Design of 100 µW Wireless Sensor Nodes on Energy Scavengers for Biomedical Monitoring

Lennart Yseboodt[1], Michael De Nil[1], Jos Huisken[2], Mladen Berekovic[3], Qin Zhao[3], Frank Bouwens[3], and Jef Van Meerbergen[1,4]

1. Eindhoven University of Technology
Den Dolech 2
5612 AZ Eindhoven, Netherlands
lennart@belf.be
michael@flex-it.be

2. Silicon Hive
High Tech Campus 45
5656 AA Eindhoven, Netherlands
jos.huisken@philips.com

3. Holst-centre
High Tech Campus 48
5656 AA Eindhoven, Netherlands
mladen.berekovic@imec-nl.nl
frank.bouwens@imec-nl.nl
qin.zhao@imec-nl.nl

4. Philips Research Eindhoven
High Tech Campus 5
5656 AA Eindhoven, Netherlands
jef.van.meerbergen@philips.com

Abstract. Wireless sensor nodes span a wide range of applications. This paper focuses on the biomedical area, more specifically on healthcare monitoring applications. Power dissipation is the dominant design constraint in this domain. This paper shows the different steps to develop a digital signal processing architecture for a single channel electrocardiogram application, which is used as an application example. We aim for less than 100µW power consumption as that is the power energy scavengers can deliver.
assuming the constraint on power dissipation equals 100µW.
We follow a bottleneck-driven approach, the following steps are applied: first the algorithm is tuned to the target processor, then coarse grained clock-gating is applied, next the static as well as the dynamic dissipation of the digital processor is reduced by tuning the core to the target domain. The impact of each step is quantified. A solution of around 11µW is possible for both radio and DSP with the electrocardiogram algorithm.

1 Introduction

A new generation of biomedical monitoring devices is emerging. The main challenge for this kind of devices is low power dissipation. In this context a power budget of only 100µW is available for the whole system including radio, digital processing and memories. This power is taken from extremely small batteries or energy scavengers. To reduce the power dissipation of the radio data compression or feature extraction is used to reduce the number of bits that must be transmitted. Thus the bottleneck shifts towards the digital part which is the focus of this paper.

The goal of our work is to create a low-power C-programmable DSP, optimized for the application domain via hardware support for application specific instructions. As starting point a reconfigurable processor from Philips' technology incubator Silicon Hive [4] is selected. This technology includes a retargetable C compiler making code development and portability for these processors easy. This programmability is important because of the wide range of applications that can run on the nodes. Programmable nodes allow a lower non recurring engineering cost for the software and the hardware.

We differentiate between static and dynamic power dissipation. The dynamic power is the power consumed due to switching and the internal power, which is the power used inside the cells due to short-circuits and all the power used in the internal nets. It includes the functional units, memories, controller and clock. Current CMOS technology trends indicate that leakage is becoming more dominant with every new process generation. In our experiments leakage power soon turns out to be an important factor, up to 100µWof leakage was measured. Our focus has gone both into reducing static as in reducing dynamic power by minimizing the time the processor is active. As a case study we examined an ECG algorithm running on the proposed platform, what we learned from this example led to more general system level conclusions.

2 System level architecture

A generic sensor node consists of several subsystems as depicted in Fig. 1. There is a digital processing subsystem with level 1 local memory, a level 2 memory subsystem, including RAM and non-volatile memories, an array of sensors and possibly actuators, a radio system and a power subsystem including a source and powermanager, which is responsible for waking up various parts of the node when needed. This conceptual model holds independent of specific chip or die boundaries and leaves open several packaging technologies. If level 2 memories are kept off-die then multiple instances of the sensor node can be made without having to create a new chip.

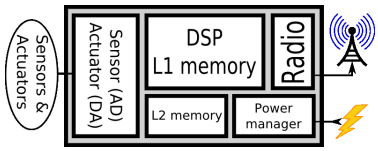

Fig. 1: Overview of the architecture of a wireless sensor node

In current systems the power is supplied by a small battery or from energy scavengers. Battery powered nodes have the disadvantage of requiring maintenance. Different forms of energy scavenging are possible but in this paper we

assume a power budget of around 100µW [5]. This number includes power consumed by the radio and the sensors, it is the global power budget of the entire sensor node.

