

# Low-power LFSR Kernel Architecture in Mobile Transmitter and Receiver Protocols and Software Defined Radios

C N Khairnar<sup>1</sup>, Raj Kamal<sup>2</sup>, and Sanjiv Tokekar<sup>3</sup>

<sup>1</sup>Department of Basic Studies, Faculty of Communication Engineering, Military College of Telecommunication Engineering, Mhow(M.P)-453441, India, Email: cnkhairnar@yahoo.com

<sup>2</sup>School of Computer Science, Electronics and Information Technology, Devi Ahilya University, Indore (M.P)-452017, India, Email: dr\_rajkamal@hotmail.com

<sup>3</sup>Department of Electronics and Telecommunication Engineering, IET, Devi Ahilya University, Indore (M.P)-452017, India, Email: sanjivtokekar@yahoo.com

**Abstract**— The LFSR operations are performed in implementing the different mobile-device protocols and encryption standards during transmission and reception. These operations are performed at the PN coding stages in CDMA, Gold code generation stage in WCDMA and several operation stages in the mobile transmitter and receiver protocols in Software Defined Radios (SDRs).

A kernel can instantiate several processing objects when processing in the processor. Use of the kernel is known to lead to reduced energy dissipation by reducing the need of just a few memory accesses. A kernel for the multi-bit output Linear Feedback Shift Register (LFSR) operations is proposed in this paper. An Instruction Set Architecture (ISA) proposed for the kernel gives the functional instantiation at the stages of generation of PN long sequence, PNI short sequence and PNQ short sequence, M-sequence, Gold code sequence, CRC sequence and Signature sequence in the SDR link-applications, and other wireless applications.

The paper also gives the results of computations of dynamic power consumption (DPC) in the kernel architecture. The estimation of required number of memory accesses of the kernel based upon the proposed ISA is also given in the paper. The LFSR kernel is shown to need just a few memory accesses, thus significantly reducing the power requirement in comparison to the software implementation of the above stages.

**Index Terms**— LFSR kernel, Switching activity factor ( $\alpha$ ), Dynamic power consumption, Memory accesses, Software Defined Radio

## I. INTRODUCTION

It has been shown that deploying a processing element (PE) reduces the number of MIPS (Million Instructions Per Second) and thus energy dissipation in the SDRs [1],[2]. Deploying the (i) CMAC (Complex Multiply and Accumulate Unit) as processing element and (ii) smart address generator reduce the number of MIPS by one sixth for several operational-stages; also the use of memory sharing and Direct Memory Access (DMA) reduces the requirement of memory accesses and thus reduces the MIPS cost in power dissipation by 30% [1]. It has also been shown [3], [4] that a kernel, an object having a programmable micro-sequencer, local memory,

PE and appropriate input-output interface also reduces the number of MIPS and thus the requirement of memory accesses. The kernel also reduces the MIPS cost in power dissipation. The kernel is deployed as a computational subunit in the processor that instantiates the objects with negligible external memory accesses and that has a few input/output interfaces to main processor(s) and memory. Use of an appropriate kernel, for example, Butterfly kernel reduces:

- (i) number of memory accesses significantly and
- (ii) number of Mops (Million operations per second) of the processor in given time slot and thus clock frequency with the processor that is required for finishing the operations.

The Butterfly kernel has the property of functionally instantiating the number of operations [3], for example, Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Forward Hadamard Transform (FHT) and several other operations.

The energy consumption in Software Defined Radios (SDRs) thus reduces by combination of the followings actions [1], [2], [3]:

- 1) reducing the operational voltages,
- 2) using the gated clocks,
- 3) reducing the number of Mops for the given schedule of the operations thus reducing the operational clock frequency and operational voltage requirements for that schedule,
- 4) reducing the number of memory accesses required at the various stages of the operations by smart address generation units (AGUs),
- 5) distributing the scratchpad memories and register files or use of local memory in operational units,
- 6) deploying the processing elements in order to functionally reduce the memory accesses,
- 7) deploying the kernels, which have the input-output interface, local memory and micro-sequencers in order to functionally instantiate the objects required in several operations.

