A GALS Many-Core Heterogeneous DSP Platform with Source-Synchronous On-Chip Interconnection Network

Anh T. Tran, Dean N. Truong, and Bevan M. Baas

Department of Electrical and Computer Engineering
University of California - Davis, USA
{*anhtr, hottruong, bbaas*}*@ucdavis.edu*

Abstract

This paper presents a many-core heterogeneous computational platform that employs a GALS compatible circuit-switched on-chip network. The platform targets streaming DSP and embedded applications that have a high degree of task-level parallelism among computational kernels. The test chip was fabricated in 65nm CMOS consisting of 164 simple small programmable cores, three dedicated-purpose accelerators and three shared memory modules. All processors are clocked by their own local oscillators and communication is achieved through a simple yet effective source-synchronous communication technique that allows each interconnection link between any two processors to sustain a peak throughput of one data word per cycle.

A complete 802.11a WLAN baseband receiver was implemented on this platform. It has a real-time throughput of 54 Mbps with all processors running at 594 MHz and 0.95 V, and consumes an average 174.76 mW with 12.18 mW (or 7.0%) dissipated by its interconnection links. We can fully utilize the benefit of the GALS architecture and by adjusting each processor's oscillator to run at a workload-based optimal clock frequency with the chip's dual supply voltages set at 0.95 V and 0.75 V, the receiver consumes only 123.18 mW, a 29.5% in power reduction. Measured results of its power consumption on the real chip come within the difference of only 2-5% compared with the estimated results showing our design to be highly reliable and efficient.

1. Introduction

Fabrication costs for state-of-the-art chips can now easily exceed several million dollars; and design costs associated with ever-changing standards and end user requirements are also extremely expensive. In this context, programmable and/or reconfigurable platforms that are not fixed to a single application or a small class of applications become increasingly attractive.

The power wall limits the performance improvement of conventional designs exploiting instruction-level parallelism that rely mainly on increasing clock rate with deeper pipelines. Many new techniques and architectures have been proposed in the literature; and multiple-core designs are the most promising approaches among them [1]. Recently, a large number of multi-core designs were found in both industry and academia [2–5]. Also, reconfigurable and programmable many-core designs for DSP and embedded applications are becoming popular research topics [6–8].

Transistor density and integration continue to scale with Moore's Law, and for practical digital designs, clock distribution becomes a critical part of the design process for any high performance chip [9]. Designing a global clock tree for a large chip becomes very complicated and it can consume a significant portion of the power budget, which can be up to 40% of the whole chip's power [10]. One particular method to address this issue is through the use of globally-asynchronous locally-synchronous (GALS) architectures where the chip is partitioned into multiple independent frequency domains. Each domain is clocked synchronously while inter-domain communication is achieved asynchronously [11]. GALS, therefore, becomes a top candidate for multi- and many-core chips that wish to do away with complex global clock distribution networks. In addition, GALS allows the possibility of fine-grained power reduction through frequency and voltage scaling [12].

The method of inter-domain communication is a crucial design point for GALS architectures. One technique is asynchronous clockless handshaking, which uses multiple phases of signal (i.e. *request/send/valid/ack*) exchange to transfer data. Due to the round-trip signal exchange, the transferring latency between two consecutive data words is high. Besides that, the asynchronous clockless circuits are difficult to verify in traditional CAD flows, and they also demand a comparatively large area and energy requirement [13, 14]. An alternative is source-synchronous clocking, commonly used in off-chip communication, whose design only requires a sender's clock signal to be sent with the sender's data to the receiver. For synchronization, a dual-clock FIFO at the receiver is used to buffer the data between two clock domains with the FIFO's write side clocked by the sender while its read side is clocked by the receiver. This method achieves high efficiency by obtaining a peak throughput of one data word per cycle with low area and power costs [15,16].

In this paper, we present the design of a GALS many-core computational platform utilizing a source-synchronous communication architecture. In order to evaluate the efficiency of this

platform and its interconnection network, we mapped a complete 802.11a WLAN baseband receiver on this platform. Actual chip measurement results are reported, analyzed, and compared against simulation. The outline of the paper is organized as follows. Section 2 explains our motivation for designing a GALS many-core heterogeneous DSP platform. The design of our computational platform is described in Section 3. Section 4 presents the architecture of our reconfigurable high-throughput GALS-compatible circuit-switched inter-processor communication network. The implementation and measurement results of the test chip are shown in Section 5. Mapping, analyzing and measuring the performance and power consumption of an 802.11a baseband receiver on this platform is discussed in Section 6. Finally, Section 7 concludes this paper.

2. Motivation

Our design is highly scalable and consists of a large array of small fine-grained cores plus dedicated-purpose accelerators, forming a GALS many-core heterogeneous platform that targets DSP, multimedia and embedded workloads motivated by the following key observations.

2.1. High Performance with Many-Core Design

Pollack's Rule states that performance increase of an architecture is roughly proportional to the square root of its complexity [12]. This rule implies that if we try to apply many sophisticated techniques to a single processor and make its logic area double, we only speedup its performance by around 40%. On the other hand, with the same area increase, a dual-core design using two identical cores could achieve a 2x improvement assuming that applications are 100% parallelizable. With the same argument, a design with many small cores should have more performance than one with few large cores for the same die area. However, performance increase is heavily hindered by Amdahl's Law, which implies that this speedup is strictly dependent on the application's inherent parallelism:

$$Speedup \approx \frac{1}{(1 - Parallel\%) + \frac{1}{N} \cdot Parallel\%} \qquad (1)$$

where N is number of cores.

Fortunately, for most applications in the DSP and embedded domain, a high degree of task-level parallelism can be easily exposed [6]. By partitioning the natural task-graph description of an embedded application, where each task can easily fit into one or a few small processors, the complete application will run much more efficiently. This is due to the elimination of unnecessary memory fetching and complex pipeline overheads. In addition, the tasks themselves run in tandem like coarse pipeline stages.

2.2. Power Savings through GALS Clocking Style

Since each core is in its own frequency domain, we are able to reduce the power dissipation and increase energy efficiency on a fine-grained level as illustrated in Fig. 1 in many ways:

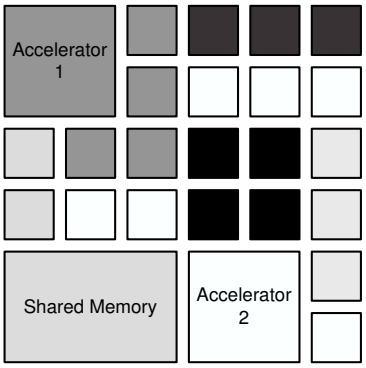

Figure 1. Illustration of a GALS many-core heterogeneous system consisting of many small identical processors, dedicated-purpose accelerators and share memories.

