The Next Generation Challenge for Software Defined Radio

Mark Woh[1], Sangwon Seo[1], Hyunseok Lee[1], Yuan Lin[1], Scott Mahlke[1], Trevor Mudge[1], Chaitali Chakrabarti[2], and Krisztian Flautner[3]

[1] University of Michigan - Ann Arbor, Ann Arbor MI, USA
[2] Arizona State University, Tempe, AZ, USA
[3] ARM Ltd., Cambridge, UK

Abstract. Wireless communication for mobile terminals has been a high performance computing challenge. It requires almost super computer performance while consuming very little power. This requirement is being made even more challenging with the move to Fourth Generation (4G) wireless communication. It is projected that by 2010, 4G will be available with data rates from 100Mbps to 1Gbps. These data rates are orders of magnitude greater than current 3G technology and, consequently, will require orders of magnitude more computation power. Leading forerunners for this technology are protocols like 802.16e (mobile WiMAX) and 3GPP LTE.

This paper presents an analysis of the major algorithms that comprise these 4G technologies and describes their computational characteristics. We identify the major bottlenecks that need to be overcome in order to meet the requirements of this new technology. In particular, we show that technology scaling alone of current Software Defined Radio architectures will not be able to meet these requirements. Finally, we will discuss techniques that may make it possible to meet the power/performance requirements without giving up programmability.

1 Introduction

The Third Generation Wireless age (3G) has provided an increase in data rate to the user which allows them to experience more than just voice over the air. Fourth Generation (4G) wireless networks is aimed at increasing that data rate by an order of magnitude in order to allow for users to experience richer content and get true mobility, freeing themselves from the need for wires or WiFi networks. The International Telecommunications Union (ITU) released a recommendation ITU-R M.1645 which sets data rate goals for 4G. They proposed a maximum data rate of 100Mbps for high mobility situations and 1Gbps for stationary and low mobility situations like hot spots. These targets are being used by most research on 4G today. It is also envisioned that 4G will include earlier standards and their protocols, and that they will work harmoniously together. SDR solutions can help reduce the cost of systems, which are required to support such a wide range of existing wireless technologies.

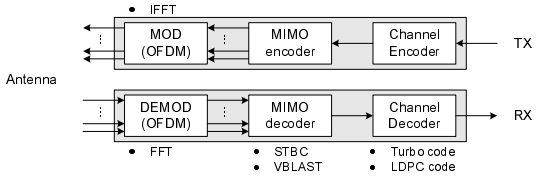


Fig. 1. The physical layer for a 4G terminal

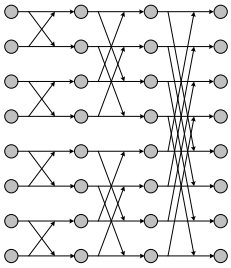
Previous papers have characterized the computational requirments of 3G [1]. There have been several proposals for SDR architectures capable of supporting 3G W-CDMA and 802.11 physical layers. Examples are Sandbridge's Sandblaster [2] and SODA [3]. But these architectures are not able to handle the almost 10-1000x increase in throughput required for 4G systems. This paper outlines the 4G physical layer. The aim is to show the requirements that are needed to process the new 4G physical layer and also to identify computational patterns that might suggest an architecture that can support 4G.

The 4G system we will study is based on orthogonal frequency division multiplexing (OFDM) that uses a 1024-point FFT/IFFT, a 4x4 16QAM multiple input multiple output (MIMO) antenna system, and a low density parity (LDPC) encoder and decoder. Detailed analysis of the major algorithms that make up these components and their computational characteristics show the following repeated computational pattern: load data from a memory element (initially this is the received data), permuting that data, performing one or two ALU operations, and storing the processed data back to memory. These patterns are similar to those found in 3G kernels. The architectures that are designed to support them, such as SODA, will not be able to meet the 4G requirements through technology scaling alone. As we will show, other techniques will have to be enlisted such as wider SIMD engines, special purpose functional units, and special memory systems.