There is need to explore the deployment of additional operational elements and kernels in the SDRs which are

like the CMAC [1] and Butterfly kernel, Add Compare and Select (ACS) Kernel [3], [4] that can instantiate other operations in mobile standard protocols or in the SDRs. LFSR kernel is explored in this paper.

LFSR is required in the Pseudo Noise (PN) coding stages in the Code Division Multiple Access (CDMA) and Wideband Code Division Multiple Access (WCDMA) [2] mobile transmitter and receiver protocols and in low power cryptographic key generation [5], [6].

A genetic algorithm approach is given in [7] for specific search of a LFSR configuration. LFSR kernel/object has to be based on a typical class of algorithms working at the bit level. The output sequences of LFSR kernel is cyclic and have explicit properties of randomness. The energy efficiency of LFSR kernel depends upon its configuration. The enhancement in processing speed of LFSR kernel algorithms has been obtained by increasing LFSR phase cycle computation at the same time [8]. To obtain a low power design of proposed LFSR kernel, it required a systematic optimization approach at all levels of the design [9].

LFSR kernel and optimized ISA for it is proposed in this paper. The proposed LFSR kernel is applicable for the functional instantiation in generation of PN long sequence, PN short sequence and PNQ short sequence, M-sequence, Gold code sequence, Cyclic Redundancy Check (CRC) sequence and Signature sequence in SDR link applications and other wireless applications.

It is described in [5] that increased power dissipation in LFSR leads to component failure. The computation of dynamic power consumption (DPC) is thus important. Previous work on the computation of DPC of different LFSR architectures is as follows:

- 1) the computations are carried out for fixed switching activity factor of 50% and considering switching capacitance of a logic element and D flip-flop [10].
- 2) multi-bit output LFSR low-power architecture is presented and its DPC has been calculated in [11]. It is also described that reduced switching activity of a gate with constrain on its physical size can further reduce DPC. The LFSR architecture is based on multi-bit output interface in several clock cycles and thus reducing the operating frequency and hence switching activity.

The detailed dynamic power consumption is also computed in this paper for the proposed kernel architecture. It is shown to

- (i) improve upon the throughput and
- (ii) reduce the DPC and memory accesses per unit performance of implemented algorithms after functional instantiation.

## II. LFSR KERNEL

The low dynamic power consumption in LFSR kernel is critical due to use of computationally intensive signal processing algorithms at bit level in SDR link and other wireless applications. Figure 1 shows the kernel and

instantiation of new objects from that. With appropriate instantiated objects and extension, the kernel can be used as the building block for generation of PN long sequence, PN short sequence and PNQ short sequence [12], M-sequence, Gold code sequence [13], CRC sequence and Signature sequence [11] for use in converged SDR platform and several other operations. The type-II polynomial forms are used as the basic element of the LFSR kernel as they give correct remainder and quotient, and suitable for cyclic code applications.

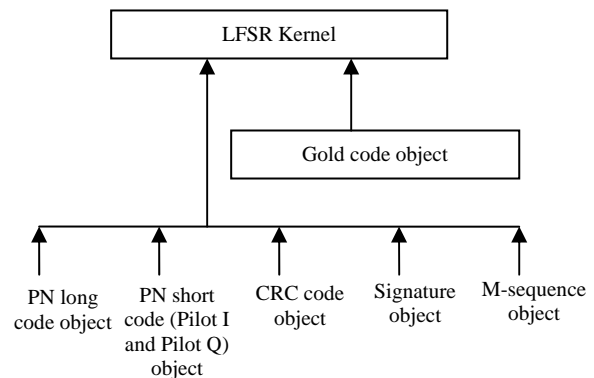


Figure 1. LFSR kernel and instantiation of its objects.