- GALS clocking design allows to utilize simple local ring oscillator for each core, and hence eliminates the need of complex and power hungry global clock trees [10].
- Unused cores can be effectively disconnected by power gating, and thus reducing leakage.
- When workloads distributed for cores are not identical, we can allocate different clock frequencies and supply voltages for these cores either statically or dynamically. This allows the total system to consume a lower power than if all active cores had been operating at a single frequency and supply voltage [17].
- We can reduce more power by architecture-driven methods such as parallelizing or pipelining a serial algorithm over multiple cores [18].
- We can also spread computationally intensive workloads around the chip to eliminate hot spots and balance temperature.

From the advantages on both performance and power consumption above, clearly, a many-core GALS design style is highly desirable for programmable/reconfigurable DSP platforms.

2.3. High Efficiency from Heterogeneous Architecture

For many tasks that have computationally intensive requirements such as error control coding/decoding, security encryption/decryption, FFT/IFFT, video motion estimation, etc., which do not map well to a set of fine-grained cores, one compromise is to build dedicated-purpose accelerators for tasks that are commonly found in many embedded, multimedia, and DSP applications. These accelerators are then integrated into the rest of the GALS array of identical processors to form a heterogeneous many-core platform. This approach is found in many designs such as high-speed SDR platforms [19, 20], and modern multi-core GPUs.

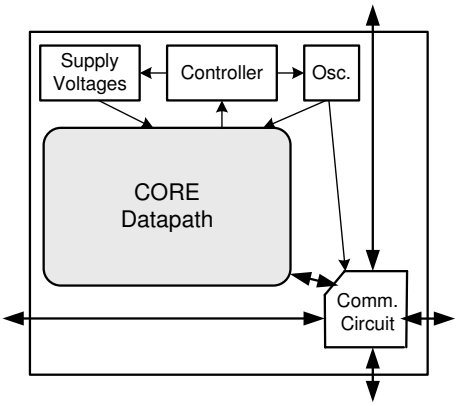

Figure 2. Common block diagram of processors or accelerators in our heterogeneous system. The main differing component among these processors is their computational cores.

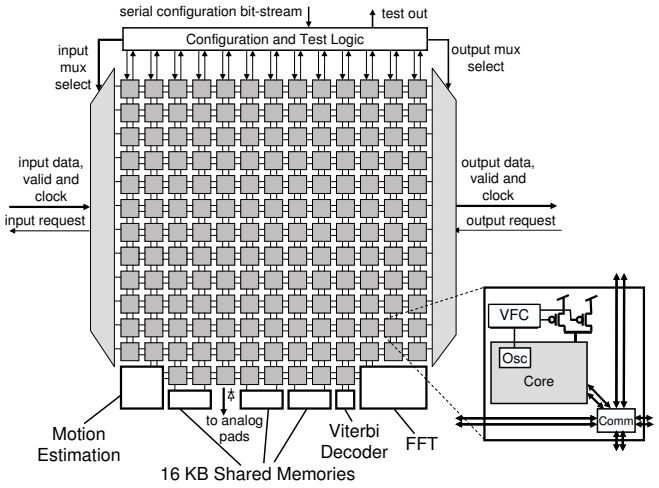

Figure 3. Block diagram of the 167-processor heterogeneous computational platform

3. Design of Our Programmable Heterogeneous GALS Many-core Platform

We implemented the many-core platform using a standard-cell design flow. Because of the array nature of the platform, the local oscillator, voltage switching, configuration and communication circuits are reused throughout the platform. These common components are designed as a generic "wrapper" which could then be reused to make any computational core compatible with the GALS array, and thus allowing easy design enhancements. The difference between the programmable processors and the accelerators is mainly in their computational datapaths as illustrated in Fig. 2.

The top level block diagram of our 167-processor computational platform is shown in Fig. 3. The platform consists of 164 small programmable processors, three accelerators (FFT, Viterbi decoder and Motion Estimation), and three big shared memory modules. All processors, accelerators and memory modules operate at their own clock frequency and share multiple global

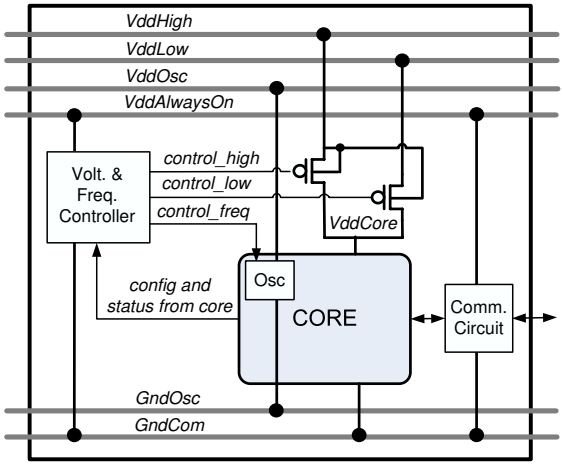

Figure 4. The Voltage and Frequency Controller (VFC) architecture

power supply voltages. Their clock frequency and supply voltage can be set statically or dynamically by their local voltage and frequency controllers (VFC).

In this section, we briefly present the design of the processors, accelerators, shared memory modules of the system, and the local VFC architecture. The communication network for these processors and accelerators will be discussed in Section 4.

3.1. Programmable Processors, Reconfigurable Accelerators and Shared Memories

Processors utilize an in-order single-issue six-stage pipeline with a 16-bit fixed-point ALU and a 40-bit MAC. Each processor has a local 128x35-bit instruction memory and a local 128x16-bit data memory. It supports more than 60 basic instructions.

The Viterbi and FFT accelerators, which are computationally intensive in high-speed communication systems, and thus are included in the platform. The Viterbi can decode convolution codes up to a constraint length 10 and the FFT is capable of performing 16- to 4096-point FFT/IFFT transformations. The platform also has a motion estimation processor typically used for video processing applications. Furthermore, the platform contains three 16-KB shared memory modules used for applications that require a shared set of data memory among different kernels and/or a local stored cache. The accelerators and memories are highly configurable depending on user requirements.

