An ultra-low energy asynchronous processor for Wireless Sensor Networks

L. Necchi*, L. Lavagno*, D. Pandini† and L. Vanzago†

*Politecnico di Torino
Torino, Italy

†STMicroelectronics
Agrate (MI), Italy

***Abstract*— This paper describes the design flow used for an asynchronous 8-bit processor implementing the Atmel AVR instruction set architecture. The goal is to show dramatic reductions in power and energy with respect to the synchronous case, while retaining essentially a traditional design flow. The processor was implemented in a 130nm technology using desynchronization, starting from an initial design downloaded from OpenCores.org. It consumes 14 pJ per instruction to deliver 170 MIPS at 1.2 V, and 2.7 pJ per instruction to deliver 48 MIPS at 0.54 V. It thus dramatically improves the energy consumed per instruction with respect to previous results from the literature.**

***Index Terms*— Wireless sensor networks, low-power, low-energy, low EMI, desynchronization, AVR CPU.**

I. Introduction

Wireless sensor networks (WSNs) promise to offer a variety of low-cost measurement and actuation services for a very broad range of applications, from home and factory automation, to health care, to environment and building monitoring. Most WSN architectures rely on a multitude of small devices spread over the covered area, in order to provide services in an unobtrusive, reliable and economical manner. Hence node cost, due to fabrication, deployment and maintenance, is a key aspect of WSN proliferation. Reducing both power and energy requirements per computation activity (e.g. sensing operation, packet reception or transmission, routing action) permits to reduce the size of the battery and increase its lifetime, thus impacting both unit and deployment costs.

While a majority of current demo applications for WSNs spend most of the node power operating the radio, it is clear that requiring more and more services from WSNs in the future will mean increased power consumption for the digital portion of the node as well. Examples of these power-hungry services include network security [16], in-network aggregation [22] and distributed data processing [20]. While adoption of these services in real WSN deployments is still very preliminary, they are generally recognized as very significant in the future, in order to improve WSN robustness and global network lifetime.

This paper describes an implementation of a typical WSN processor, the 8-bit Atmel AVR, using asynchronous techniques. The resulting processor is called YUPPIE, which stands for Yet another Ultra-low Power asynchronous Processor for wIreless sensor nEtworks. The *key contribution* of this exercise is to show that it is indeed possible to *simultaneously achieve*:

- design re-use, since the synthesizable VHDL model of the original microcontroller was downloaded from the `OpenCores.org` web site.
- implementation with a standard design flow and library, based on a 130 nm technology from STMicroelectronics.
- extremely short design time, since the complete design and simulation was completed in about four months of work by a person with no previous exposure to asynchronous design techniques, and limited exposure to synchronous ones. Most of the time was actually spent learning to use the tools from the industrial partner's design flow.
- aggressive power management by dynamic voltage scaling, which in an asynchronous context naturally implies frequency scaling as well.
- virtually zero wake-up time, when returning from sleep mode, and frequency change recovery time, In synchronous circuits both these require from hundreds to thousands of clock cycles, due to the long settling time of Phase Locked Loops (e.g. the Atmel AVR requires 2 ms at 4MHz).
- minimization of energy consumption, by working with a voltage supply very close to the process threshold.
- reduction of (estimated) electro-magnetic emission, with a promise of beneficial effects for the analog part of the circuit as well.

Our work uses the desynchronization approach [2], because it dramatically eases the transition from a synchronous to an asynchronous design style. This was a key requirement from the industrial partner of the project, STMicroelectronics.

It is well known that asynchronous circuits reduce electromagnetic emission with respect to equivalent synchronous ones [15], [11], because they reduce the power consumption peaks in the vicinity of clock edges. Hence they produce a flatter power spectrum and exhibit smaller voltage supply drops.

It is also well known that asynchronous circuits can be reliably operated at very low voltages, when device characteristics exhibit second and third order effects, hence when synchronous operation becomes very problematic [13]. This is of course especially true of quasi-delay insensitive or delay-insensitive circuits, since the datapath signals its completion reliably, no matter how slowly gates and wires switch. However, our experiments with a fabricated circuit [5] showed that even cheaper bundled-data circuits can be operated at

Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)
0-7695-2498-2/06 $20.00 © 2006 IEEE

voltages. This is due to the fact that the delay characteristics of different gates, operating at the same low voltage, vary in a reasonably similar manner.

