ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.13, No.6, 1–11

VLSI architecture of a Kalman filter optimized for real-time applications

Ramón Chávez-Bracamontes[1a)], **Marco A. Gurrola-Navarro**[2], **Humberto J. Jiménez-Flores**[3], **and Manuel Bandala-Sánchez**[1]

[1] *Centre for Engineering and Industrial Development, Department of Microsystems Research, 702 Pie de la Cuesta, 76125, Querétaro, Mexico*

[2] *Universidad de Guadalajara, Electronics Department, CUCEI, 1421 Blvd. Gral. Marcelino García Barragán, Guadalajara, Mexico*

[3] *Tecnológico de Monterrey, Campus Querétaro, 500 Epigmenio González Col. San Pablo, 76130, Querétaro, Mexico*

a) *rachavez@cidesi.edu.mx*

Abstract: This paper presents a parametrized VLSI architecture for an n-state Kalman filter implementation intended for real-time applications that typically require a sensing rate not far from 300 samples per second. The architecture has been optimized in silicon area and power consumption. This approach has been proved with a fabricated chip using a 0.5 µm CMOS technology. The fabricated integrated circuit executes a two-state Kalman filter employing 70 K transistors. For a performance of 50 filter iterations/second, the chip requires a clock frequency of 200 KHz where a negligible power consumption of 1.1 mW is observed. This performance can be increased up to 176,991 iterations/second at a clock frequency of 20 MHz.

Keywords: Kalman filter, on-chip algorithm, VLSI, CMOS

Classification: Electron devices, circuits, and systems

References

[1] R. E. Kalman: J. Basic Eng. **82** (1960) 35. DOI:10.1115/1.3662552

[2] M. S. Grewal and A. P. Andrews: *Kalman Filtering Theory and Practice Using MATLAB* (Wiley-IEEE Press, New York, 2015) 4th ed. 281.

[3] Y. Guo: IEEE EUROCON (2007) 2503. DOI:10.1109/EURCON.2007.4400366

[4] A. García-Quinchía, Y. Guo, E. Martin and C. Ferrer: IEEE ISIE (2009) 603. DOI:10.1109/ISIE.2009.5220306

[5] A. Sudarsanam, R. Barnes, J. Carver, R. Kallam and A. Dasu: IET Comput. Digit. Tech. **4** (2010) 126. DOI:10.1049/iet-cdt.2008.0139

[6] K. R. Santha and V. Vaidehi: IEEE ICSCN (2007) 172. DOI:10.1109/ICSCN.2007.350725

[7] K. Sakkay, D. Massicotte and A. Barwicz: IEEE CCECE (1998) 357. DOI:10.1109/CCECE.1998.682758

[8] Z. Youmin, P. Quan, Z. Hongcai and D. Guanzhong: IEEE Proc. of the 32nd Conference on Decision and Control (1993) 3590. DOI:10.1109/CDC.1993.325888

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

[9] A. Barwicz, D. Massicotte, Y. Savaria, P. A. Pango and R. Z. Morawski: IEEE Trans. Instrum. Meas. **44** (1995) 720. DOI:10.1109/19.387317
[10] A. L. T. Mozipo: IEEE CCECE (1999). DOI:10.1109/CCECE.1999.807259
[11] A. Chacón-Rodríguez, P. Julian and F. Masson: Electron. Lett. **46** (2010) 533. DOI:10.1049/el.2010.2669
[12] D. Simon: *Optimal State Estimation* (Wiley-Interscience, New J., 2006) 1st ed. 552.

1 Introduction

The Kalman Filter (KF) provides a real-time solution to recursively estimate an unknown state vector from a given measurement in a linear dynamic system. This filter is used in several control applications, such as navigation systems. In the literature, there has been several formulations for the KF based on the conventional formulation proposed by Rudolf Kalman in 1960 [1, 2]. For most applications, the computation involved in updating the filter estimations is notoriously demanding ($O[n^3]$ per iteration, where n is the number of states). This work is proposed particularly for the application of the KF in inertial navigation. For instance, dynamic inclination sensing in rotational inertial systems is important for unmanned aerial vehicles, such as radio-controlled helicopters, planes and drones. In these applications the maximum processing speed is about 300 KF iterations/second, limited by gyroscopes and accelerometers bandwidth.