3.2. Per-Processor Clock Frequency and Supply Voltage Configuration

Each processor has its own ring-oscillator that can be configured to operate over a wide range of frequencies from 5 MHz to 1.7 GHz. At runtime, if the processor is idling the clock oscillator fully halts after 6 cycles; and it restarts immediately once work becomes available.

In order to achieve energy-efficiency, the processor should be supplied at the appropriate voltage level corresponding to its current operating frequency. This means we need to supply differ-

ent frequencies and voltages to different processors depending on their workloads. Because on-chip DC-DC converters have high design cost (complexity and area) and also large voltage switching delay [21], we use multiple global external supply voltages with hierarchical power grids that are simple and efficient [22, 23]. The core of each processor is configured to connect to one of two supply voltages V_{ddHigh} or V_{ddLow} or fully disconnected if unused. The benefits of having more than two supply voltages are small when compared to the increased area and complexity of the controller needed to effectively handle voltage switching [24].

The frequency and supply voltage of each processor are controlled dynamically or statically by its VFC as shown in Fig. 4. Dynamic control is based on the workload of processor that is derived from its input FIFO utilization status. Static configuration is intentionally set by the programmer for a specific application to optimize its performance and power consumption. This configuration is useful for many DSP applications that have static load behavior at runtime.

To avoid the effect of noise caused by voltage switching, the oscillator is powered by its own voltage and ground. The VFC and communication circuit also have their own voltage that is shared among all processors in the platform to guarantee the same voltage level for all interconnection links, thus only requiring level conversion to and from the processor core.

4. GALS Compatible Source-Synchronous Inter-Processor Communication Network

The communication circuit of each processor is also a part of the generic "wrapper" as mentioned previously (Fig. 2). This section describes the design of the communication network using source-synchronous interconnection technique across clock domains.

4.1. Reconfigurable High-Speed Circuit-Switched Interconnection Network

Figure 5 depicts the interface between any two neighboring processors in the platform. Each processor communicates with other processors through its two switches. Each switch has five ports: the Core port which is connected to its local core, and the North, South, West, and East ports which are connected to its four nearest processors' switches.

As shown in the figure, an input from the West port of one switch can be configured to go out to any port among the Core, North, South, East ports and vice versa. For simplicity, Fig. 5 only shows the full connections to and from the West port of one switch; all its other ports are connected in a similar fashion. Connections of these switches form two separate networks such that one processor can send data to any of the eight directions and can receive data from any two directions through its two input FIFOs.

The multiplexers of each switch are configured pre-runtime which fixes the communication link between any two processors. Thus, the circuit-switched link is guaranteed to be independent and never shared. So long as the destination processor's FIFO

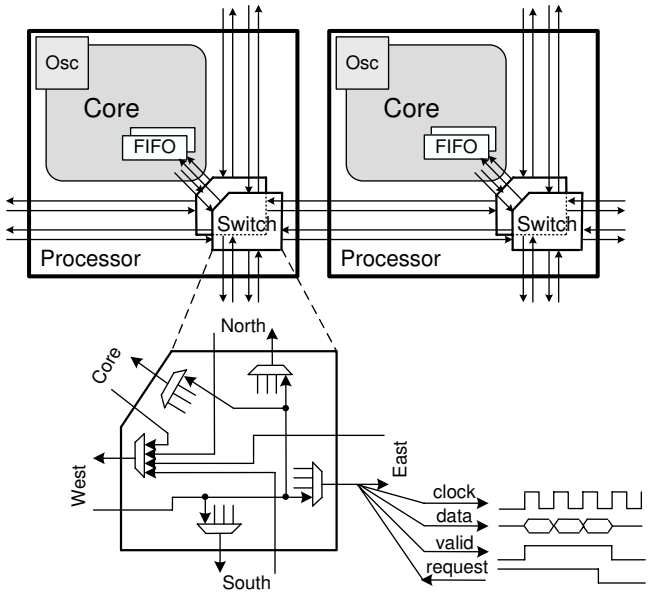

Figure 5. Interconnection network architecture with source-synchronous communication technique.

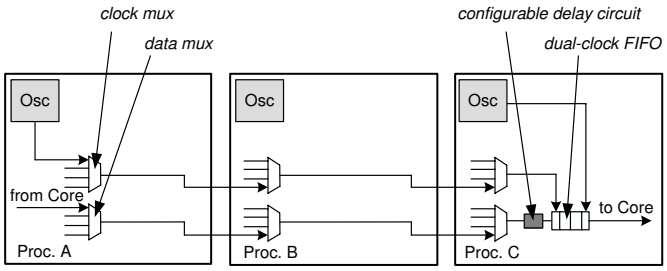

Figure 6. Example of a long-distance source-synchronous communication through one intermediate processor

is not full, a one data word per cycle throughput can be sustained. This compares favorably to a packet-switched network whose runtime network congestion can significantly degrade its communication performance [25, 26]. Our interconnection network's architecture is well suited for DSP applications that have high-speed interconnect requirements fixed at runtime.

4.2. Communication Reliability

Figure 6 shows an example of a communication link that is configured to connect two long-distance processors. This link passes through one intermediate processor, Proc. B, which is in between the source and destination processors, Proc. A and Proc. C. The figure also shows both clock and data being multiplexed by the circuit-switched architecture. The destination processor (Proc. C) uses a dual-clock FIFO to buffer the received data before processing. Its FIFO's write port is clocked by the source clock of Proc. A, while its read port is clocked by its own oscillator, and thus supports GALS communication [27].

Data and clock are sent by the source processor to the destination processor through a sequence of multiplexers of intermediate

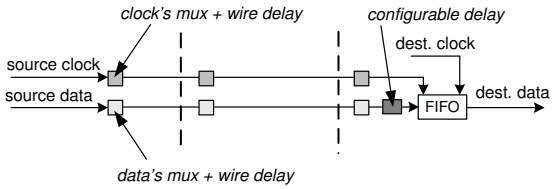

Figure 7. A simplified communication model derived from Fig. 6

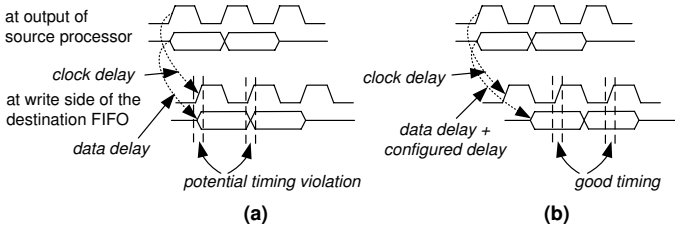

Figure 8. Timing illustration. a) Without using configurable delay circuit. b) With an appropriately configured delay

switches, and each data word is valid for one cycle. Fig. 7 is a simplified version of Fig. 6 focusing only on the impact of delay on source-synchronous timing. The dotted lines represents the boundary between two nearest processors. The total delay of wire and multiplexer in each processor's switch is depicted as a delay block. As shown, the clock and data signals are sent to the destination in the same way; thus, with a careful layout, their total delay can be nearly equal.