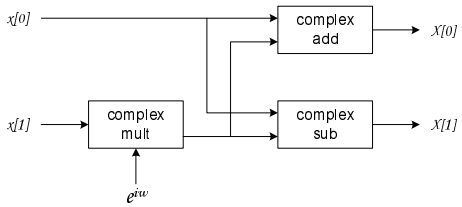
This paper is organized as follows. In the next section, we begin by presenting a simplified 4G system and by describing some of major kernels: an OFDM modulator/demodulator, a MIMO modulator/demodulator, and a channel decoder for LDPC. In section 3, we give a brief overview of SODA and use it as a baseline to identify the dominate workload profiles and common computational patterns of the kernels. In section 4, we present programmable hardware support for implementing these kernels efficiently to meet the high throughput required for 4G. The summary and concluding remarks are given in section 5.

2 4G Physical Layer

Figure 1 shows a 4G wireless terminal. Like other wireless communication system, its major blocks are a channel encoder/decoder and a modulator/demodulator. The role of the channel encoder is forward error correction that enables receivers



(a) Data movement patterns



(b) Computation patterns

Fig. 2. The data movement of an 8 point FFT and the computations in a 2 point FFT

to correct errors without retransmission. Modulation maps input data sequence onto signal waveforms which are specifically designed for the wireless channel. Demodulation estimates the transmitted data sequence from the received waveform, which have been corrupted by noise and interference when they traversed the wireless channel.

In order to satisfy the gigabit level throughput requirement, 4G systems employ three techniques not found together in 3G: 1) orthogonal frequency division multiple access (OFDMA); 2) MIMO to support multiple antennas; and 3) LDPC codes for the channel encoder/decoder.

2.1 OFDMA

OFDMA is a modulation scheme which transmits input signals over multiple narrow sub-channels. Both modulation and demodulation in OFDMA systems can be implemented with fast fourier transforms (FFT). Although additional synchronization procedures are required in OFDMA receivers, we can ignore them because their contribution is small.

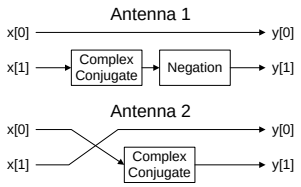
FFT As shown in Figure 1, the transmitter uses an inverse FFT (IFFT) for modulation and the receiver uses an FFT for demodulation. Because FFT and IFFT are almost identical, we will just analyze the FFT.

The FFT operation consists of a data movement followed by multiplication and addition on a complex number. If we assume an N point FFT, it consists of $\log_2 N$ stages. As an example, Figure 2(a) shows the data movement pattern of an 8 point FFT. It consists of 3 stages. Each stage shows a different but regular data movement pattern. The operation of each stage can be divided into several 2 point FFT operation as depicted in Figure 2(b).

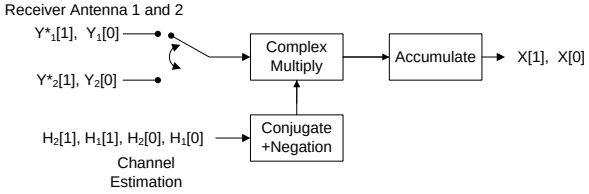
The FFT allows wide data level parallelism because all 2 point FFT operations required for proceeding from one stage to the next can be done in parallel. It is important to exploit this type of data level parallelism to meet power and performance requirements of 4G system, because the FFT width of 4G systems can be as large as 2048.

Time slot, T	Antenna	
	Tx 1	Tx 2
1	x_1	x_2
2	$-x_2^*$	x_1^*

(a) Transmission Matrix—the * indicates complex conjugate.



(b) Computation patterns of an STBC encoder



(c) Computation pattern of an STBC decoder

Fig. 3. Transmission code matrix and computation patterns of the Alamouti 2x2 STBC

2.2 MIMO

MIMO is a technique that uses multiple antennas both for the transmission and reception. It can be used for two purposes: signal quality enhancement by transmitting identical signal through multiple antennas and channel capacity enhancement by transmitting different signals on multiple antennas. Space time block codes (STBC) is a popular MIMO technique for the signal quality enhancement and the vertical Bell Laboratories layered space-time (V-BLAST) technique is popular for channel capacity enhancement.