Asynchronous circuits also offer the possibility of dramatic increases in performance, in the presence of extreme process and operating condition variability. This advantage, which was not a primary goal of this exercise due to the specific application domain, is permitted by the reduced margins that asynchronous design requires as compared to synchronous. In the synchronous case, all design, manufacturing and environment uncertainties must be handled by adding margins to the clock cycles. It is widely reported that these margins are currently in excess of 100% for a typical ASIC design flow and manufacturing process. This means that a device in which the operating frequency is not determined a priori, but chosen at run-time depending on both fabrication and operating environment effects, can work twice as fast as an equivalent synchronous one.

We are thus confident that desynchronization can be easily used to perform initial experiments in the domain of aggressive power, energy and EMI minimization. The hope, of course, is that more radical asynchronous design techniques, which of course offer better advantages along all these dimensions, will be embraced once the advantages of asynchronicity have been shown.

This paper describes in detail the design flow that has been used and the various choices that have been made. While our design case study covers one specific architecture, we believe that the techniques that we propose can be applied to any digital architecture for low-power low-energy battery-powered applications.

The desynchronization flow has the following characteristics:

- it accepts as input a synchronous design, described with a VHDL or Verilog RTL specification.
- it uses standard EDA tools to synthesize, map, place and route the design. Standard tools are also used for timing analysis, equivalence checking, local clock generation, extraction, scan insertion, automated test pattern generation and so on. In this experiment we used Synopsys Design Compiler and Cadence SOC Encounter for logic synthesis and physical design respectively, and Synopsys Nanosim for performance and power simulation.
- it fully automates the de-synchronization step [1], which removes the clock tree and replaces it with a network of asynchronous controllers.
- it generates a desynchronized netlist which outputs an asynchronous "clock", which can be used to drive synchronous peripherals (e.g. A/D converter interfaces, UARTs, etc.) whose performance and power requirements are not critical.

II. Previous work

A. Wireless Sensor Network platforms and applications

Over the past few years, Wireless Sensor Networks have gained more and more attention as a basic technology enabling the vision of Ambient Intelligence. Nowadays, the advances of silicon processing technologies permit the miniaturization of low-cost devices that, by integrating sensing, computational and communication capabilities, constitute the basic interface with the physical world. One of the most important constraints in Wireless Sensor Networks, mainly built with battery-powered nodes, is low power consumption, which has been driving research in the definition of both system and node architecture components [9]. Several device platforms, diversified in terms of storage, communication and computational capabilities, have been identified as a need for most Wireless Sensor Networks, corresponding to the layers of multi-tiered network architectures [8]. Berkeley Motes (i.e. the MICA Family) are a notable example of a general-sensing-class device with limited on-board capabilities. These motes, especially the MICA2 version, are extensively used by a huge number of research groups for Wireless Sensor Network prototyping in several typical ambient intelligence application scenarios. These include, for example, Environmental monitoring [12], Structural health monitoring [21], Personal health monitoring [19], and Tracking [18].

The most recently commercially developed version of the MICA family, i.e. the MICAZ, replaces the CC1000 radio used in MICA2 with a CC2420 radio that is compatible with the standard IEEE 802.15.4. This standard defines the physical and Medium Access Control layer specifications for low data rate wireless connectivity. It has been adopted by the Zigbee Industrial Consortium, aimed at promoting the usage of low data rate networks in WSN application scenarios such as industrial and home automation.

Currently, much of the research on WSN design at the system level is based on reducing the quantity of raw sensing data transmitted by the nodes, by increasing the amount of local computation performed on each node, in order to reduce the overall data traffic on the network. This increases the computing capability requirements for each node, and as a consequence it increases the importance of reducing the power consumption of its digital components.

Our design example is thus oriented towards the definition of an application-independent architecture, because it addresses a general purpose CPU design, namely a desynchronized version of the same processor used in the MICA family motes. We also claim that desynchronization can be profitably used also to derive asynchronous implementations of dedicated hardware unit for signal processing, encryption, and so on, in order to further reduce peripheral power and energy consumption.

B. Desynchronization

De-synchronization modifies the standard design flow for synchronous circuits [4] in three key steps:

1) *Conversion of the flip-flop-based synchronous circuit into a latch-based one (M and S latches in Figure 1(b)).* D-flip-flops are conceptually composed of master-slave latches. To perform de-synchronization, this internal structure is explicitly revealed (see Figure 1(b)) to decouple local clocks for master and slave latches (in a D-flip-flop they are both derived from the sam

Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)
0-7695-2498-2/06 $20.00 © 2006 IEEE

