conference*, Florance Italy September 2002.

[18] "Texas Instruments TMS320C6713- TMS320C6713B Floating point Digital Signal Processors datasheet". Literature No: SPRS186H.

[19] "Texas Instruments TMS320C6713/12C/11C Power Consumption Summary", Literature No: SPRA889

[20] Texas Instruments, MSP430F11x1A Mixed signal Microcontroller datasheet", Literature No: SLAS2416