Because the delay of the clock and data is generally close, a timing violation can occur as illustrated by the waveform in Fig. 8(a). Also, the data bus can have mismatches due to variations and crosstalk, and the clock can have jitter causing unreliable communication in actual chip implementations.

Instead of leaving reliable communication up to chance, we purposefully add a delay circuit before the input FIFOs of each processor. This delay circuit is configurable in order for the clock rising edge to trigger within the safe timing window where the data is stable, as depicted in Fig. 8(b). This requires the delay value to be adjusted to satisfy the following timing constraint:

$$t_{hold} < D_{conf} + D_{data} - D_{clk} + t_{clk-q} < T - t_{setup} \qquad (2)$$

where, D_{conf}, D_{data} and D_{clk} are the configured, data and clock delays, respectively; T is the source clock cycle time.

The value of t_{setup}, t_{hold} and t_{clk-q} are mainly dependent on the standard cell design, technology process and fabrication variation; thus the value of D_{conf} can be different even for two connections with the same length and the same source clock frequency. The best value of D_{conf} for each link can be found through chip testing. The testing results confirm that all processors correctly communicate their data at 1.2 GHz (the maximum operating frequency of processors) when delay values are appropriately configured; that gives a peak throughput of 19.2 Gbps per 16-bit connection link.

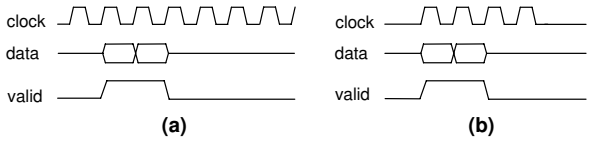

Figure 9. Source-synchronous communication methods. a) Source clock is alway active along the connection path. b) Only sending clock when having data. Two extra active cycles after the last valid data word to increase the communication robustness.

4.3. Low-Power Communication Method

Figure 9 illustrates two strategies of source-synchronous data communication from one processor to another. In the case depicted by Fig. 9(a), the clock is always active even without any accompanying data. The clock travels along the connection path from the source to destination and consumes some power for doing nothing while there is no data sent. Measurement results show that when sending clock alone without data, intermediate switches and the destination FIFO can dissipate 45% of the total power had it been sent with data (including the power dissipated by interconnect wires).

Our proposed method is shown in Fig. 9(b), where the source clock is only sent when there is data to transfer. However, if we aggressively send only one cycle of clock for each data word, the data can be lost if there is a large delay mismatch between clock and data links. Thus, we add two more cycles of clock after the last valid word sent. This method compromises between the aggressive and always active methods in order to maintain the high robustness at lower power.

Most importantly, the high energy-efficiency of our circuit-switched communication network is achieved due to its simple switch architecture, which does not buffer at the switch's input or output ports, and has no arbitration circuit, therefore no power is wasted for resolving runtime traffic congestion which is a significant portion of the power budget in dynamic packet-switched networks [28].

5. Test Chip Implementation and Measurement

5.1. Chip Implementation

The platform was fabricated in ST Microelectronics 65nm low-leakage CMOS process and its die micrograph is shown in Fig. 10. It has a total of 55 million transistors with an area of 39.4 mm^2. Each programmable processor occupies 0.17 mm^2, with its communication circuit occupying 7% including the two switches, wires and buffers. The area of the FFT, motion estimation and Viterbi decoder accelerators is six times, four times and one time, respectively that of one processor; the memory module is two times the size of one processor.

5.2. Measurement

At 1.3 V, the programmable processors can operate up to 1.2 GHz. The configurable FFT, Viterbi, motion estimation pro-

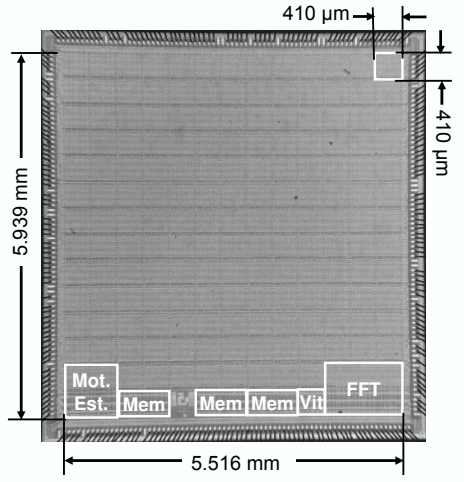

Figure 10. **Die micrograph of the test chip.**

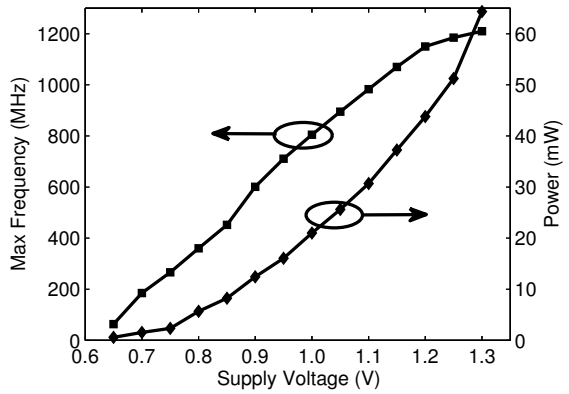

Figure 11. **Maximum frequency and 100% active power dissipation of one programmable processor over various supply voltages**

cessors, and memory modules can run up to 866 MHz, 894 MHz, 938 MHz and 1.3 GHz, respectively.

The maximum frequency and power consumption of the programmable processors versus supply voltage is shown in Fig. 11. As shown in the figure, they have a nearly linear and quadratic dependence on the supply voltage, respectively. These characteristics are used to reduce power consumption of an application by appropriately choosing the clock frequency and supply voltage for each processor as detailed in Section 6.