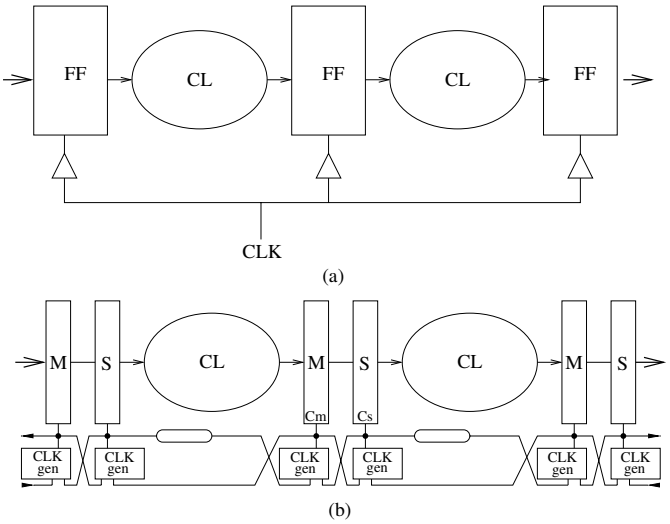

Fig. 1. (a) Synchronous circuit, (b) de-synchronized circuit.

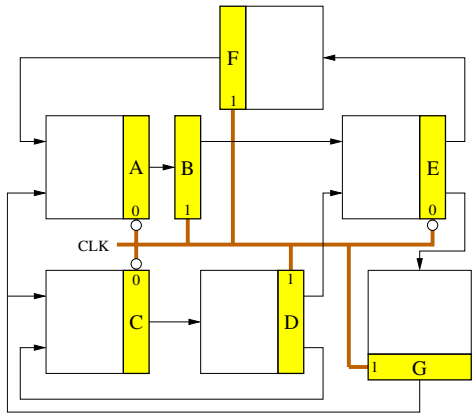

Fig. 2. A synchronous circuit with a single global clock.

so as to separate concerns between setup and hold time satisfaction. This step is essential to avoid the low clock skew constraints that are inherent in the flip-flop-based design style.

2) *Generation of matched delays for the combinational logic* (denoted by rounded rectangles in Figure 1(b)). Each matched delay must be greater than or equal to the delay of the critical path of the corresponding combinational block. Each matched delay serves as a completion detector for the corresponding combinational block.
3) *Implementation of the local controllers*. Each controller must open the latch after the following one has been closed (to avoid double latching or hold time violations) and close it after the previous one has been opened and the combinational logic has settled (to avoid setup time violations). Protocols and controller implementations are described more in detail in [2].

Figure 2 depicts a synchronous netlist after the conversion into latch-based design. The shadowed boxes represent latches, whereas the white boxes represent combinational logic. Latches must alternate their phases. Those with a label 0 (1) at the clock input represent the *even* (*odd*) latches. All latches are transparent when the control signal is high (CLK=0 for even and CLK=1 for odd). Data transfers must always occur from even (master) to odd (slave) latches and vice-versa. Usually, this latch-based scheme is implemented with two non-overlapping phases generated from the same clock.

The correctness of the desynchronization approach, i.e. the fact that the asynchronous circuit implements exactly the same I/O functionality as its synchronous counterpart has been proved in [2].

It can also be shown that a de-synchronized circuit can be interfaced with a synchronous one (the environment) in one of two ways:

1) Either by using its output handshake (which requires concentration of all interface signals in a single, potentially large, bundle) as the clock driving the synchronous circuit.
2) Or by driving its input handshake signals with the global clock and ignoring its output handshake signals.

Which approach is best essentially depends on which portion of the circuit is most performance critical. If the desynchronized portion is most critical (as in this case, since the AVR peripherals are quite simple), then approach (1) is best. On the other hand, if the circuit that has not been desynchronized is more performance critical, then approach (2) should be used.

Note that case (2) may become appealing even if originally the desynchronized portion was most performance critical, since desynchronization performs intrinsic automatic cycle-stealing, thus helping balance an initially unbalanced multi-stage synchronous design.

C. Dynamic voltage and frequency scaling

It is well-known that, even under normal operating conditions, the delay of a CMOS circuit scales linearly with its voltage supply, while its power scales quadratically. Thus the normalized energy-per-cycle or energy-per-operation efficiency measure scales linearly with voltage supply. However, it is very difficult to use this optimization opportunity to the extreme, by operating very close to the threshold voltage (as we do in this paper), because:

1) Library cells are seldom characterized by the manufacturer at such extreme operating conditions. Hence the normal ASIC design flow is unsuitable to guarantee correct operation under the lowest voltage, and hence energy and power, operating conditions.
2) The gate delay models deviate significantly from those used under nominal conditions, and make a straightforward scaling of performance and power impossible, or at least very risky.
3) The effects of various random or hard-to-predict phenomena, such as threshold voltage variations, wire width variations, local voltage supply variations due to IR drop, are significantly magnified.