STBC This is used to increase the signal quality by transmitting the same signal multiple times through different antennas. Signal quality is increased by receiving those redundant copies of the same signal and using the information from each receiver to optimally combine them to produce a better signal. The implementation we used is based on Alamouti's 2x2 scheme [4], which uses 2 transmit and 2 receive antennas.

STBC Encoder The encoder orders and transmits data based on the transmission matrix shown in figure 3(a). The operation consists of transmitting two different symbols at the first time instance, then transmitting the conjugate of the same two symbols with antennas switched (see the matrix in figure 3(a)). Figure 3(b) shows the computation needed to perform this operation. First the data is sent to each modulator and then the conjugate and negation are performed. This corresponds to a simple predication operation to obtain the real and imaginary

values. This is highly parallelizable, and a 1024 point FFT could be run in parallel on a 1024 wide SIMD (Single Instruction, Multiple Data) processor.

STBC Decoder The decoder takes the transmitted data from both time instances and combines them together to create the original two symbols. The decoder operation consists of performing complex multiplications between each of the received signals and the channel estimation for each antenna and then summing the values. Figure 3(c) shows this operation pattern. Calculating both symbols can be done at the same time with the least amount of data movement. Once again, because subcarriers are totally independent, this algorithm is highly data parallel, and a 1024 point FFT could be run in parallel on a 1024 wide SIMD.

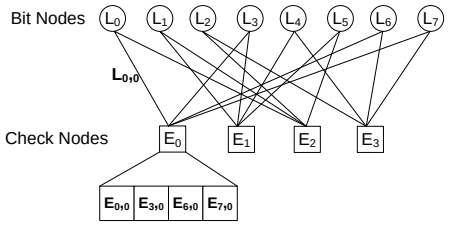
V-BLAST This is one of the spatial multiplexing schemes that improves multiplexing gain by transmitting independent data streams over different antennas. This technique combines multipath signals to obtains higher data rate compared to STBC. The V-BLAST algorithm that was used was based on work from [5] which reduces the computational complexity of V-BLAST.

V-BLAST encoder The V-BLAST encoder is similar to the STBC encoder. It also uses a transmission matrix to decide ordering, conjugating and negating for a block of data. Therefore, the pattern of required operations is: load the real and imaginary received data, permute the data based on the transmission matrix, then negate and store the result before sending it to the OFDM modulators associated with the multiple antennas. The computation pattern would be the same as figure 3(b) except the matrix for V-BLAST is 4x4.

V-BLAST decoder The decoding process of V-BLAST consists of two major steps: channel estimation and signal detection. The channel matrix is estimated based on pre-defined training symbols. The operations for channel estimation are relatively simple with shift and sign-change operations. Once the channel matrix has been estimated, the detection order is determined. The detection order is based on signal strength found among all the signals received. The strongest signal is selected and extracted from the received signal. This process is repeated for the remaining signals. This process is iterative and is referred to as successive interference cancelation. The signal detecting operations can be described by the following steps: 1) load the received signal; 2) vector multiplication for obtaining the stongest signal; 3) vector multiplication and subtraction for canceling the strongest signal; and 4) repeat.

2.3 Channel Encoder/Decoder

4G systems are expected to use both Turbo codes and LDPC codes as channel coding schemes. We limit our discussion to the characteristics of the LDPC codes in this section, because Turbo codes have already been used in 3G systems and their characteristics have been well documented elsewhere.