Figure 12 shows the measured leakage power of processors over various supply voltages. As shown, this leakage power is exponentially dependent on supply voltage and is negligible which can be ignored when compared with the dynamic power in a real application.

Table 1 shows the average power dissipation of processor, accelerators and communication circuit at 0.95 V and 594 MHz. This supply voltage and clock frequency is used to evaluate and test the 802.11a baseband receiver application described in the next section. The FFT is configured to perform 64-point transformations, and the Viterbi is configured to decode 1/2-rate convolution codes.

Also shown in the table, during stalls (i.e. non-operation

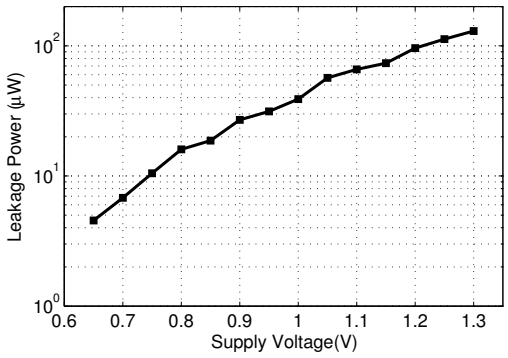

Figure 12. **Leakage power of one programmable processor over various supply voltages**

Table 1. **Average power consumption measured at 0.95 V and 594 MHz.**

Operation of	100% Active (mW)	Stall (mW)	Standby (mW)
Processor	17.6	8.7	0.031
FFT	12.7	7.3	0.329
Viterbi	6.2	4.1	0.153
FIFO Write	1.9	0.7	~0
Switch	1.1	0.5	~0

while the clock is active) the processors and communication circuits (including wires) also consume significant portions, approximately 35-55%, of their normal operating power. Leakage power are very small while processors are in the standby mode with clock halting.

6. Application Mapping: a Case Study

In order to relatively evaluate the performance and energy-efficiency of the platform, we mapped and tested a real 802.11a baseband receiver. Some steps to reduce its power consumption while keeping the real-time throughput requirement are also presented.

6.1. Mapping a Complete 802.11a Baseband Receiver

The receiver is complete including all necessary features of a practical one such as frame detection and timing synchronization, carrier frequency offset (CFO) estimation and compensation, and channel estimation and equalization. It consists of 23 processors plus the FFT and Viterbi accelerators as shown in Fig. 13. In this implementation, the CFO compensation uses a lookup table to compute the complex unit vector of the accumulated offset angle, and then uses a complex multiplication for sample rotation instead of using CORDIC algorithm as reported in our previous published paper [29] (all other processors are unchanged).

Processors are programmed using our simple C language version combined with assembly code for configuration of interconnect links and also for optimization. The compiled code of the whole receiver is simulated on the Verilog RTL model of our platform using Cadence NCVerilog and its results are compared with

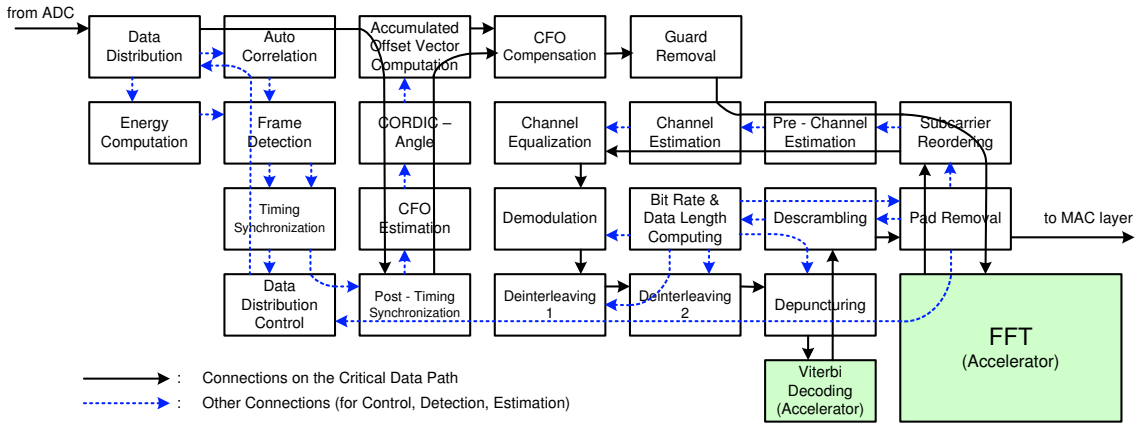

Figure 13. Mapping of a complete 802.11a baseband receiver on the many-core computational platform

a Matlab model to guarantee its accuracy. By using the activity profile of the processors reported by the simulator, we evaluate its throughput and power consumption before testing it on the real chip. This implementation methodology reduces debugging time and allows us to easily find the optimal operation point of each task.

6.2. Receiver Critical Data Path

The dark solid lines in Fig. 13 show the connections between processors that are on the critical data path of the receiver. The operation and execution time of these processors determine the throughput of the receiver. Other processors in the receiver are only briefly active for detection, synchronization (of frame) or estimation (of the carrier frequency offset and channel); then they are forced to stop as soon as they finish their job[1]. Consequently, these non-critical processors only add latency to the system and do not affect the overall data throughput[2] [29].

6.3. Performance Evaluation

Figure. 14 shows the overall activity of the critical path processors. In this figure, the Viterbi accelerator is shown to be the system bottleneck. It is always executing and forces other processors on the critical path to stall while waiting either on its output to send data or on its input to receive data[3]. Therefore, the total execution time and waiting time of each processor equals to the total execution time of the Viterbi accelerator (2376 cycles) during the processing of a 4-µs OFDM symbol. In essence, all OFDM symbols are processed by a sequence of processors on the critical path in a way that is similar to a pipeline (with 4 µs per stage per 2376 cycles). Therefore, the receiver can obtain a real-time 54 Mbps throughput when all processors operate at the same clock frequency of 594 MHz. According to measured data,

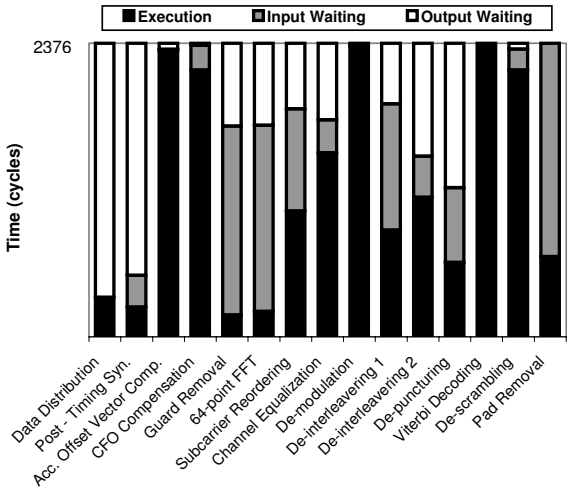

Figure 14. The overall activity of processors for processing a 4 µsec-OFDM symbol in the 54 Mbps mode

in order for all processors operate correctly they must be supplied at the lowest voltage level of 0.95 V.