For real-time applications, implementations in either a powerful embedded system or in a modern PC have been preferred. Several authors have developed KF implementations in different platforms such as FPGAs [3, 4, 5, 6] and DSPs [7]. Others have presented simulations in parallel computing hardware [8]. Few authors have proposed VLSI implementations of synthesized KF algorithms and tested a fabricated chip. A number of implementations are intended for high-speed applications such as adaptive channel equalization [9], spectrometric measurement [10], and localization of impulsive noise [11]. These implementations require parallel architectures such as systolic or semi-systolic arrays since their goal is to increase the speed of the KF iterations. Only two works present a synthesized VLSI architecture where either the corresponding physical layout or a fabricated chip has been tested [10, 11]. Specifically, [10] reports a synthesized (unfabricated) chip that required 750 K transistors. The chip performs a three-state KF algorithm with a 20-bit fixed-point data representation using a parallel systolic architecture. A non-parallel approach is presented with a remarkably simplified version of the filter. This KF implementation was specifically designed to work along with an acoustic sensor network [11]. This approach considers a one-state system where the steady-state Kalman gain is calculated off-line.

2 Design considerations

In a common mobile inertial navigation application, the maximum speed expected from a KF is in the range of 300 KF iterations/second since that is the typical bandwidth offered by most commercially-available MEMS-based inertial sensors.

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

Hence, under such conditions, there is no need to run the KF algorithm at high speed. Therefore, when designing a digital VLSI integrated circuit, it is preferred to keep the silicon area as small as possible for budgetary reasons, and the power consumption at acceptable levels. For inertial sensing applications, it can be proved that estimating the states of the filter at a low rate, a single processor architecture is sufficient to process up to a ten-state KF while maintaining a performance of a few hundreds iterations/second. It is important to point out that most of the published works on VLSI KF architectures, these are presented either in equation form or in a block diagram.

In this paper, a parametrized architecture that is capable of computing a KF up to n-states is proposed. Nevertheless, a two-state KF with a single measurement input is deployed in order to demonstrate the characteristics of the architecture. This particular configuration has been fabricated and the resulting chip has been tested.

3 Algorithm requirements

The discrete-time model for a linear stochastic system is

$$X_k = \Phi_{k-1}X_{k-1} + G_{k-1}U_{k-1} + w_{k-1}, \tag{1}$$

and the observational model is

$$Y_k = H_kX_k + v_k. \tag{2}$$

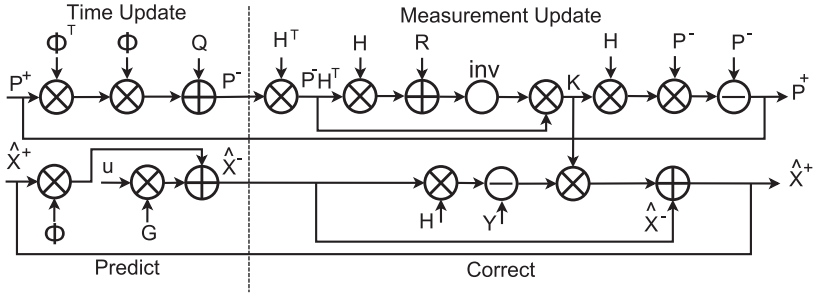


Fig. 1. Operation dependencies in the KF algorithm.

where $X_k \in \Re^n$, $U_k \in \Re^m$, $Y_k \in \Re^r$, w_k and v_k are independent purely random Gaussian sequences with zero-mean that have covariance matrices Q_k and R_k for the process and the measurement respectively. Fig. 1 shows the operations dependencies of the KF algorithm evaluated in two steps, *time update*, equations (3) and (4), and *measurement update*, equations (5), (6) and (7).