(a) Graphical representation of LDPC code

1. Initialization $E_{n,m} = 0, L_{n,m} = I_n$

2. Bit Node Operation $L_{n,m} = L_n - E_{n,m}$

3. Check Node Operation $E_{n,m}^{New} = -\prod_{n' \in N(m) \setminus \{n\}} sign(L_{n',m}) \cdot \min_{n' \in N(m) \setminus \{n\}} |L_{n',m}|$

4. Bit Update $L_n^{New} = L_{n,m} + E_{n,m}^{New}$

(b) LDPC decoding in 4 steps

Fig. 4. LDPC graphical representation and decoding operations

LDPC Figure 1 shows the channel encoder and decoder for LDPC. It is currently used in IEEE 802.16e and 802.11n. The encoder for LDPC is trivial in the sense that for each LDPC code there are a set of codewords available. For different data rates there are different number of codewords. In order to transmit data a codeword is picked and sent through the transmitter. Because the operation is fairly simple we will only discuss the LDPC decoding operation.

Decoding is based on an architecturally aware design for LDPC codes given in [6]. The code rates and the block sizes used were based on the IEEE 802.16e standard [7] and picked in order to meet the 100Mbps and 1Gbps target data rate.

The graphical representation of LDPC is shown in figure 4(a). The check nodes represents the number of rows in the parity check code and the bit nodes represent the number of columns. The edges connecting the check nodes and bit nodes are the 1's in the parity check code matrix—all other values are 0. The LDPC decoding operation is broken down into 4 stages as shown in figure 4(b). These four stages are the Initialization, Bit Node, Check Node, and Bit Update operation. This implementation is based on the Min-Sum algorithm.

The major operation in the implementation of LDPC is to first load the L_n and $E_{n,m}$ values. The next step is to permute the L_n's so they align with the $E_{n,m}$ values. Then it is possible to compute $L_{n,m}$ by performing an subtraction. Finally we do a compare and select to find the first and second minimum. This operation performs the Bit Node operation and the Check Node operation. The Bit Update operation first loads the L_n, then it does a comparison to determine whether the location of the minimum $E_{n,m}$ is the same as the L_n position. If it is not, then it will use the first minimum as the minimum $E_{n,m}$. Otherwise it will use the second minimum. Finally, it adds the new $E_{n,m}$ value to L_n , updating the L_n value. This operation is done for each block row of the code. After all block rows have been updated an iteration is complete.

LDPC exhibits considerable data level parallelism. For each $E_{n,m}$ we process one L_n at a time. Potentially we can do an N SIMD wide operation for the Bit Node and Check Node operation where N is the number of Check Nodes.

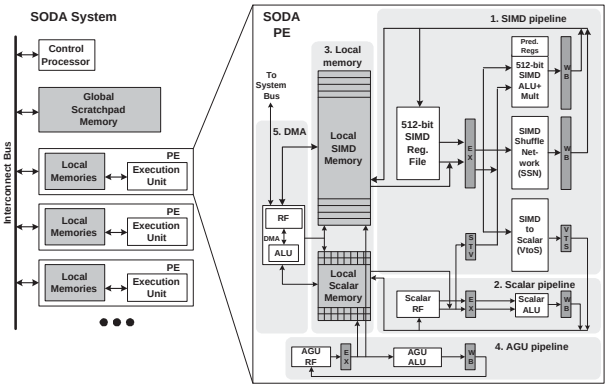


Fig. 5. SODA Architecture for SDR

3 Computational Analysis

3.1 Baseline Architecture

In order to calculate the workload characteristic we took an existing architecture for 3G and programmed the 4G algorithms onto it. The architecture we used is SODA. The SODA multiprocessor architecture is shown in Figure 5. It consists of multiple processing elements (PEs), a scalar control processor, and global scratchpad memory, all connected through a shared bus. Each SODA PE consists of 5 major components: 1) an SIMD pipeline for supporting vector operations; 2) a scalar pipeline for sequential operations; 3) two local scratchpad memories for the SIMD pipeline and the scalar pipeline; 4) an AGU (address generation unit) pipeline for providing the addresses for local memory access; and 5) a programmable DMA unit to transfer data between memories and interface with the outside system. The SIMD pipeline, scalar pipeline and the AGU pipeline execute in VLIW-styled lock-step, controlled with one program counter.