All this means that, even if one were able to use the traditional design flow for circuits that will be operated at a voltage supply that is close to the transistor threshold voltage, the performance margins that one would have to use t

correct operation would be huge (they are already exceeding 100% under nominal conditions).

Two approaches have been proposed in the literature to tackle this problem with purely synchronous means. Both are based on sampling the output of a signal which is forced to make a transition very close to the clock cycle, and slow down the clock frequency or increase the voltage supply if this critical sampling happens at the current voltage and frequency conditions.

The Razor CPU [7] is designed with double slave latches and an ex-or in each master-slave pair (thus increasing by over 100% the area of each converted latch). The second slave is clocked half a clock cycle later than the first slave. In case the comparator detects a difference in values between the slaves, this means that the input changed very close to the falling edge of the clock of the first slave, and it memorized an incorrect value. The Razor in that case "skips a beat" and restarts the pipeline with the value of the second latch, which is always (assuming that environmental conditions change slowly) latched correctly. An external controller always keeps voltage and clock frequency very close to this "critical clocking" condition, in order to operate the processor very close to the best Vdd point for the required clock frequency, under the current temperature conditions.

The approach, while very appealing for processors, has an inherent problem that makes it not applicable to ASICs. Due to the near-critical clocking, it is always possible that the first latch goes meta-stable. In that case, the whole detector and the clock controller may suffer from meta-stability problems. That case is detected with analogue mechanisms, and the whole pipeline is flushed and restarted. This is easy to do in a processor, for which flushing and restarting is already part of a modern micro-architecture. It is very difficult, if not impossible, to achieve automatically for a generic logic circuit, such as those found in modern ASICs.

Another technique that has been proposed to dynamically monitor the delay of the logic at the current voltage value, and adjust the clock frequency accordingly, is the PowerWise^{TM} technology from National Semiconductors. It basically samples, with a high frequency clock, the output of a digital delay line that toggles once per system clock cycle. This is used, more or less as in Razor, to measure the delay of the line in the current environment conditions (temperature, Vdd etc.). The scheme is safer than Razor, because it allows one to insert enough synchronizers after the delay line to reduce the meta-stability danger. However, it is an indirect measure, and requires a complicated (patented) logic to monitor and control the clock speed.

Desynchronization achieves similar goals with much simpler logic, because the delay line output is directly used to generate the clock period. Moreover, several controllers can be interspersed with the logic, so that local conditions (e.g. temperature, Vdd, transistor and wire parameter variations) can be controlled much more closely, thus requiring potentially smaller margins.

Moreover, desynchronization reduces electro-magnetic emission (EMI) problems, because the clock for every cluster of latches, driven by a single controller, is out of phase with respect to the others. This spreads the power spectrum, reducing both irradiated and on-chip power noise. It also can improve performance, because it computes on-line the *useful skew* for each individual group of latches. The performance of the system is thus not given by the longest pipeline stage, as in the synchronous case, but by the maximum cycle mean[1]. This means that even unbalanced pipelines are *automatically balanced* at runtime by the handshaking mechanism.

D. Low-power asynchronous processors

Several asynchronous micro-processors reported in the literature exhibit very favorable power, energy and EMI characteristics with respect to their synchronous counterparts. For example, the Lutonium from CalTech [14] implements the 8051 ISA and reportedly achieves 500 pJ per instruction at 1.8 V and 200 MIPS, and 140 pJ per instruction at 0.9 V and 66 MIPS, with a 180 nm process.

According to [3], scaling CMOS technology to the next generation improves performance and reduces power consumption, thus reducing overall energy consumption by about 65% per generation. Based on this analysis, the Lutonium should achieve 175 pJ per instruction at full voltage supply and 49 pJ per instruction at minimum voltage supply, using a 130 nm process like our design.

The CalTech processors are based on Quasi-Delay-Insensitive (QDI) circuits, and hence are extremely robust with respect to all sorts of variations, including *data-dependent computation delays*.

Desynchronization uses a bundled-data completion detection mechanism, which results in smaller circuitry, but requires margins to ensure correct operation. Hence it cannot exploit as effectively the inherent characteristics of asynchronicity (low power, modularity, etc.), but it is much simpler to implement, by using a fairly standard design flow.