6.4. Power Consumption Estimation

The overall activity of processors allows us to reasonably estimate the average power consumption of the receiver. Based on the analysis results done with simulation and estimation steps, we configure the processors accordingly when running on the test chip.

6.4.1. Power Consumption on the Critical Path

Power consumption of the receiver is primarily by processors on the critical path because all non-critical processors have stopped when the receiver is processing data OFDM symbols. In this time, the leakage power dissipated by these ten non-critical processors is 0.31 mW (10×0.031). The total power dissipated by the critical path processors is estimated by:

$$P_{Total} = \sum P_{Exe.i} + \sum P_{Stall.i} + \sum P_{Standby.i} + \sum P_{Comm.i} \quad (3)$$

[1] Processors stop working after six cycles if their input FIFOs are empty.

[2] These non-critical processors will be woken up to detect and synchronize new frame after the current frame is completely processed. The control information is provided by the Pad Removal processor.

[3] This assumes that the input is always available from the ADC and the MAC layer is always ready to accept outputs.

Table 2. Operation of processors for processing one OFDM symbol in the 54 Mbps mode, and their power consumptions

Processor	Execution Time (cycles)	Stall with Active Clock (cycles)	Standby with Halted Clock (cycles)	Output Time (cycles)	Comm. Distance (# switches)	Execution Power (mW)	Stall Power (mW)	Standby Power (mW)	Comm. Power (mW)	Total Power (mW)
Data Distribution	320	960	1096	80×2	6	2.37	3.56	0.01	1.14	7.08
Post-Timing Sync.	240	960	1176	80×2	5	1.78	3.56	0.01	1.00	6.34
Acc. Off. Vec. Comp.	2320	56	0	80×2	2	17.19	0.21	0	0.53	17.93
CFO Compensation	2160	216	0	80×2	2	16.00	0.80	0	0.53	17.33
Guard Removal	176	768	1432	64×2	6	1.30	2.84	0.01	0.92	5.07
64-point FFT	205	768	1403	64×2	3	1.10	2.36	0.20	0.55	4.21
Subcarrier Reorder.	1018	576	782	48×2	4	7.62	2.13	0.01	0.51	10.27
Channel Equal.	1488	576	312	48×2	2	11.02	2.13	0.01	0.31	13.47
De-modulation	2352	24	0	288	2	17.42	0.09	0	0.96	18.47
De-interleav. 1	864	1512	0	288	2	6.40	5.60	0	0.96	12.96
De-interleav. 2	1130	1246	0	288	2	8.37	4.62	0	0.96	13.95
De-puncturing	576	1800	0	432	2	4.27	6.67	0	1.44	12.38
Viterbi Decoding	2376	0	0	216	3	6.20	0	0	0.93	7.13
De-scrambling	2160	216	0	216	2	16.00	0.80	0	0.72	17.52
Pad Removal	648	1296	432	216	2	4.80	4.80	0.01	0.72	10.33
Ten non-critical Proc.s								*0.31*		*0.31*
Total						**121.84**	**40.17**	**0.57**	**12.18**	**174.76**

$\times$ 2: *2 words (real and imaginary) of each sample or subcarrier*

where $P_{Exe.i}$, $P_{Stall.i}$, $P_{Standby.i}$ and $P_{Comm.i}$ are the power consumed by computational execution, stalling, standby and communication activities of the i^{th} processor, respectively, and are estimated as follows:

$$\begin{aligned} P_{Exe.i} &= \alpha_i \cdot P_{ExeAvg} \\ P_{Stall.i} &= \beta_i \cdot P_{StallAvg} \\ P_{Standby.i} &= (1-\alpha_i-\beta_i) \cdot P_{StandbyAvg} \end{aligned} \tag{4}$$

here P_{ExeAvg}, $P_{StallAvg}$ and $P_{StandbyAvg}$ are average power of processors while 100% execution, stalling or in standby (leakage only); α_i, β_i and $(1-\alpha_i-\beta_i)$ are percentages of execution, stall and standby activities of processor i, respectively.

For the worst case communication power, a processor will send its output words discretely, thus each data word is sent along with three cycles of clock as described in Section 4.3. Therefore, the communication power of processor i is estimated by

$$\begin{aligned} P_{Comm.i} = \gamma_i \cdot [&(P_{SwitchActive} + 2P_{SwitchStall}) \cdot L_i + \\ &(P_{FIFOWriteActive} + 2P_{FIFOWriteStall})] \end{aligned} \tag{5}$$

where L_i is communication length of its output link counted by the number of switches that it passes through; γ_i is its communication activity percentage. $P_{SwitchActive}$, $P_{SwitchStall}$ and $P_{FIFOWriteActive}$, $P_{FIFOWriteStall}$ are the average power consumed by one switch and one FIFO write, respectively with and without data sent while the clock is active.

While measuring the chip with all processors running at 0.95 V and 594 MHz the values of P_{ExeAvg}, $P_{StallAvg}$, $P_{StandbyAvg}$, $P_{SwitchActive}$, $P_{SwitchStall}$, $P_{FIFOWriteActive}$ and $P_{FIFOWriteStall}$ are shown in Table 1. For the i^{th} processor, its α_i, β_i and $(1-\alpha_i-\beta_i)$ values are derived from Column 2, 3 and 4 of Table 2; and γ_i, L_i are derived from its Column 5, 6 with a note that each processor computes one data OFDM symbol in 2376 cycles.

The power consumed by execution, stalling, standby and communication activities of each processor are listed in Column 7, 8, 9 and 10; and their total is shown in Column 11. In total, the receiver consumes 174.76 mW with a negligible standby power due to leakage (only 0.57 mW including the ten non-critical processors). The power dissipated by communication of all processors is 12.18 mW, which is only 7% of the total power.