The digital subsystem must be programmable in order to be able to run different algorithms such as ECG or EEG analysis, or altogether new algorithms from the biomedical domain. Furthermore real time constraints must be met especially when actuators are involved.

From a power dissipation point of view the most important consumers are the radio, the memory and the digital subsystem. Commercially available radios consume 150nJ/bit [7] and as a consequence the transmission of raw data can be expensive. An algorithm to reduce the amount of data via compression or feature extraction usually is a better compromise between computation and communication. In addition to the radio most subsystems exploit duty cycling and sleep modes to reduce the dissipation. Next the DSP must be tuned to the application. Also the memory subsystem can dissipate a lot of power. What is needed is a hierarchical memory subsystem optimized for power dissipation by reducing the size of the lowest level memories. These design principles will now be discussed in more detail and illustrated with an example, which is explained first.

3 Application

The electrocardiogram is a well studied topic, several interesting algorithms exist. One of the simplistic functions such an algorithm can offer is the detection for the ventricular contraction, when the heart pumps blood to the lungs and the body. In an ECG we call this event the R peak, situated in the QRS complex (Fig. 2). The algorithm we use as a testcase is based on the opensource ECG detection program from EP Limited [3].

The algorithm uses the Pan-Tomkins [1] method for R peak detection. The Pan-Tomkins method is a filtering based method to detect the frequency that is unique to the steeper R peak.

This algorithm extracts the key features reducing the amount of transmitted bits by 100x. The minimum frequency for ECG analysis is 200Hz, with a 16 bit sample width. Indeed sending raw data requires $200 * 16b = 400B/s$. Assuming 150 nJ/ bit the dissipation is 480µW which is higher than the available budget. The Pan-Tomkins method reduces this to 4B/s or 4.8 µW. The 4 bytes can hold all the information that can be extracted by this algorithm: the time between R peaks, the height of the R peak and the baseline drift.

4 Optimization DSP

After removing the radio bottleneck the problem shifts towards the DSP. Therefore we have chosen an ASIP (Application Specific Instruction set processor [12]) approach which allows to tune the core to the application domain. First we describe the reference core followed by the power optimizations.

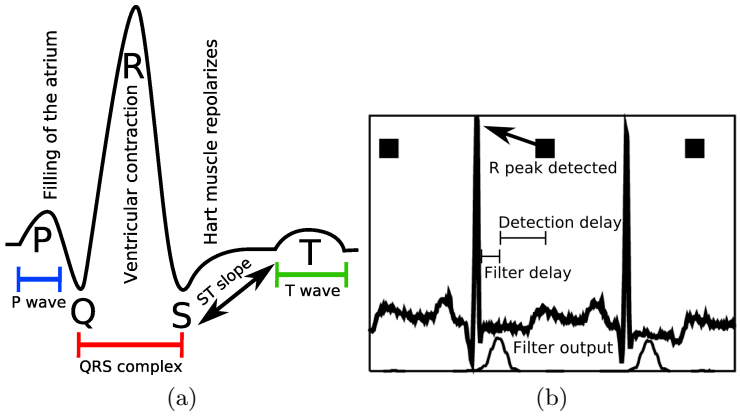

Fig. 2: The QRS complex and the detection of beats

4.1 Reference core

Because of flexibility (easy to modify) a PearlRay processor from Silicon Hive [6] was selected. The processor is reconfigurable, i.e. there exists a parameterizable description of the architecture and a C-compiler that can generate code for any possible architecture instance. The top level configuration file controls certain aspects of the processor: data widths, functional unit placement, custom functional units, configurations of the issue slots... We generated a default configuration with 32kB of data memory and 32kB of program memory. The processor is a VLIW with three issue slots, 128 bit wide instructions and is synthesized for a speed of 100MHz. This speed is the 'sweet spot' for this design. Synthesizing the core for several clock frequencies shows that speeds above 100MHz make the design grow exponentially in area and leakage as depicted in Fig. 3.