The SIMD pipeline consists of a 32-way 16-bit datapath, with 32 arithmetic units working in lock-step. It is designed to handle computationally intensive DSP algorithms. Each datapath includes a 2 read-port, 1 write-port 16 entry register file, and one 16-bit ALU with multiplier. The multiplier takes two execution cycles when running at the targeted 400MHZ. Intra-processor data movements are supported through the SSN (SIMD Shuffle Network). The SIMD pipeline can also take one of its source operands from the scalar pipeline. There are also several SIMD reduction operations that are supported, including vector summation, finding the minimum and the maximum.

3.2 Workload Profile

The breakdown of the major algorithms in our 4G protocol is listed in table 1. This analysis is based on the algorithms as they would be programmed for the

Algorithm Name	100Mbps Data Rate	1Gbps Data Rate
	MCycle/s	MCycle/s
FFT	360	360
IFFT	360	360
STBC	240	-
V-BLAST	-	1900
LDPC	7700	18500

Table 1. Cycle Count of Major 4G Kernels on SODA

SODA architecture. We calculated the number of cycles per second needed to support the data rate shown. Referring back to the system diagram in figure 1: for the 100Mbps rate we assume the STBC algorithm based on the Alamouti scheme which uses 2 transmit and 2 receive paths; and for the 1Gbps rate we assume a 4 transmitter and 4 receiver multiplexing diversity scheme based on V-BLAST. In the STBC algorithm we require that each receiver performs one FFT but only one STBC decoder for all the receivers. Each receiver is independent of the other's operation so both FFTs can run on separate processors. For the multiplexing diversity scheme each receiver processes separate data. That means that for the 1Gbps data rate we have 4 independent streams of 250Mbps being processed, but still only one V-BLAST decoder has to be performed.

From the table we can see that the channel coding algorithm is the dominate workload. Assuming we were processing each multiplexing diversity stream on one processor it would require us to run SODA at more than 10GHz for the 100Mbps case and almost 30Ghz for the 1Gbps case. An alternative approach would be to have one processor for each kernel. This would mean we would need the maximum frequency of SODA to be 8GHZ and 20Ghz for the 100Mbps and 1Gbps cases respectively. Though it may seem that the FFT, IFFT, STBC and V-BLAST algorithms are somewhat negligible compared to the channel coding we should not forget that the workload of channel coding is related to the data rate. As the data rate decreases the workload of the channel coding also decreases but the other kernels do not. At low data rates the other algorithms become comparable in cycle count and the optimization for these algorithms will then be key to an efficient design.

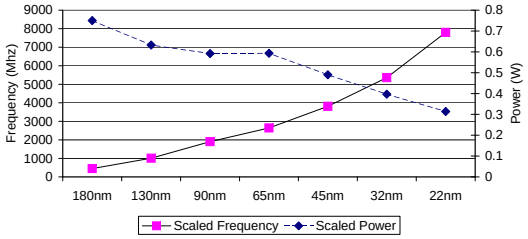
3.3 Computational Patterns

Analysis of each algorithm reveals that there is a consistent computational pattern. Table 2 shows each kernel's inner loop broken down into simpler operations. The pattern of loading the received data, permuting the data, performing an ALU operation, then a secondary ALU operation and finally storing the result back is very common to all the algorithms. These patterns make up the majority of the cycle time and are repeated for all the data being streamed in.

Another point to the note is that the data is streamed through the operations. Once the data is consumed we do not refer back to it until the next iteration,

Algorithm Name	Load	Permute	First ALU Op	Secondary Op	Store
FFT	X	X	X		X
IFFT	X	X	X		X
STBC	X	X	X	X	X
V-BLAST	X	X	X	X	X
LDPC	X	X	X	X	X

Table 2. Computational Pattern of 4G algorithms



(a) Technology Scaled SODA

Technology Node	180nm	130nm	90nm	65nm	45nm	32nm	22nm
Vdd (V)	1.8	1.3	1.1	1.1	1	0.9	0.8

(b) Vdd Voltage Scaling

Fig. 6. Technology scaling from 180nm to 22nm with respect to Frequency, Power, Vdd on SODA for 4G

or a summation, or a max/min is performed. Often sequences of operations are performed before having to store results. This suggests that there is little temporal locality of the data. Once the data is consumed we do not expect it to be used again. This is true for most DSP applications [8].