6.4.2. Power Reduction

The power dissipated by the stalling activity is 40.17 mW, which is 23% of the total power. This wasted power is caused by the fact that the clocks of processors are almost active while waiting for input or output as shown in Column 3 of Table 2. Clearly, we expect to reduce this stall time by making the processors busy executing as much as possible.

To do this, we need to reduce the clock frequency of processors which have low workloads. Recall that in order to keep the 54 Mbps throughput requirement, each processor has to finish its computation for one OFDM symbol in 4 µs, and therefore, the optimal frequency of each processor is computed as follows:

$$f_{Opt.i} = \frac{N_{Exe.i} \quad (cycles)}{4 \quad (\mu s)} \qquad (MHz) \tag{6}$$

where, $N_{Exe.i}$ is number of execution cycles of processor i for processing one OFDM symbol, which is listed in Column 2 of Table 2. From this, the optimal frequencies of processors are shown in Column 2 of Table 3.

By running at these optimal frequencies, the power wasted by stalling and standby activities of the critical processors is eliminated while their execution and communication activity percentages increase proportionally to the decrease of their frequencies. Therefore, total power is now 134.32 mW as listed in Column 3 of Table 3, a reduction of 23% when compared with the previous case[4].

Now that processors run at different frequencies, they can be supplied at different voltages as shown in Fig. 11. Since power consumption at a fixed frequency is quadratically dependent on

[4]Ten non-critical processors still dissipate the same leakage power of 31 mW.

Table 3. Power consumption while processors running at optimal frequencies when: a) Both V_{ddLow} and V_{ddHigh} are set at 0.95 V; b) V_{ddLow} is set at 0.75 V and V_{ddHigh} is set at 0.95 V

Processor	Optimal Frequency (MHz)	Power Consump. (mW)	Optimal Voltage (V)	Power Consump. (mW)
Data Distribution	80	3.52	0.75	2.63
Post-Timing Sync.	60	2.78	0.75	2.11
Acc. Off. Vec. Comp.	580	17.72	0.95	17.72
CFO Compensation	540	16.53	0.95	16.53
Guard Removal	44	2.23	0.75	1.73
64-point FFT	51	1.64	0.75	1.23
Subcarrier Reorder.	257	8.12	0.75	5.22
Channel Equal.	372	11.34	0.95	11.34
De-modulation	588	18.38	0.95	18.38
De-interleav. 1	216	7.36	0.75	4.95
De-interleav. 2	283	9.34	0.95	9.34
De-puncturing	144	5.70	0.75	4.10
Viterbi Decoding	594	7.13	0.95	7.13
De-scrambling	540	16.72	0.95	16.72
Pad Removal	162	5.52	0.75	3.71
Ten non-critical Proc.s		*0.31*	*0.95*	*0.31*
Total (mW)		**134.32**		**123.18**

supply voltage, more power can be reduced due to voltage scaling. Because our platform supports two global supply voltage grids, V_{ddHigh} and V_{ddLow}, we can choose one of these voltages to power each processor depending on its frequency[5].

Since the slowest processor (Viterbi) is always running at 594 MHz to meet the real-time 54 Mbps throughput, V_{ddHigh} must be set at 0.95 V. If V_{ddLow} is set to equal to V_{ddHigh}, the power consumption does not change. If V_{ddLow} is lowered to where its supported maximum frequency is smaller than the optimal frequencies of all processors, then in order to correctly operate, all processors must be set to V_{ddHigh}. In this case, power consumption is also not improved.

To find the optimal V_{ddLow} we changed its value from 0.95 V (i.e V_{ddHigh}) down to 0.6 V where its maximum frequency begins to be smaller than the lowest optimal frequency among processors. The total power consumption corresponding to these V_{ddLow} values (while processors are set appropriately) is shown in Fig. 15. As shown in the figure, the optimal V_{ddLow} value is 0.75 V with total power of 123.18 mW as detailed in Column 5 of Table 3.

Notice that the power reduction comes from the effect of voltage scaling on the processor's execution activity. The communication circuits use their own supply voltage which is always set at 0.95 V, so they still consume the same 12.19 mW, which now is approximately 10% of the total power.

6.5. Measurement Result

We tested and measured this receiver on the real chip with the same configuration modes of clock frequency and supply voltage as used in the previous estimation steps. In all configuration modes, the receiver operates correctly and shows the same computational results as with simulation. The power measurement results are shown in Table 4. When all processors run at 0.95 V and 594 MHz, they consume a total of 177.96 mW that is a 1.8% difference from the estimated result. When all processors run at their optimal frequencies with the same 0.95 V supply voltage, they consume 139.64 mW; and when they are appropriately set at 0.75 V or 0.95 V as listed in Column 4 of Table 3, they consumes 129.82 mW. In these configurations, the differences between the measured and estimated results are only 3.9% and 5.1%, respectively.

These differences are small that show that our design methodology is highly robust. Our simulation platform allows programmers to map, simulate and debug applications correctly before running on the real chip reducing a large portion of application development time. For instance, we mapped and tested this complex 802.11a receiver in just two months plus one week for finding the optimal configuration compared to tens of months if implemented on ASIC which includes fabrication, test and measurement.

[5]Non-critical processors are always set to run at V_{ddHigh} and 594 MHz for minimizing the detection and synchronization time.

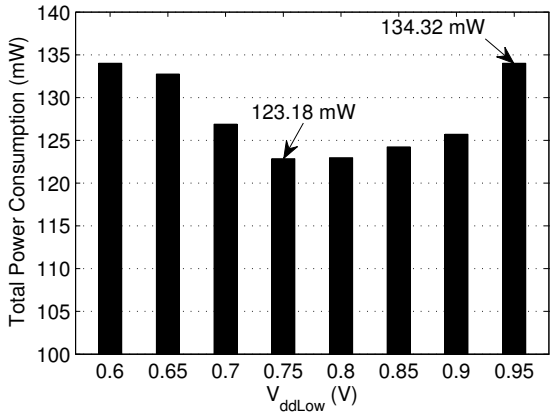

Figure 15. The total power consumption over various values of V_{ddLow} (with V_{ddHigh} is fixed at 0.95 V) while processors running at their optimal frequencies. Each processor is set at one of these two voltages depending on its frequency.