$$\hat{X}_k^- = \Phi_{k-1}\hat{X}_{k-1}^+ + G_{k-1}U_{k-1}, \tag{3}$$

$$P_k^- = \Phi_{k-1}P_{k-1}^+\Phi_{k-1}^T + Q_{k-1}, \tag{4}$$

$$K_k = P_k^-H_k^T(H_kP_k^-H^T + R_k)^{-1}, \tag{5}$$

$$\hat{X}_k = \hat{X}_k^- + K_k(Y_k - H_k\hat{X}_k), \tag{6}$$

$$P_k = (I - K_kH_k)P_k^-. \tag{7}$$

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

The algorithm is typically programmed with a high-level language in a PC using a floating-point numeric system. Nevertheless, as it has been stated before, when the KF is used along with real-time sensors, the implementation on embedded systems or VLSI with a fixed-point numeric system should be considered.

In order to allow the testing conditions to be as close as possible as a real-time application, a set of test signals were experimentally obtained from the inertial sensor Invensense MPU6000^{TM} with a 16-bit output resolution. The data were later used for comparison in a simulation run in Matlab©. This simulation implemented a KF using a floating-point 64-bit precision format. This simulation was taken as reference.

On the other hand, a careful analysis has been done in order to find the adequate word length representation for KF inertial applications. In order to spare silicon area, the use of fixed-point over the floating-point numeric representation was proposed with a sign and magnitude numeric format. The fixed-point sign and magnitude format required a single-bit for the sign, 9 bits for the integer part (enough to represent a 360° range of rotation), and f bits for the fractional part.

Fig. 2 illustrates that when the KF is evaluated with $f < 9$ bits, the algorithm presents a large error compared to the reference simulation. On the other hand, it can be seen that when $f = 9$ bits is sufficient to reach the expected response close to the reference simulation. Even though $f = 9$ showed to be sufficient, it was conservatively decided to use $f = 14$ bits for the fractional part, resulting in a word-length of 24 bits.

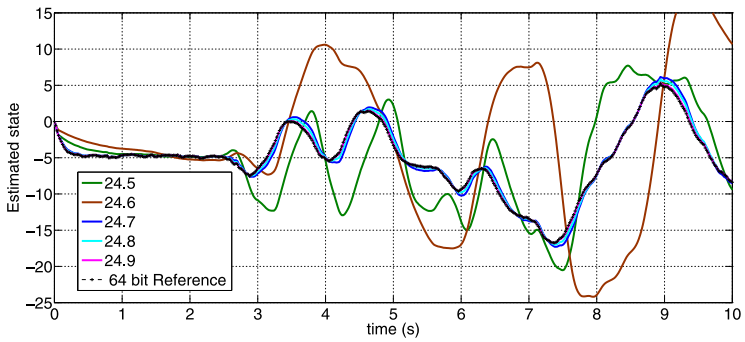


Fig. 2. KF fixed-point simulations for different word lengths in the fractional part compared to the 64-bit reference.

4 System architecture

The proposed architecture is shown in Fig. 3. It is a simplified microprocessor architecture with a simple instruction set that includes only the essential operations for the KF algorithm: addition/subtraction, multiplication, and multiplicative-inverse. The system executes the program from an internal ROM with the help of a sequencer that includes a program counter (PC). The ROM is composed by 256 16-bit registers. When the external RESET signal is high, the output READY is set active. When READY is active the PC does not increment. When the external

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

START signal is active the PC is set to point to the 0x00 address and the system begins the execution of the ROM instructions. The PC will increment by 1 when either an addition/subtraction or a multiplication operation is executed (both performed in 1 clock cycle), and when the inverse operation is completed (performed in 24 clock cycles). The architecture has no branch instructions. The last instruction of the program is the stop or HALT instruction. When HALT is executed the READY signal is again activated and the PC stops incrementing. When the READY signal is active, the result can be read from the DATA_OUT bus. Consequently, the next sample can be written to the Data-Bank by using DIR and the DATA_IN bus pointing to a specific NR register and by applying an external WRITE control signal to store the sample.