Data alignment is a key problem in each of the algorithms. Each algorithm has to align data before any computation can be performed. In the SODA architecture we use the SSN which includes a perfect shuffle network to perform this operation.

4 Architectural Implications

The frequency that the SODA processor would need to operate at in order to processes 4G was estimated at 20Ghz. Based on data from the ITRS roadmap [9] and [10] we show in figure 6(a) that technology scaling will still leave us a factor of 3x behind in frequency for a given power budget at 22nm. The power budget was set at $3W/mm^2$ combined for all cores. It is set by limitations of cooling and packaging based on data from ITRS. At 22nm this would be around 1W. Until recently technology scaling has also been accompanied by a scaling in supply voltage. As we get to smaller technology nodes this is no longer the case and the supply voltage is not scalling as much [11]. Figure 6(b) shows the decrease in supply voltage with technology node. The table shows that power

consumption will be decreasing more slowly and also that frequency scaling and voltage scaling will be less effective in terms of power reduction.

From the figure we see that at 22nm we could support the 100Mbps data rate on SODA and still meet the power requirement. The 100Mbps solution would require 2 SODA processors running at 10Ghz. If our projections are correct, this is a possible future solution, because the 22nm technology node is expected to be in production in 2011 [12] which coincides with when ITU expects 4G networks to be deployed. This still does not leave us with any solution for the 1Gbps data rate. However, there are many features of the algorithms which we can exploit architecturally to help us reach the goal of 1Gbps and still retain the flexibility of a programmable SDR processor.

Multi-Processor Most of the 4G algorithms can be divided onto multiple processors especially for FFT, and STBC, and even LDPC. The workload can be divided evenly among the processors. However, as we subdivide the algorithms across processes we get an increase in data communication. Although each stage of an algorithm is highly data parallel, stages requires data movement between different subcarriers in the FFT and between different check nodes in the LDPC. As we subdivide the algorithms, communication will increase, but, because the operations of each stage are streamed, we may be able to hide the latency of this communication under the computations itself. This would require an efficient routing and interconnect network and also scheduling that would be able to meet the constraints of data communication when multiple processors are used.

By dividing the workload across multiple processors we would be able to meet the frequency target for the 4G 1Gbps workload but we would still be 3x off the power budget. Multicore designs themselves cannot solve the problem of meeting the 4G requirement.

Wider SIMD Increasing the SIMD width of the processors takes advantage of the highly data parallel nature of the algorithms. Based on historical transistor growth, at the 22nm node we can expect to grow from a 32 wide SIMD to a 2048 wide SIMD machine. This assumes a fixed area constraint. This increase in width would allow us to reduce the cycle count to compute any size FFT as long as N is greater than or equal to the SIMD width. For FFT, the data movement can be accomplished by the SSN shuffle network.

For LDPC this increase in SIMD would also be beneficial because we can process more parity check nodes for LDPC at once. LDPC though would not gain the same data movement advantages as FFT, because it needs to align the check nodes and the bit nodes. However, this would not increase the amount of data movement dramatically.

STBC would also benefit, because it would be possible to process more subcarriers at one time. Because there is little data movement within the STBC we can expect gains equal to the increase in width.

Special Purpose Functional Units Currently in SODA the operations are RISC like in that after every instruction is simple and then writes back to the register file. This can be costly in terms of power and latency, because, as we stated earlier, the algorithms are streaming in nature. Writing back the data may not be very efficient. This suggests that functional units that chain operations will be beneficial not only in performance but also power. There has been work [13] that shows that using special functional units to streamline common operational patterns may not only increase performance but also will be more area and energy-efficient.

LDPC would also benefit from having special minimum and maximum registers embedded into the ALU. For each row operation of the parity check matrix that is performed the result will be compared with the current state of the register and swapped if the condition is met. In comparison with SODA, by implementing this special functional unit, LDPC can be reduce in cycle count by about 30 percent.