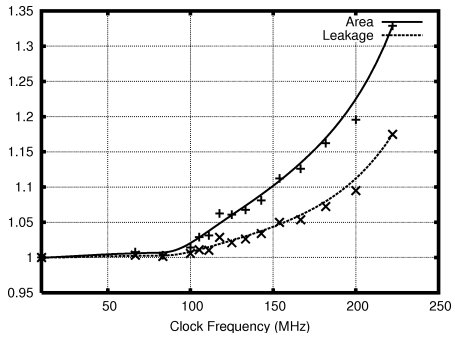

Fig. 3: Clock frequency vs. Area & Leakage

The algorithm was optimized by recoding the filters in such a way that their behavior was largely unaffected, when several expensive divisions were replaced by shifts. The PearlRay does not have a hardware divider and relies on a software divider taking 25 cycles per division. After these optimizations the cost of analyzing one sample of ECG data at a 200Hz sampling frequency was 250 cycles, however when a beat is detected this number is higher: 1200 cycles. A detection of a beat occurs only once or twice every second so on average it takes $198 \cdot 250 + 2 \cdot 1200 = 51900$ cycles per second. If the PearlRay is running at 100MHz the duty cycle is $51900/100 \cdot 10^6 = 0.05\%$.

Power figures for the processors, as seen in Table 2, were obtained using Synopsys PrimePower with layout extracted capacitances. As input a vector file from a netlist simulation was used, which was generated using Cadence Ncsim. Simulations were based on the processor netlist after layout on a 90nm CMOS process.

The power dissipation of the PearlRay was analyzed first. Three modes are identified: active, idle and sleep. In active mode the processor is running a program and processing samples. In idle mode the clock is still running. In sleep mode the only dissipation is due to leakage.

Graphically sketched this is visible in Fig. 4. In this diagram the x axis is the time that elapses, while the y axis represents the power consumption at that time. The area of the bars represents the energy consumed. The lightest bars represent the active energy, which can vary dependent on the input sample. We also observed this behavior in our ECG software. The middle bar is the idle energy and the darkest block is the ever present leakage energy.

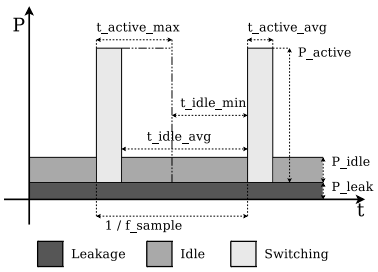

Fig. 4: Causes of power consumption over the time domain

$$P_{\mathrm{Tot}} = P_{\mathrm{Leak}} + f_{\mathrm{sample}}\Big((P_{\mathrm{Act}} \cdot t_{\mathrm{Act\_avg}}) + (P_{\mathrm{Idle}} \cdot t_{\mathrm{Idle\_avg}})\Big)$$

The ECG application is an example of an algorithm that does not require a large portion of the processing power that a typical DSP offers. The developed processor is optimized for algorithms with a low duty cycle. Table 1 shows the power characteristics of the standard version of the PearlRay, which is used as a

reference. At first glance the active power is dominant, but since the processor is only 'active' for a small fraction of the time, the actual energy usage attributed to active mode constitutes only to 0.4% of the total energy consumption. The power used in idle mode is the dominant factor here.

Table 1: Standard version of the PearlRay used as a reference. The last column shows the energy for one input sample and one ECG computation

Source	**Power**	**Duration**	**Mean Power**
Active	6.87mW	496µs	3.41µW
Idle	0.76mW	1s−496µs	758µW
Leak	100µW	1s	100µW
Total power 861.4µW			

4.2 Reduce idle mode dissipation

To counter the effects of idle energy we use coarse grained clockgating. The PearlRay reference core was already using fine grained low-level clock gates but the top level clock gate was not implemented. The top level clock gate disconnects the clock from the entire clocktree, meaning that when this gate is open no switching will occur in the processor. As a consequence an external piece of circuitry must revive the processor when this is required. Such a clock gate was very important as shown by the results in Table 3. After this optimization the dominant energy component is leakage (96%).