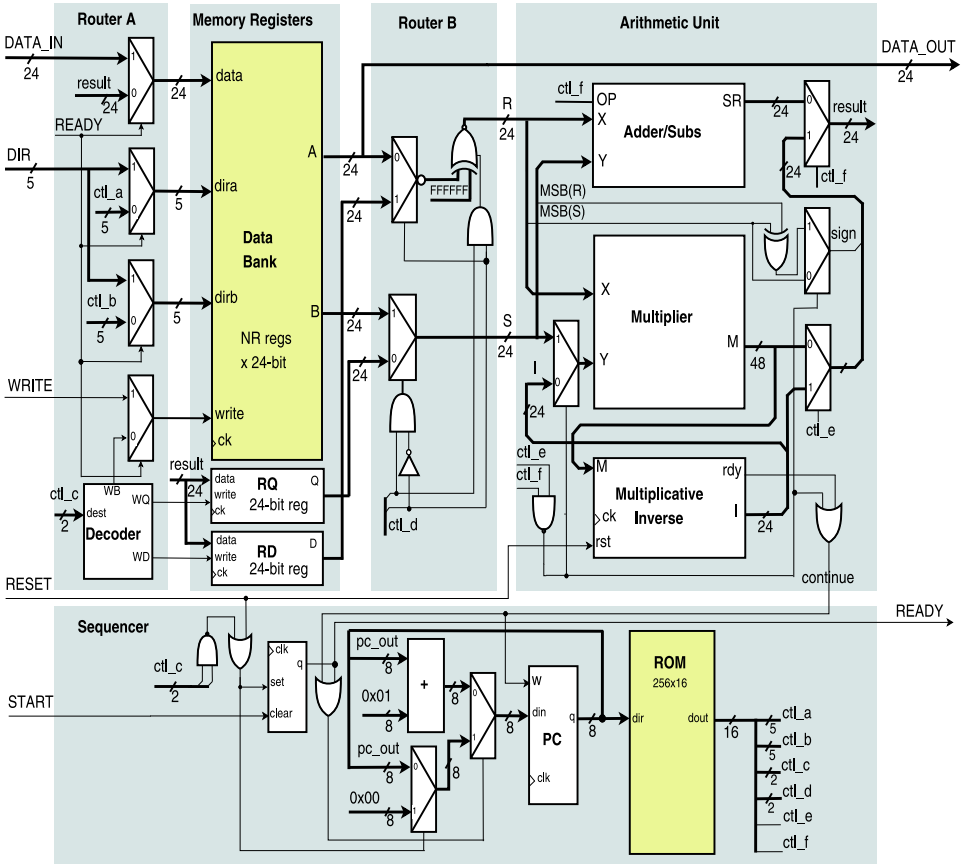


Fig. 3. Proposed KF architecture.

No instruction fetch cycles or any instruction register are required. The sequencer block provides the instruction codes to compute all the operations required for the KF iteration. The KF matrix operations are split into their simplest scalar arithmetic operations (shown in Table I) and are codified in a 16-bit word/instruction formatted as "$a_4\, a_3\, a_2\, a_1\, a_0\ b_4\, b_3\, b_2\, b_1\, b_0\, c_1\, c_0\, d_1\, d_0\, e\, f$".

The performance of a KF depends upon the clock frequency and the computational complexity of the algorithm formulation [2]. The complexity derived for the conventional KF formulation is a third order polynomial

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

Table I. KF matrix operations and their computational complexity.