The hardware requirement of LFSR kernel has been obtained from low-power multi-bit output LFSR architecture design proposed in [11]. For generating PN long sequence, PN short sequences, CRC sequence, and Signature sequence only corresponding polynomial coefficients has to be updated by fetches from external (non-local) memory. The Gold code sequence generation is obtained by the lookup table approach by combining two M-sequences and modulo-2 adding the output of these sequences [13]. An ISA of proposed LFSR kernel is motivated from the work described in [1], which in specific avoids energy cost carrying multiplication and modulus operations in computations. In addition, special method is adapted to reduce energy-cost of computationally intensive modulus operations [14]. The efficient operation in the kernel is obtained by confining processing locally and use of local memory [15], and by avoiding redundant computations.

## III. HARDWARE REQUIREMENT OF LFSR KERNEL FOR LOW-POWER ARCHITECTURE

The hardware requirement of LFSR kernel/object functions is obtained by the approach given in [11]. Table I shows hardware requirement of different LFSR kernel/object functions (polynomials are given in Table II). In the LFSR kernel/object function, maximum number of flip-flops changing state every clock cycle is equal to the lowest non-zero exponent of a term in that polynomial. The polynomial with high lowest non-zero exponent and low value of number of taps or XORed terms obtains major reduction in hardware for a LFSR kernel/object function.

TABLE I  
HARDWARE REQUIREMENT OF DIFFERENT LFSR KERNEL/OBJECT FUNCTIONS

Sr. No.	Kernel/object Function (order of polynomial)	Number of phases required	Number of OR gates	Average number of inputs to each OR gate	Number of switches	Number of Flip-Flops	Number of XOR gates	Number of inputs to each XOR gates
1	PN long (42)	42	42	42	20	42	1	20
2	PNI short (15)	3	3	≈2	60	15	5	6
3	PNQ short (15)	5	5	≈3	45	15	3	8
4	Gold code (25)	9	9	≈2	28	25	3	2
5	Gold code (25)	25	25	25	4	25	1	4
6	CRC 1 (16)	4	4	≈2	35	16	5	3
7	CRC 2 (32)	32	32	56	8	32	1	14
8	Signature (05)	3	3	≈2	7	5	2	2
9	M-sequence (08)	3	3	≈3	17	8	3	4

IV. DYNAMIC POWER CONSUMPTION (DPC)

The dynamic power consumption (DPC) estimation in low-power architecture for different kernel/object function is obtained using the same method as described in [11] and given in Table II. The 0.18 μm, CMOS process, with gate capacitances (C<sub>gate</sub>); capacitance of D flip-flop (C<sub>df</sub>) = 0.0027 pF, capacitance of XOR gate (C<sub>XOR</sub>) = 0.0042 pF, capacitance of OR gate (C<sub>OR</sub>) = 0.0026 pF, capacitance of inverter gate(C<sub>INV</sub>) = 0.0027 pF, capacitance of AND gate (C<sub>AND</sub>) = 0.00215 pF, operating voltage (V<sub>dd</sub>) = 1.8 V, fixed switching activity factor, α = 0.5 and clock frequency (f<sub>clk</sub>) of 30 MHz is used in calculation of DPC. The DPC is calculated as given below [11]:

$$DPC = C_{gate} \times V_{dd}^2 \times f_{clk}$$

Table II shows that polynomial with high lowest non-zero exponent of a term in kernel/object functions improves DPC. The DPC can be further corrected when the exact switching activity factor (α) and thus the effective switching capacitance of a logic element is considered in DPC estimation. The Figures 2, 3, 4, 5 and 6 shows the DPC in hardware requirement of LFSR kernel/object in phase generator, OR gates, flip-flops, XOR gates and total DPC respectively. In these figures, Gold code polynomial with highest DPC is considered. Total DPC in specific LFSR kernel is estimated after considering DPC in phase generator, OR gates, flip-flops and XOR gates and then dividing it by lowest non-zero exponent term of polynomial.