The Philips 8051 is another 8-bit micro-controller [17], which was synthesized starting from a Tangram specification. Unlike the Lutonium, the Philips 8051 used a bundled data datapath, thus resulting in (presumably) lower area overheads. Its main advantages over its synchronous counterpart, which led to its inclusion in some commercial products, were the low power consumption, but especially the low EMI, which dramatically reduced the cost of implementing the analogue RF portion of a pager. A second-generation implementation of the Philips 8051, commercialized by Handshake Solutions using a 180 nm process, consumes 150 pJ per instruction delivering 6.3 MIPS at 1.8 V. If implemented with a 130 nm process, this micro-controller should achieve the energy consumption of 53 pJ per instruction.

The Tangram flow is very similar in spirit to the desynchronization flow, because it uses mostly standard tools for physical design, design rule checking and so on. However, it requires one to use a new proprietary language, based on Communicating sequential Processes, for specification and high-level simulation. Moreover, synthesis and design for testability

[1]The maximum cycle mean is defined as the maximum over all cycles in the logic, which may include one or more latches, of the total cycle delay divided by the number of latches in the cycle.

tools are Tangram-specific. This is both an advantage, in that it improves the optimality of the result, and a disadvantage, because it requires extensive designer training and a partially new set of tools. In comparison, desynchronization is much easier to pick up and use by an experienced synchronous designer.

The SNAP/LE processor [10], [6] is an asynchronous low power 16-bit processor that was designed specifically for wireless sensor network applications. Its instruction set architecture was optimized specifically for WSN applications, because it operates in *event-driven* mode: basically, all code is executed as an interrupt service routine. It also features a set of instructions that directly control the various peripherals (timer, message handler) as co-processors, rather than by using the normal memory-mapped register interface. SNAP/LE, implemented in a 180 nm process like the Lutonium, requires 218 pJ per instruction to deliver 240 MIPS at 1.8 V and 23 pJ per instruction to deliver 28 MIPS at 0.6 V. It was implemented, like the Lutonium, using the QDI implementation style, which is a significant departure from traditional synchronous design. If implemented with a 130 nm process, this micro-controller should achieve the energy consumption of 76 pJ per instruction at full voltage supply and 8 pJ per instruction at minimum voltage supply.

Our processor, as will be described more in detail in Section IV, was implemented in a 130 nm process (one generation after the other processors described above). It requires 14 pJ per instruction to deliver 170 MIPS at 1.2 V, and 2.7 pJ per instruction to deliver 48 MIPS at 0.54 V. These results do not include program and data RAM access power. We estimated, from the data sheets of memory components for the 130 nm process that we used, that using 4KB of RAM for code and data, as could be required by a small WSN application, would increase energy consumption by about 20%.

III. The asynchronous AVR processor

A. The synchronous RTL design

The project started from an open-source implementation of an 8 bit microcontroller that is compatible with the Atmel AVR architecture, the "AVR_CORE" available from the `OpenCores.org` web site. We chose this processor because its Instruction Set Architecture is compatible with that of most WSN nodes of the MICA family.

Our aim was to demonstrate the possibility to re-design asynchronously an existing standard CPU core for a WSN node in a very short time and with reasonable costs. As mentioned above, we wanted to show reductions in terms of power consumption and noise emission, as well as potential increases in performance, while keeping compatibility with the AVR instruction set. This permits to re-use directly the binary software developed for this ISA.

The VHDL design from which we started provides a full implementation of the instruction set of the Atmel AVR microcontroller, and includes various extra component, such as data memory, program memory, timers, UART.

The design also includes a fully featured VHDL testbench that allows one to test the behaviour of the entire platform. This was very important, since it dramatically speeded up all phases of design, from debugging the implementation of the flow, all the way to measuring performance and power by simulation (we used Mentor Graphics Modelsim and Synopsys Nanosim for this purpose).

B. Logic synthesis

The first step, after making sure by simulation that the downloaded "AVR_CORE" indeed worked as advertised, was the logic synthesis of the initial synchronous design. For this step we used Synopsys Design Compiler. A short script was written in order to:

- load the VHDL core file,
- define performance and area constraints,
- synthesize the netlist,
- map it on the chosen industrial standard cell library,
- modify the project hierarchy to suit the needs of the desynchronization step,
- estimate critical path delay and
- save the resulting netlist on a Verilog file.

C. De-synchronization

The de-synchronization methodology [2] has been implemented, as described in [1], in the standard design flow of our industrial partner. The `desync` tool converts a technology-mapped synchronous netlist into an asynchronous one, performing the following steps:

- loading and parsing the Verilog input file.
- replacing registers with pairs of latches.
- automatically grouping the combinational logic into latch-separated islands,
- estimating, using Synopsys PrimeTime, the logic delay of each group,
- implementing the delay chains and the controllers,
- saving the de-synchronized output file in Verilog format.