Eq.	Operation	(×)	(+)	(−)	(a^{-1})
3	$\Phi\hat{x}_k$	n^2	$n(n-1)$		
	Gu_k	nm	m		
4	$P_k\Phi^T$	n^3	n^2		
	$\Phi P_k\Phi^T$	n^3	n^2		
	$\Phi P_k\Phi^T + Q_k$		n^2		
5	$P_kH_k^T$	n^2	$n(n-1)r$		
	$H_kP_kH_k^T$	nr^2	$(n-1)r^2$		
	$H_kP_kH_k^T + R_k$		r^2		
	$(H_kP_kH_k^T + R_k)^{-1}$				$24r+1$
	$P_kH_k^T(H_kP_kH_k^T + R_k)^{-1}$	nr^2	$n(r-1)$		
6	$H_k\hat{x}_k$	nr	$r(n-1)$		
	$z_k - H_k\hat{x}_k$			r	
	$K_k(z_k - H_k\hat{x}_k)$	nr	$n(r-1)$		
	$\hat{x}_k^- + K_k(z_k - H_k\hat{x}_k)$		n		
7	K_kH_k	n^2r	n^2		
	$K_kH_kP_k$	n^3	n^2		
	$P_k - K_kH_kP_k$		n^2	n^2	

$$O[n^3] = (2n^3 + 5n^2 + nm + m - n) + r(n^3 + 5n^2 + 4nr + 3nr^2 + 2n^2r + 25r + 1) \quad (8)$$

where n is the number of states to be estimated, r is the number of measurements and m is the number of control references.

A complex instruction decoder is not required since each instruction bit is used directly as a control signal (exceptions are bits c_1 and c_0 that require a small decoder). For the reasons stated in the previous section, the arithmetic blocks were defined for a fixed-point sign-and-magnitude numeric representation using 14 bits for the fractional part, 9 bits for the integer part and one bit for the sign. The *arithmetic unit* includes the following blocks: *adder-subs*, *multiplier* and the *multiplicative-inverse*. The multiplier is a combinational circuit that allows multiplications to be performed in a clock cycle. The *multiplicative-inverse* follows a successive approximation technique using the *multiplier* block and an internal numeric comparator to determine the result bit by bit, requiring 24 clock cycles to finish the task. Although the *multiplicative-inverse* implementation is time consuming, it is known that the proposed KF model requires few matrix inversion operations (i.e. the algorithm requires only a single multiplicative-inverse when the system has only one measurement, $m = 1$). For $m > 1$, sequential filtering [12] can be deployed in order to process measurements one at a time.

The number of 16-bit registers required in the ROM depends on Eq. (9). The 256-instruction ROM shown in Fig. 3 is sufficient to solve a three-state one-measurement KF. In any case, the word length of the PC register must be increased in order to be able to address all the instructions of a bigger ROM.

A KF requires a limited amount of memory registers to storage temporary data from the previous KF iteration and for manipulating the vectors and matrices of the filter's parameters during the solving of the present iteration. Eq. (10) represents the

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

number of 24-bit registers (NR) required in the Data-Bank. This is dependent on the number of states and measurements to be estimated. 32 registers of the data-bank shown in Fig. 3 are sufficient to estimate a two-state, one-measurement KF. Similarly, 1024 registers should be sufficient for a ten-state, 10-measurement KF. Note that the word length of the instruction code should also be increased in order to access the addresses in a data-bank with a larger amount of registers. Additionally, the architecture includes registers *RQ* and *RD* that are used as accumulators for temporary operands. Blocks *router A* and *router B* set the correct data-path configuration for each instruction.