Memory System Most of the algorithms like LDPC, FFT and STBC all treat each row of the SIMD as independent. The data is loaded from memory then permuted and stored back. There is no instance in those algorithms where two rows have to access the same data at the same time. This suggests that the memory system does not have to be a large global shared memory. Instead it can be divided into banks. Banking the memory as much as possible will reduce the cost of reading and writing data into a large global memory. Banking will allow us to reduce the size of each memory, increase the speed, and lower power of the memory system. In algorithms like LDPC, which may need block sizes that are larger than currently used, we would be able to efficiently scale the size of the memories too.

Algorithms would also benefit from a smarter memory systems that support flexible gather/scatter accesses. Currently many cycles are wasted in LDPC aligning the check nodes and bit nodes. V-BLAST would also benefit, because the algorithm has to read and write back data in changing orders.

5 Conclusion

The power/performance requirements for 4G presents a significant challenge for computer architects, especially if some degree of programmability is to be retained. Currently technology is not capable of processing a 4G system on a single processor. In this paper we have analyzed a 4G system in the context of the SODA architecture and have shown that 3G solutions cannot meet the performance of 4G even if technology scaling is taken into account. We have presented architectural options that can improve the performance and reduce the power consumption of 4G solutions. We have argued that one solution to the power/performance challenge for 4G will increase the number of cores, and that each core will include a very wide SIMD processor with special purpose function units and highly banked memories.

References

1. Lee, H., Lin, Y., Harel, Y., Woh, M., Mahlke, S.A., Mudge, T.N., Flautner, K.: Software defined radio - a high performance embedded challenge. In Conte, T.M., Navarro, N., mei W. Hwu, W., Valero, M., Ungerer, T., eds.: HiPEAC. Volume 3793 of Lecture Notes in Computer Science., Springer (2005) 6–26
2. Schulte, M., Glossner, J., Jinturkar, S., Moudgill, M., Mamidi, S., Vassiliadis, S.: A low-power multithreaded processor for software defined radio. J. VLSI Signal Process. Syst. **43** (2006) 143–159
3. Lin, Y., Lee, H., Woh, M., Harel, Y., Mahlke, S.A., Mudge, T.N., Chakrabarti, C., Flautner, K.: Soda: A low-power architecture for software radio. In: ISCA, IEEE Computer Society (2006) 89–101
4. Alamouti, S.M.: A simple transmit diversity technique for wireless communications. IEEE J. on Select Areas in Communications **16** (1998) 1451–1458
5. Guo, Z., Nilsson, P.: A vlsi architecture of the square root algorithm for v-blast detection. J. VLSI Signal Process. Syst. **44** (2006) 219–230
6. Zhu, Y., Chakrabarti, C.: Architecture-aware ldpc code design for software defined radio. IEEE Workshop on Signal Processing Systems (2006)
7. : (http://www.ieee802.org/16/pubs/80216e.html)
8. Robelly, J.P., Seidel, H., Chen, K.C., Fettweis, G.: Energy efficiency vs. programmability trade-off: architectures and design principles. In: DATE '06: Proceedings of the conference on Design, automation and test in Europe, 3001 Leuven, Belgium, Belgium, European Design and Automation Association (2006) 587–592
9. : (http://public.itrs.net)
10. Rodriguez, S., Jacob, B.: Energy/power breakdown of pipelined nanometer caches (90nm/65nm/45nm/32nm). In: ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design, New York, NY, USA, ACM Press (2006) 25–30
11. McPherson, J.W.: Reliability challenges for 45nm and beyond. In: DAC '06: Proceedings of the 43rd annual conference on Design automation, New York, NY, USA, ACM Press (2006) 176–181
12. Chau, R., Doyle, B., Doczy, M., Datta, S., Hareland, S., Jin, B., Kavalieros, J., Metz, M.: Silicon nano-transistors and breaking the 10 nm physical gate length barrier. Device Research Conference (2003) 23–25
13. Karnik, T., Borkar, S., De, V.: Sub-90nm technologies: challenges and opportunities for cad. In: ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, New York, NY, USA, ACM Press (2002) 203–206