TABLE II  
DPC ESTIMATION IN LOW-POWER ARCHITECTURE FOR DIFFERENT FUNCTIONAL INSTANTIATIONS OF THE KERNEL/OBJECT

Sr. No.	Kernel/object Function (order of polynomial)	Polynomial	DPC in Phase generator (μW)	DPC in OR gates (μW)	DPC in Flip-Flops (μW)	DPC in XOR gates (μW)	Total DPC (μW)
1	PN long (42)	$f(x)=1+x+x^2+x^3+x^5+x^6+x^7+x^{10}+x^{16}+x^{17}+x^{18}+x^{19}+x^{21}+x^{22}+x^{25}+x^{26}+x^{27}+x^{31}+x^{33}+x^{35}+x^{42}$	18.1	5.1	0.1	0.3	23.5
2	PNI short (15)	$f_1(x)=1+x^5+x^7+x^8+x^9+x^{13}+x^{15}$	1.2	7.6	2.6	1.7	2.6
3	PNQ short (15)	$f_0(x)=1+x^3+x^4+x^5+x^6+x^{10}+x^{11}+x^{12}+x^{15}$	1.6	6.1	1.2	1.0	3.3
4	Gold code (25)	$f(x_1)=1+x^3+x^{25}$	1.3	1.2	0.4	1.0	1.4
5	Gold code (25)	$f(x_2)=1+x+x^2+x^3+x^{25}$	11	1.0	0.021	0.3	12.3
6	CRC 1 (16)	$f(x_1)=1+x^5+x^{12}+x^{16}$	1.2	3.8	1.5	1.7	1.6
7	CRC 2 (32)	$f(x_2)=1+x+x^2+x^4+x^5+x^7+x^8+x^{10}+x^{11}+x^{12}+x^{16}+x^{22}+x^{23}+x^{25}+x^{32}$	23.9	3.5	0.033	0.3	27.8
8	Signature (05)	$f(x)=1+x^2+x^5$	2	1.0	0.4	0.7	2.0
9	M-sequence (08)	$f(x)=1+x^3+x^5+x^6+x^8$	1.4	3.0	0.8	1.0	2.1

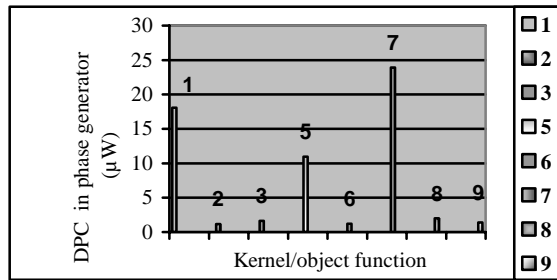


Figure 2. DPC in phase generator.

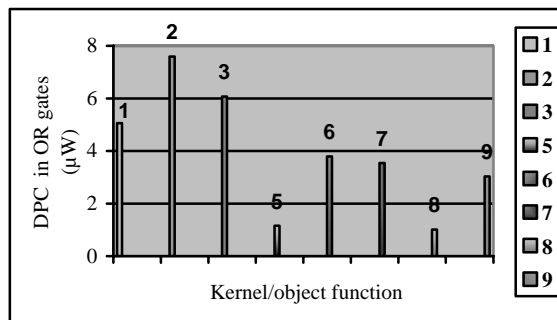


Figure 3. DPC in OR gates.

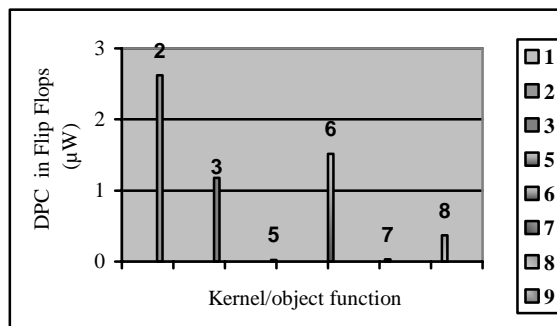


Figure 4. DPC in flip-flops.