$$ROM = (3n^3 + 10n^2 + 2n^2r + 3nr^2 + 4nr + nm + m - n + r) \quad (9)$$

$$NR = 5(n^2) + (r^2) + 2(nr) + (nm) + (n + r + m) + 1 \quad (10)$$

5 VLSI fabrication

In order to confirm the applicability of the proposed architecture, an integrated circuit has been designed and manufactured for the two-state, one-measurement and one control input KF case (i.e. $n = 2$, $r = 1$, $m = 1$). The chip was designed following a digital VLSI design flow using the tools and standard cell libraries of the Alliance/LIP6© CAD Software. The system was defined using a combined behavioural and structural VHDL description. The regular structures of the architecture (i.e. register bank, adders) were placed in a designer-guided manner, while the random logic structures (i.e. state machines) were synthesized from the VHDL behavioural description. Both, the placement and routing (interconnections) of the cells were performed using automatic tools. The CMOS target technology was On Semi C5F with a feature size of 0.5 µm and 3 metal layers. Finally, Tanner EDA© has been used for design-rule-checking (DRC) and layout-versus-schematic (LVS) physical verification. The resulting layout and a microphotograph of the manufactured chip are shown in Fig. 4. The manufactured chip required approximately 70 K transistors in a silicon area of 5.6 mm^2, demanding a power consumption of 1.1 mW.

Using post-layout results and the measurements from the manufactured chip, Tables II and III were obtained. The Tables show the estimated transistors and clock cycles required for hypothetical chips, using the proposed architecture, as a function of the number of states and measurements (n, r). It is shown in Table III that each multiplicative inversion requires 24 clock cycles. In the case of the already fabricated chip, dynamic flip-flops in the data-bank block were used. Hence, significant silicon area was reduced compared to having used static flip-flops. The maximum clock frequency used for the selected 0.5 µm CMOS technology was 20 MHz. After the post-layout LVS verification was completed, the functionality of the proposed design was verified by means of a digital level simulation, where the experimentally-obtained signals mentioned in section 3 were used. As expected, the response of the fabricated chip matched the digital simulation presented in Fig. 2.

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

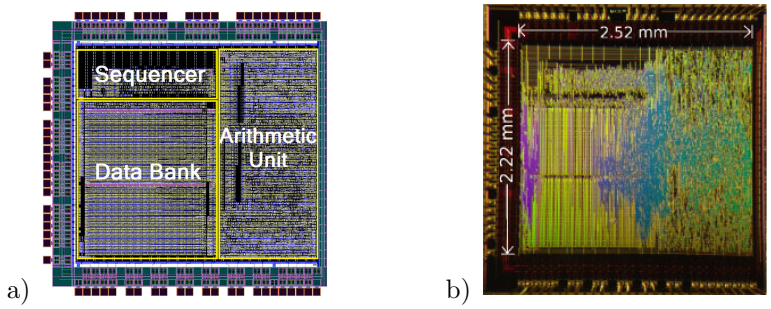


Fig. 4. a) Synthesized layout. b) Microphotograph of the manufactured core (5.6 mm^2).

Table II. Transistors and power with respect to the number of states

n ($r = 1, m = 1$)	2	3	4	5	6	7	8	9	10
Transistors (thousands)	70	101	144	197	261	336	422	519	627
Power (mW @ 10 KHz)	0.27	0.40	0.56	0.77	1.03	1.32	1.74	2.05	2.47
Power (mW @ 20 MHz)	55.3	80	112	154	206	264	348	410	494

Table III. Clock cycles per iteration

	Measures									
n	1	2	3	4	5	6	7	8	9	10
2	113	309	661	1205	1977	3013	4349	6021	8065	10517
3	237	540	1063	1860	2985	4492	6435	8868	11845	15420
4	439	889	1631	2737	4279	6329	8959	12241	16247	21049
5	737	1380	2395	3872	5901	8572	11975	16200	21337	27476
6	1149	2037	3385	5301	7893	11269	15537	20805	27181	34773
7	1693	2884	4631	7060	10297	14468	19699	26116	33845	43012
8	2387	3945	6163	9185	13155	18217	24515	32193	41395	52265
9	3249	5244	8011	11712	16509	22564	30039	39096	49897	62604
10	4297	6805	10205	14677	20401	27557	36325	46885	59417	74101

6 Results and discussion

Using the proposed approach, a three-state KF fabricated chip would require 101 K transistors for a 24-bit fixed-point data representation in contrast to the 750 K transistors in the layout synthesized by [10] with a three-state KF with a 20-bit fixed-point data representation.