In this case we chose to use only one logic group (i.e. one controller for the master and one for the slave latches), because AVR_CORE only has 4500 gates, organized as a 2-level pipeline (fetch and execute).

The total area of the desynchronized core is about 5% larger than that of the synchronous one. All our results do not include the effect of placement and routing, which are unlikely to be significant for such a small design.

D. Logic simulations of the asynchronous core

The de-synchronized version of AVR_CORE was then simulated, again using Modelsim, in order to verify the correct behaviour of the core. This required some minor modifications of the simulation settings and of the environment model. For example, we imported in Modelsim a new Standard Delay Format (SDF) file that contains the detailed gate-level delay information and library gate models.

Furthermore we modified the VHDL testbench to properly connect the handshake lines added by the de-synchronization tool, instead of the externally generated clock. We connected each peripheral with the falling edge of the "Requ

handshake signal of the desynchronized partition (we will call this signal "pseudo-clock"). After this, logical simulation was executed correctly.

E. Power estimations with Nanosim

Although Synopsys Design Compiler can perform power estimations for both synchronous and asynchronous circuits, using switching activity files (SAIF) derived from a simulator such as Modelsim, we cannot use it beyond the characterization corners of the standard cell library. Since we wanted to verify how power varies with very low supply voltages, which are usually not considered among these corner cases, we had to perform transistor-level simulations.

For this purpose we used Nanosim, an advanced transistor-level circuit simulation and analysis tool for analog, digital and mixed-signal design verification. Nanosim uses the same electrical model parameters that simulators such as Spice use, but it has more simplified transistor and wire models, that permit much faster simulation, with a minimal loss of accuracy.

The inputs of the synchronous and desynchronized circuit were driven using test vectors generated by Modelsim, and the output vectors produced by both simulators under nominal conditions were compared against each other, to check that indeed no errors had happened during the various synthesis, optimization and translation steps. The next section describes the results of such simulations.

IV. Simulation results and analysis

The simulation results that we obtained indeed show the expected behavior when lowering the voltage supply of both the synchronous and the asynchronous circuit. The delay of each circuit decreases with increasing Vdd as shown in Figure 3, while the power consumption increases with increasing Vdd, as shown in Figure 4. The energy per instruction, shown in Figure 5, increases almost linearly with the voltage supply. At the best power supply level, around 0.5 V, the desynchronized circuit is about 5 times more energy efficient than the synchronous one. Furthermore, the power consumption is reduced by more than one order of magnitude by scaling the voltage from the nominal level, 1.2 V, to about 0.5 V.

Of course, one could also scale the voltage supply for the synchronous circuit as well, and obtain similar power reductions, by using either the Razor or the PowerWise approaches mentioned above. However, this is much more difficult to do, as discussed in Section II-C, because it amounts to essentially measuring the timing of a delay line at a given supply voltage, and then tuning the clock to match the circuit performance. Moreover, Razor would require one to change drastically the AVR architecture, since it does not support pipeline squashing and restarting.

The desynchronized circuit offers the same advantages "for free". Moreover, it allows one to spread delay lines wherever they are needed to measure the effects of spatially correlated effects, for example temperature and wire thickness[2]. It is also

[2]Wire thickness, which affects both resistance and side capacitance, is spatially correlated due to the effects of the chemical-mechanical polishing step.