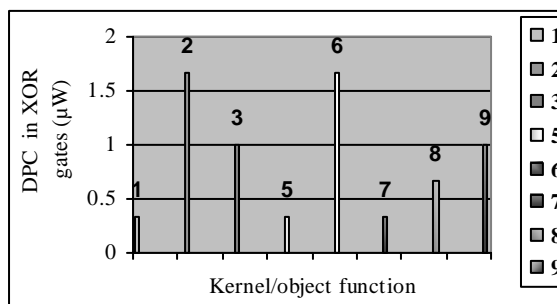


Figure 5. DPC in XOR gates.

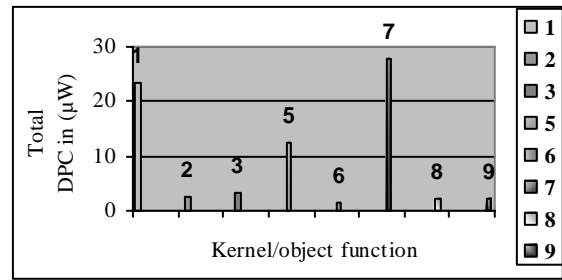


Figure 6. Total DPC.

### V. INSTRUCTION SET ARCHITECTURE (ISA) OF LFSR KERNEL

The optimum data path length in case of LFSR architecture is obtained by computing several LFSR steps at the same point of time as described in [8]. Computational priority based scalable architecture of LFSR with dynamically varying supply voltage to satisfy throughput requirement can lead to optimum dynamic power consumption [6]. The proposed ISA is motivated by ISA investigated for base-band processor for software defined radio application in [1]. The optimum data path length in case of LFSR kernel/object is suggested by performing computations of the several LFSR steps at the same instance over the polynomial connections. This approach reduces operand fetches; instruction fetches, and store outputs. As a result of this dominant part of DPC in case of global memory accesses reduces drastically.

The circular buffer method [14] proposed earlier in case of processing of one modulus operation on ARM core, is adapted here as an alternative to the modulus operations. This reduces energy-cost of modulus computation by reducing the number of machine cycles from 47 to 3 for single modulus operation.

The kernel consists of programmable processing unit, local memory and micro-sequencer. Table III gives instructions in optimized ISA required in the kernel/object in producing output for n-order polynomial. Memory accesses required with use of global memory are estimated by reducing frequency of operations based upon lowest order of non-zero coefficient of polynomial to produce multi-bit output for particular object [11].

#### A. Steps to reduce cost of computations in the ISA of LFSR kernel/object

The highest common computation among LFSR kernel/object functions is the modulus operation. The

Instructions	Functional specification
Output of LFSR sequence	For I = 1 to N { LFSR output array [n++] <= Seed Register [LFSR polynomial length - 1]}
Repeated sum with circular buffer size adjust	For I = 1 to LFSR polynomial length do { repeated addition to obtain output if polynomial connection exist; if result is >= 2, adjust circular buffer size to 2 }
Shift seed register in place	For LFSR polynomial length to I = 1 { Seed Register[k] <= Seed Register[k-1] and Seed Register[0] <= one bit output result }
Modulo-2 add M-sequences	{ Gold code[i] <= M-sequence1[i] (modulo-2 add) M-sequence2 [i] }

modulus operation is computationally intensive and takes large number of machine cycles. The energy-cost of computations is an important aspect of the kernel [16]. The fundamental computation in the kernel is modulus which is slow and must be avoided to reduce number of machine cycles.

TABLE III  
INSTRUCTIONS REQUIRED IN THE KERNEL/OBJECT TO  
PRODUCE OUTPUT FOR N-ORDER POLYNOMIAL

Following example illustrates the circular buffer size adjust method used to replace modulus operation [14]. The same has been adapted in proposed optimized ISA of Table III.