Even though the proposed architecture is intended for a low iteration rate, and the one presented in [10] is intended for a high-speed application (channel equalization), both architectures were fabricated with a similar 0.5 µm CMOS technology. In order to compare the two implementations, a figure of merit (FOM) has been defined as the inverse of the product of the number of transistors in the design and the latency at the maximum usable frequency. For the architecture presented by [10], the maximum reported frequency is $f_c = 3$ MHz and the latency is defined by $(3 + 4n)/f_c$, where $n = 3$ is the number of states. This results in a FOM of 0.267 Hz/transistor. For the proposed architecture, using $n = 3$, 101 K transistors,

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

237 clock cycles per iteration at 15 MHz, the FOM is 0.623 Hz/transistor. The figure of merit of both architectures are in a similar order of magnitude.

Tables IV and V show the performance for different frequencies and the latency behaviour with respect to the number of states respectively. It can be seen, for the simplest filter ($n = 2$) at 10 KHz, the performance is 113 iterations/second with a 11.3 ms latency and 0.27 mW maximum power consumption. These figures are suitable for inertial applications. Understandably, by increasing the frequency up to 20 MHz, the chip will drain more power.

Table IV. Performance with respect to frequency when $n = 2$, $m = 1$.

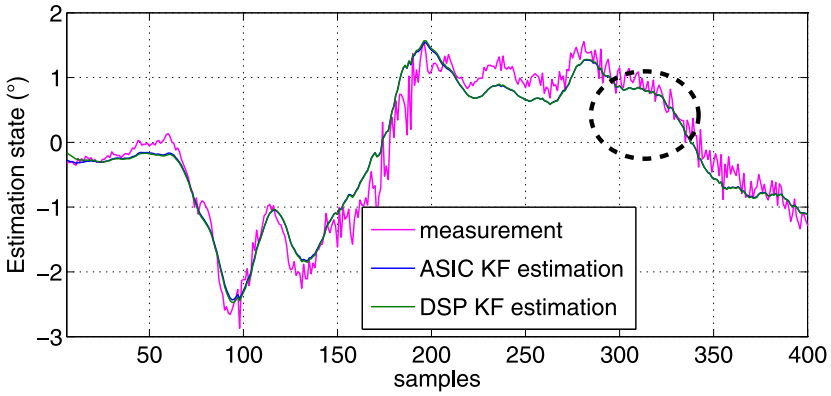
Frequency (MHz)	Latency (ms)	Performance (updates/second)
0.01	11.3	113
0.05	2.26	442
0.20	0.56	1,770
2	0.056	17,700
12	0.011	106,194
15	0.007	132,743
20	0.005	176,991

Table V. Latency behaviour with respect to number of states at 20 MHz

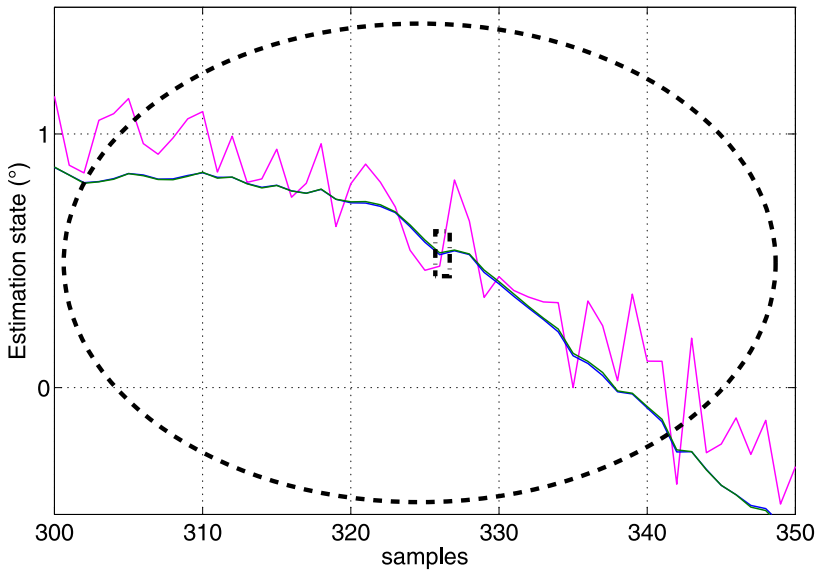
n	cycles	Latency (µs)
2	113	5.7
3	237	11.9
4	439	22.0
5	737	36.9
6	1149	57.5
7	1693	84.7
8	2387	119.4
9	3249	162.5
10	4297	214.9