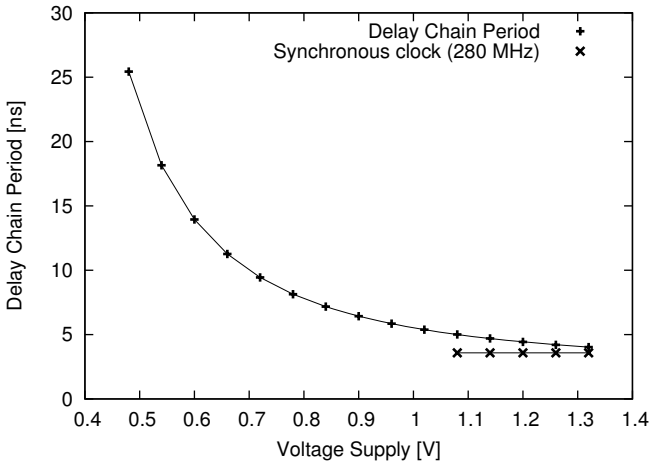

Fig. 3. Desynchronized processor cycle period versus supply voltage

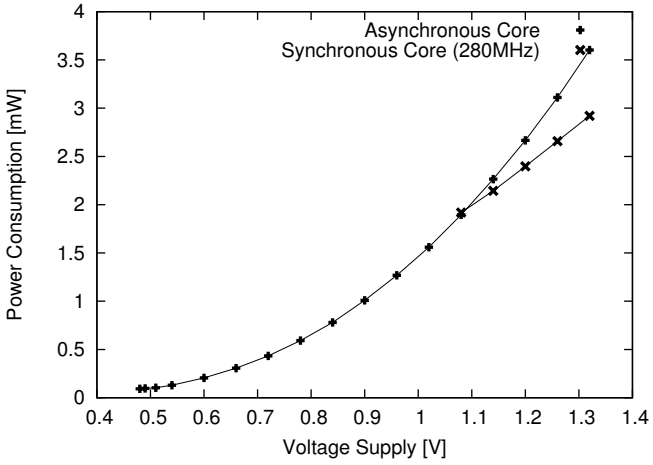

Fig. 4. Desynchronized processor power versus supply voltage

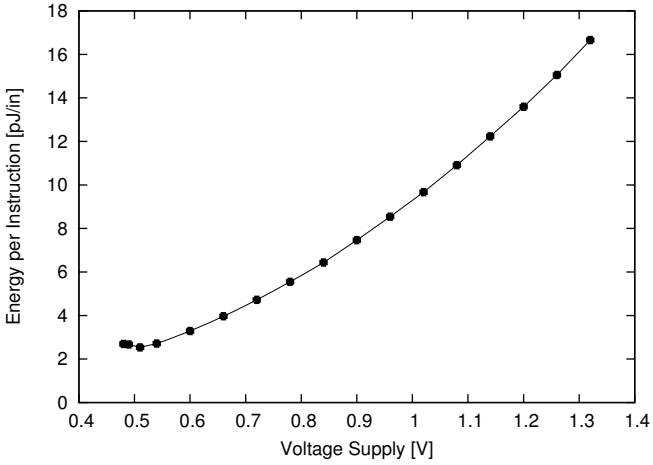

Fig. 5. Desynchronized processor energy per instruction

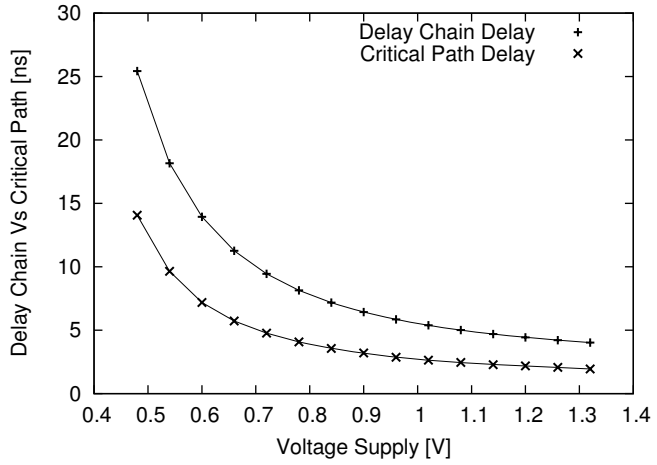

Fig. 6. Critical path versus matched delays

not subject to the meta-stability problems that are inherent in the near-critical sampling of a changing signal.

Figure 6 shows the timing of the matched delay lines and of the actual critical path of the combinational logic, when varying Vdd. Remember that in order to ensure correct operation of a bundled-data circuit the delay lines must always be slower than the longest combinational path. The figure shows that the matched delays track very well the actual logic delay, and suggests that true measurement of logic delays, by dual-rail encoding e.g., may not be necessary after all, at least for the technology under consideration, which is 130 nm. For this example, we did not try to fine-tune the delay line, since performance was not the primary goal of the exercise, but we wanted to keep the design fairly robust. Hence we chose a margin of about 100%, which is close to what synchronous design would require under nominal conditions, and much better than what the same would require at low voltage supplies..

V. Conclusions

Our processor, which is ISA compatible with the Atmel AVR, requires 14 pJ per instruction to deliver 170 MIPS at 1.2 V, and 2.7 pJ per instruction to deliver 48 MIPS at 0.54 V. These results seem to be much better than those of the other asynchronous processors reported in Section II-D (which were implemented with a 180 nm process), even when those are scaled to 130 nm. This means that even after considering the improvement in technology over one generation, which reduces according to [3] energy consumption by 65%, and after considering RAM access energy, which would approximately add 20% energy consumption for 4 KB of RAM, the results are quite comforting with respect to previous work in both the synchronous and the asynchronous domain.