Let us assume circular buffer of size B bytes and a position indicated by a buffer A. Then to advance the offset by C bytes it can be written as

$$A = (A + C) \% B ;$$

This can be more *efficiently* written as

$$A += C;$$

If (A >= B)

{

$$A -= B;$$

}

Here it is assumed that C < B; which can be easily arranged. The efficient version reduces 47 machine cycles as compare to its first version [14] for single modulus operation. The LFSR kernel/object is modified to reduce machine cycles in computationally intensive modulus operation by modifying part of the code for circular buffer size adjust. The LFSR kernel/object implements following code to replace modulus operation

If (A >= 2)

{

$$A -= 2;$$

}

The kernel is capable of executing task from start to finish. This avoids frequent access to main memory and confines processing at local level to the kernel.

### B. Memory accesses in LFSR kernel/object

The instruction and operand fetches from global memory are estimated in comparison to the case when using the kernel with local memory and programmable processing unit. Access to distributed scratchpad/local memory [2] helps in reducing energy requirements. Local memory and operational unit of kernel/object process intensive data processing algorithms without frequent accesses to the main memory. The less frequent access to the main memory and local programmable operational unit reduces energy consumption. To improve data locality in respect of local memory, instruction and iteration level of the main loop are adjusted in the kernel [15]. The required polynomial coefficients are accessed into the local memory and redundant computations are

avoided for n-order polynomial of kernel/object function as given below

$$f(x) = 1 + a_1x + a_2x^2 + \dots + a_nx^n ;$$

if  $a_i=1$ , connection exist and

if  $a_i=0$ , connection does not exist

The polynomial coefficients once fetched into local memory are used for producing all the output bits over the LFSR polynomial length. This improves use of local memory performance and reduces main memory fetches to great extent thus benevolent energy efficiency [15].

The local memory fetch behavior as compare to global memory for operand fetches, instruction fetches, and in-between outputs is estimated by considering n-order polynomial of multi-bit output. The optimum data path length in case of LFSR kernel/object polynomial is obtained in producing output at the same time as described in [8]. Table IV gives computed number of operand fetches, instruction fetches, and in-between outputs from the kernel with the use of programmable processing unit, micro-sequencer and local memory.

Table V gives what would have been the required number of opcode fetches, instruction fetches, and in-between outputs to memory with the use of General Purpose Processor (GPP) and global memory access.

In Table V operand fetches, instruction fetches, and in-between outputs estimation for multi-bit output is obtained by reducing frequency of operations of particular object-polynomial based upon the lowest order of non-zero coefficient of that object-polynomial.

TABLE IV  
REQUIRED NUMBER OF OPCODE FETCHES,  
INSTRUCTION FETCHES, AND LOAD OUTPUTS WITH THE  
USE OF KERNEL

Kernel/ object Function	Operand fetch	Instruction fetch	In- operation outputs	Instan- tiation
PN long (42)	93	--	--	01
PNI short (15)	39	--	--	01
PNQ short (15)	39	--	--	01
Gold code (25)	58	--	--	01
Gold code (25)	58	--	--	01
CRC 1 (16)	40	--	--	01
CRC 2 (32)	72	--	--	01
Signature (05)	18	--	--	01
M- sequence (08)	24	--	--	01

Comparison of Tables IV and V shows only a few operand fetches, instruction fetches, and in-between outputs are required when LFSR kernel/object is used.

Following are the advantages after adapting ISA of Table III for LFSR kernel

- (i) In producing multi-bit output, the multiplication can be avoided over entire polynomial length of seed register.
- (ii) Repeated sum computation is performed efficiently, whenever there exist a connection.
- (iii) Circular buffer adjust method is used to replace modulus operation [14].
- (iv) Number of machine cycles in modulus operation is reduced.
- (v) In-place shifting of shift register bestows high performance to the kernel.