The performance of the manufactured chip was tested along with the Invensense MPU6000^{TM} inertial sensor. A PCB containing this two elements was attached to a rotational mechanical system. External disturbances were introduced to this configuration. The sensor data were sent directly to the chip that performed the KF algorithm at 50 samples per second. Test results are illustrated in Fig. 5. These tests have resulted as expected, i.e. identical to the Matlab simulations. Note that in Fig. 5, along with the chip measurements, the results obtained with a commercially available DSP are shown. The algorithm was also performed by the DSP using a fixed-point 16-bit data representation. This extra implementation was tested under identical conditions as the manufactured chip. While both devices have almost an identical response, the manufactured chip computes the algorithm in 113 clock cycles, while the DSP needs 47,168 clock cycles. Hence, requiring 417 times as many clock cycles. The power consumption of the manufactured chip is nearly 1.1 mW at 200 KHz (for the selected CMOS technology) in contrast to the 280 mW required by the commercial DSP.

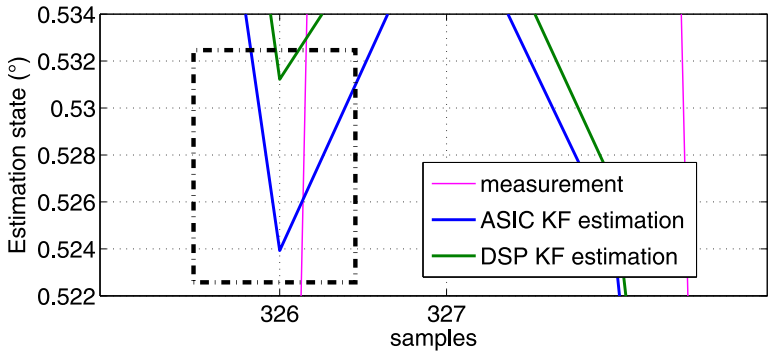
DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016



(a)



(b)



(c)

Fig. 5. Manufactured chip compared to a DSP implementation. Both experiments were run at 50 samples per second. (a) Complete signal. (b) Zooming out of the signal. (c) Close comparison of the estimations.

7 Conclusion

A specific VLSI architecture intended for inertial mobile applications of the KF has been presented. The architecture can be parametrized for a different number of states and measurements of the KF. The approach has been validated by means of a

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016

fabricated chip performing a two-state, one-measurement KF requiring 70 K transistors. The latency of the presented architecture is 10 times the latency of a previously reported work. However, this feature is purposelessly the result of a low frequency estimation. Moreover, the proposed architecture requires approximately 7 times less transistors compared to that previous report. The proposed architecture is significantly efficient regarding silicon area utilization and power consumption for KF inertial applications compared to other reported approaches.

Acknowledgments

This work was supported by the Mexican National Council for Scientific and Technological Development, scholarship 262353. The fabrication costs were covered by the MOSIS MEP Instructional program, project 92159.

DOI: 10.1587/elex.13.20160043
Received January 19, 2016
Accepted January 28, 2016
Publicized February 26, 2016
Copyedited March 25, 2016