We can attribute them to the *extremely low area and performance overhead* that desynchronization imposes over synchronicity. Hence the margins that are required due to the bundled data approach (about 2X, as discussed in Section IV) are still much less than the overhead due to dual-rail implementation of QDI circuits and to the extremely low granularity controllers required by the Tangram design flow.

Based on the results of this work, we can claim that the only *inherently robust* manner to achieve extreme Dynamic Voltage and Frequency Scaling is by using asynchronous techniques. Such techniques *measure reliably*, rather than either *estimate* or *measure with inherent meta-stability risks* the actual delay of combinational logic, thus taking into account manufacturing process variations, environmental conditions, and so on.

The desynchronization technique is not perfectly and natively asynchronous, because all blocks still operate in lockstep with respect to each other. However, it provides the means to measure logic delays, by using matched delay lines. Its very easy adoption, which was also shown in this project by re-implementing an existing legacy RTL core *without any re-design and almost without any knowledge of asynchronous design techniques*, makes it a very promising first step in the direction of more widespread adoption by industry.

We believe that wireless sensor networks provide a very promising application domain for very low-power and low-energy digital electronics, due to the increasing computational requirements and the stringent constraints on battery life.

In the future, we will explore more in detail how the low energy and (hopefully) low EMI characteristics of our processor can be exploited in actual WSN applications. We will also verify how well delay lines track the critical path delay in the presence of process variability.

Acknowledgments

The authors would like to thank Nikos Andrikos and Christos Sotiriou, from ICS-FORTH, for the extensive support while using their desynchronization tool and Maurizio Tranchero, from Politecnico di Torino, for his help during the first phase of this project.

References

[1] N. Andrikos. Personal communication.

[2] I. Blunno, J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou. Handshake protocols for de-synchronization. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 149–158, 2004.

[3] Shekhar Borkar. Design Challenges of Technology Scaling. *IEEE Micro*, July-August 1999.

[4] D. Chinnery and K. Keutzer, editors. *Closing the Gap between ASIC and Custom: Tools and Techniques for High-Performance ASIC design*. Kluwer Academic Publishers, 2002.

[5] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou. Desynchronization: synthesis of asynchronous circuits from synchronous specifications. *IEEE Transactions on CAD*, 2005. To appear.

[6] V. Ekanayake, C. Kelly, and R. Manohar. An ultra low-power processor for sensor networks. In *Proceedings of the international conference on Architectural support for programming languages and operating systems (ASPLOS)*, 2004.

[7] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner. Razor: Circuit-level correction of timing errors for low-power operation. *IEEE Micro*, November 2004.

[8] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. The platforms enabling wireless sensor networks. *Communications of the ACM*, 47(6), 2004.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[10] Clinton Kelly, Virantha Ekanayake, and Rajit Manohar. SNAP: A sensor-network asynchronous processor. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 24–33. IEEE Computer Society Press, May 2003.

[11] Joep Kessels and Ad Peeters. The Tangram framework: Asynchronous circuits for low power. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 255–260, February 2001.

[12] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.

[13] Alain J. Martin, Andrew Lines, Rajit Manohar, Mika Nyström, Paul Pénzes, Robert Southworth, and Uri Cummings. The design of an asynchronous MIPS R3000 microprocessor. In *Advanced Research in VLSI*, pages 164–181, September 1997.

[14] Alain J. Martin, Mika Nyström, Karl Papadantonakis, Paul I. Pénzes, Piyush Prakash, Catherine G. Wong, Jonathan Chang, Kevin S. Ko, Benjamin Lee, Elaine Ou, James Pugh, Eino-Ville Talvala, James T. Tong, and Ahmet Tura. The lutonium: A sub-nanojoule asynchronous 8051 microcontroller. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 14–23. IEEE Computer Society Press, May 2003.

[15] J. McCardle and D. Chester. Measuring an asynchronous processor's power and noise. In *Synopsys User Group (SNUG)*, 2001.

[16] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6), June 2004.

[17] Philips Semiconductors. P87cl888; 80c51 ultra low power (ulp) telephony controller.

[18] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, C. Karlof, S. Sastry, and D.Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *European Workshop on Wireless Sensor Networks*, January 2005.

[19] V. Shnayder, B r. Chen, K. Lorincz, T. Fulford-Jones, , and M. Welsh. Sensor networks for medical care. Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.

[20] H. Wang, D. Estrin, and L. Girod. Preprocessing in a tiered sensor network for habitat monitoring. *EURASIP JASP special issue of sensor networks*, 2003.

[21] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D.Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the international conference on Embedded networked sensor systems*, 2004.

[22] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *International Workshop on Sensor Network Protocols and Applications*, May 2003.