TABLE V  
REQUIRED NUMBER OF OPCODE FETCHES, INSTRUCTION FETCHES, AND LOAD OUTPUTS WITH THE USE OF GPP AND GLOBAL MEMORY ACCESS

Kernel/object Function	Operand fetch	Instruction fetch	In-operation outputs	Instantiation
PN long (42)	$6.4 \times 10^{14}$	$1.3 \times 10^{15}$	$4.4 \times 10^{12}$	--
PNI short (15)	$0.3 \times 10^6$	$0.7 \times 10^6$	$1.1 \times 10^4$	--
PNQ short (15)	$0.6 \times 10^6$	$1.2 \times 10^6$	$1.1 \times 10^4$	--
Gold code (25)	$0.7 \times 10^9$	$1.6 \times 10^9$	$1.1 \times 10^7$	--
Gold code (25)	$1.9 \times 10^9$	$4.6 \times 10^9$	$3.4 \times 10^7$	--
CRC 1 (16)	$0.55 \times 10^6$	$1.3 \times 10^6$	$1.3 \times 10^4$	--
CRC 2 (32)	$4.5 \times 10^{11}$	$9.2 \times 10^{11}$	$4.3 \times 10^9$	--
Signature (05)	$2.6 \times 10^2$	$3.1 \times 10^2$	16	--
M-sequence (08)	$2.5 \times 10^3$	$0.5 \times 10^4$	85	--

- (vi) Use of LFSR kernel reduces operand fetches, instruction fetches, and in-between outputs drastically. This can be observed from Table IV in comparison to Table V.

## VI. CONCLUSIONS

The multi-bit output LFSR kernel is proposed for functional instantiation of generation of PN long sequence, PNI short sequence and PNQ short sequence, M-sequence, Gold code sequence, CRC sequence and Signature sequence in SDR link-applications, and other wireless applications.

The proposed LFSR kernel has local memory and programmable processing unit, which results in major reduction of the dynamic power consumption (DPC) in hardware for the LFSR kernel/object function.

The suggested ISA for the LFSR kernel accelerates the tasks at the instruction execution levels. It is found that the total number of memory accesses reduces to few in the kernel approach while maintaining the programmability for different functions requiring LFSR object operational units. This reduces operand fetches, instruction fetches, and in-between outputs to memory as

shown in the results of computations given in Table IV. As a result, this dominant part of DPC in case of global memory accesses reduces drastically.

It is found that optimum data path length is obtained in case of LFSR kernel/object function when the kernel/object computes several LFSR steps at the same instance for given polynomial in producing multi-bit output. The circular buffer adjust method used for implementation of modulus operations further reduces the machine cycles and DPC.