Table 4. Estimation and measurement results of the receiver at different configuration modes

Configuration Mode	Estimated Power (mW)	Measured Power (mW)	Difference
At 594 MHz and 0.95 V	174.76	177.96	1.8%
At optimal frequencies only	134.32	139.64	3.9%
At both optimal freq. & volt.	123.18	129.82	5.1%

7. Conclusion

A high-performance and energy-efficient programmable DSP platform consisting of many simple cores and dedicated-purpose accelerators has been presented. Its inter-processor communication network utilizes a novel source-synchronous interconnection technique allowing efficient communication among processors which are in different clock domains.

The on-chip network is circuit-switched and is configured before runtime such that interconnection links can achieve their ideal throughput at a very low power and area cost. For a real 802.11a baseband receiver with 54 Mbps data throughput mapped on this platform, its interconnection links only dissipate around 10% of the total power. We simulated this receiver with NCVerilog and also tested it on the real chip; the small difference between estimation and measurement results confirms the robustness of our design.

Acknowledgments

The authors thank Zhiyi Yu who inspired the original idea on source-synchronous clocking technique for many-core design. This work was supported by ST Microelectronics, IntellaSys, a VEF Fellowship, SRC GRC Grant 1598 and CSR Grant 1659, UC Micro, NSF Grant 0430090 and CAREER Award 0546907, Intel, and S Machines.

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, SF, CA USA, 2007.

[2] U. G. Nawathe, M. Hassan, and L. Warriner, "An 8-core 64-thread 64b power-efficient SPARC SoC," in *Intl. Conference on Solid-State Circuits (ISSCC)*, Feb. 2007, pp. 108–109.

[3] D. C. Pham, T. Aipperspach, et al., "Overview of the architecture, circuit design, and physical implementation of a first-generation Cell processor," *IEEE JSSC*, vol. 41, no. 1, pp. 179–196, Jan. 2006.

[4] V. Yalala, D. Brasili, and D. Carlson, "A 16-core RISC microprocessor with network extensions," in *Intl. Conference on Solid-State Circuits (ISSCC)*, Feb. 2006, pp. 78–79.

[5] M. B. Taylor, J. Kim, et al., "The RAW microprocessor: A computational fabric for software circuits and general purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Feb. 2002.

[6] B. Baas, Z. Yu, et al., "AsAP: A fine-grained many-core platform for DSP applications," *IEEE Micro*, vol. 27, no. 2, pp. 34–45, Mar. 2007.

[7] S. Vangal, J.Howard, and D. Carlson, "An 80-tile 1.28 TFLOPS networks-on-chip in 65nm CMOS," in *Intl. Conference on Solid-State Circuits (ISSCC)*, Feb. 2007, pp. 98–99.

[8] S. Bell, B. Edwards, et al., "TILE64 processor: A 64-core SoC with mesh interconnect," in *Intl. Conference on Solid-State Circuits (ISSCC)*, Feb. 2008, pp. 88–89.

[9] N. A. Kurd, J. S. Barkatullah, et al., "A multigigahertz clocking scheme for the Pentium® 4 microprocessor," in *IEEE JSSC*, Nov. 2001, pp. 1647–1653.

[10] V. Tiwari, D. Singh, et al., "Reducing power in high-performance microprocessors," in *ACM/IEEE Design Automation Conference (DAC)*, June 1998, pp. 732–737.

[11] M. Krstić, E. Grass, et al., "Globally asynchronous, locally synchronous circuits: Overview and outlook," *IEEE Design and Test of Computers*, vol. 24, no. 5, pp. 430–441, Sept. 2007.

[12] S. Borkar, "Thousand core chips: a technology perspective," in *ACM/IEEE Design Automation Conference (DAC)*, June 2007, pp. 746–749.

[13] B. R. Quinton, M. R. Greenstreet, and S. J.E. Wilton, "Asynchronous ic interconnect network design and implementation using a standard ASIC flow," in *IEEE Intl. Conference of Computer Design (ICCD)*, Oct. 2005, pp. 267–274.

[14] E. Beigné and P. Vivet, "Design of on-chip and off-chip interfaces for a GALS NoC architecture," in *IEEE Intl. Symposium on Asynchronous Circuits and Systems (ASYNC)*, Mar. 2006.

[15] Z. Yu and B. M. Baas, "Implementing tile-based chip multiprocessors with GALS clocking styles," in *IEEE Intl. Conference of Computer Design (ICCD)*, Oct. 2006, pp. 174–179.

[16] Y. Hoskote, S. Vangal, et al., "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sept. 2007.

[17] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Intl. Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2007, pp. 38–43.

[18] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low power CMOS digital design," *IEEE JSSC*, vol. 27, pp. 473–484, 1992.

[19] M. Woh, Y. Lin, et al., "From SODA to Scotch: The evolution of a wireless baseband processor," in *IEEE/ACM Intl. Symposium on Microarchitecture (MICRO)*, Nov. 2008, pp. 152–163.

[20] M. Shirasaki, Y. Miyazaki, et al., "A 45nm single-chip application-and-baseband processor using an intermittent operation technique," in *Intl. Conference on Solid-State Circuits (ISSCC)*, Feb. 2009, pp. 156–157.

[21] W. Kim, M. S. Gupta, et al., "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Intl. Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2008, pp. 123–134.

[22] E. Beigné, F. Clermidy, et al., "Dynamic voltage and frequency scaling architecture for units integration within a GALS NoC," in *IEEE Intl. Symposium on Networks-on-Chip (NOCS)*, Apr. 2008, pp. 129–138.

[23] D. Truong, W. Cheng, et al., "A 167-processor 65 nm computational platform with per-processor dynamic supply voltage and dynamic clock frequency scaling," in *Symposium on VLSI Circuits*, June 2008, p. C3.1.

[24] K. Agarwal and K. Nowka, "Dynamic power management by combination of dual static supply voltage," in *Intl. Symposium on Quality Electronic Design (ISQED)*, Mar. 2007, pp. 85–92.

[25] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Intl. Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 255–266.

[26] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Intl. Symposium on Computer Architecture (ISCA)*, Mar. 2004, p. 188.

[27] R. Apperson, Z. Yu, et al., "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE TVLSI*, vol. 15, no. 10, pp. 1125–1134, Oct. 2007.

[28] A. Kumar, L. Peh, et al., "Towards ideal on-chip communication using express virtual channels," *IEEE Micro*, vol. 2, pp. 80–90, Feb. 2008.

[29] A. T. Tran, D. N. Truong, and B. M. Baas, "A complete real-time 802.11a baseband receiver implemented on an array of programmable processors," in *Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Oct. 2008, pp. MA5–6.