## REFERENCES

- [1] Dake Liu, Eric Tell, Anders Nilsson (J) and Ingemar Soderquist, "Fully flexible baseband DSP processors for future SDR/JTRS," Western European Armaments Organization (WEAO) CEPA2 Workshop, Brussels, Belgium, March 2005. Available: <http://www.da.isy.liu.se/pubs/dake/dake-cepa22005.pdf>.
- [2] Yuan Lin et al, "SODA: A Low-power Architecture For Software Radio," Proceedings of the 33rd International Symposium on Computer Architecture (ISCA'06), 2006.
- [3] Uma S Jha, "Efficient Multimodal Signal Processing Engine Ease Communication, Computing, and Multimedia convergence," Proceedings of IEEE, ICPWC-2005, New Delhi, pp 348-352, 23-25 Jan 2005.
- [4] Uma S Jha et al, "Novel Application-Specific Signal Processing Architectures for Wideband CDMA and TDMA Applications," Proceedings of Vehicular Technology Conference 2000, Tokyo, Japan, 15-18 May 2000.
- [5] Puczko M. and Yarmolik V.N., "Designing cryptographic key generators with low power consumption," Proceedings of the Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA'06), Kuala Lumpur, 17-19 Jan 2006.
- [6] James Goodman and Anantha P. Chandrakasan, "Low power scalable encryption for wireless systems," Springer Journal of Wireless Networks, Vol. 4, No.4. pp55-70, 1998.
- [7] E. Aleksejev, A. Jutman, and R. Ubar, "LFSR Polynomial and Seed Selection Using Genetic Algorithm," Proceedings of Baltic Electronics Conference, pp 1-4, 2006.
- [8] C'edric Lauradoux, "From Hardware to Software Synthesis of Linear Feedback Shift Registers," Proceedings of Parallel and Distributing Processing Symposium 2007, IPDPS 2007, pp 1-8, 2007.
- [9] Thomas D Burd and Robert W Brodersen, "Energy Efficient Microprocessor Design," Kluwer academic publishers, 2002.
- [10] Abdullah Mamun and Rajendra Katti, "A new parallel architecture for low power linear feedback shift register," Proceedings of IEEE, ISCAS-2004, pp 333-336, 2004.
- [11] Rajendra S. Katti, Xiaoyu Ruan, and Hareesh Khattri, "Multiple-Output Low-Power Linear Feedback Shift Register Design," IEEE transactions on circuits and systems -I, Vol. 53, No. 7, pp 1487-1495, July 2006.
- [12] William C. Y. Lee, "Wireless and Cellular Telecommunications," McGraw Hill, Third international edition, 2006.
- [13] ALTERA, Application Note, 295, "Gold Code Generator Reference Design," Ver. 1.0, March 2003, Available:[www.cs.cmu.edu/~sensing-sensors/readings/Gold\\_Code\\_Generator-an295.pdf](http://www.cs.cmu.edu/~sensing-sensors/readings/Gold_Code_Generator-an295.pdf).

- [14] Andrew N Sloss, Dominic Symes and Chris Wright, "ARM System Developer's Guide; Designing and optimizing system software," Morgan Kaufmann Publishers, 2005.
- [15] G. Esakkimuthu, H. S. Kim, M. Kandemir, N. Vijaykrishnan and J. Irwin, "Investigating Memory System Energy Behavior Using Software and Hardware Optimizations," VLSI Design, Vol.12, No.2, pp 151-165, 2001.
- [16] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols," IEEE transactions on mobile computing, Vol. 5, No. 2, pp 128-143, February 2006.

working as Professor and Head Dept. of Electronics and Telecommunication in Institute of Engineering and Technology, at this University. He has more than 20 research papers to his credit. He is senior member of IEEE, member of Computer Society of India, Indian Society of Technical Education. He is chair of IEEE MP Subsection of Bombay section. His teaching and research areas are Computer Networks, Telecommunication Networks, Computer Architecture, Digital Signal Processing, and Performance Evaluation of Computer Systems.



<sup>1</sup>C N Khairnar is currently serving in the Faculty of Communication Engineering, Military College of Telecommunication Engineering (MCTE), Mhow (M.P). He received his Bachelor of Engineering (Electronics), from Pune University and

Master of Technology (Electronics), from Visvesvaraya Regional Engineering College, Nagpur. He is Ph.D. in the field of Electronics and Telecommunication Engineering. He has about 14 years experience of academic institutions and industry. His areas of interest are Low-power mobile computing and Software Defined Radios.



<sup>2</sup>Raj Kamal is Professor of Computer Science and Head of School of Electronics, Devi Ahilya University, Indore. A PhD from the Indian Institute of Technology Delhi, he has about 40 years of experience in teaching and research. A prolific writer, he has authored 10 textbooks and published 95 research papers in journals and conferences of both international and national repute. His areas of interest are mobile computing,

embedded systems, computer architecture and organization, data structures, microcontrollers, and Internet and Web technologies.



<sup>3</sup>Dr. Sanjiv Tokekar received His Bachelor of Engineering, (Electronics), Master of Electronics and Ph.D, (Electronic Engineering) from Devi Ahilya University (formerly known as University of Indore), Indore, India in the years 1982, 85 and 96, respectively. He has more than 22 years of experience. Currently, he is