4.3 Reducing leakage

Now we are faced with dominant leakage power so we analyze in which part of the processor the leakage occurs. Our total leakage is 100µW, of which 50µW is caused by the data memory, 40µW by the program memory and 10µW by the processor itself. The large majority of the leakage is in the memories. We tried four things to improve this leakage.

- Reduce the size of that data memory to 2kB. Since the ECG program only requires 1.2kB and 120 bytes of stack this was possible. This reduced the leakage to 65.6 µW, a 34.5% improvement.
- By removing one of the three issue slots in the PearlRay processor and reducing the size of the immediates, the width of the program memory could be reduced from 128b to 64b. Due to the decrease of parallelism the instruction count was increased with 27%, but the instruction width was reduced by 50%, allowing us to reduce the program memory from 32kB to 16kB. This resulted in a reduction of leakage power to 82µW, a 18% improvement.

Table 2: PrimePower output results for reference PearlRay while active. The `coreio` contains the data memory.

	P_Switch	P_Int	P_Leak	P_Total	%
imec_ref	1.46e-3	5.41e-3	1.00e-4	6.97e-3	100%
core	9.11e-4	7.78e-4	9.53e-6	1.70e-3	24.4%
dec	3.86e-5	1.75e-4	2.34e-7	2.13e-4	3.1%
is_I0	1.00e-4	4.21e-5	9.67e-7	1.44e-4	2.1%
is_I1	2.71e-4	1.56e-4	2.80e-6	4.30e-4	6.2%
is_I2	8.96e-5	5.64e-5	9.61e-7	1.47e-4	2.1%
rf_I0	4.69e-5	9.61e-5	1.35e-6	1.44e-4	2.1%
rf_I1	1.03e-4	8.30e-5	2.07e-6	1.88e-4	2.7%
rf_I2	3.24e-5	5.32e-5	8.07e-7	8.65e-5	1.2%
coreio	2.21e-4	1.11e-3	5.01e-5	1.38e-3	19.8%
genI1	2.69e-6	3.45e-5	7.15e-7	3.79e-5	0.5%
genI2	3.54e-5	5.92e-5	2.69e-7	9.49e-5	1.4%
genI3	1.47e-6	6.69e-5	1.38e-6	6.98e-5	1.0%
pmem	4.14e-5	3.37e-3	3.90e-5	3.45e-3	49%

Table 3: Power results with a top level clockgate installed

Source	**Power**	**Duration**	**Mean Power**
Active	6.87mW	496µs	3.41µW
Idle	0W	1s−496µs	0W
Leak	100µW	1s	100µW
Total power 103.41µW			

– The use of memory modules designed in a technology with a high threshold option (HighV_t). This drastically reduces the leakage of the memories. They will become slower but speed was not really a constraint and the memories still operated on 100MHz. Using these memories leakage was reduced to 16.2µW, a 84% improvement.

– Reduce the datapath from 32 bit to 16 bit. As the samples are only 16 bit wide and all operations occur on them, it is optimal to scale the core to this width. This gave a moderate improvement in leakage to 94.7µW, or 5.3%.

When combining these techniques together with floorplan optimizations, the results shown in Table 4 were obtained, which reduced the leakage of the original PearlRay processor to 5.45µW, a 94.5% improvement. Furthermore scaling down the datapath to 16 bit also contributed to reduce the dissipation of the active mode.

Table 4: Power result with anti-leakage techniques combined

Source	Power	Duration	Mean Power
Active	4.7mW	628µs	2.95µW
Idle	0W	1s−628µs	0 W
Leak	5.45µW	1s	5.45µW
Total power 8.4µW			

5 System level optimization

In this section we describe system level optimizations that are a work in progress. We are currently experimenting with power gating and level 2 memories that can be used to save the state and shutdown the core.

5.1 Power down the core

From Table 4 we conclude that the leakage is still dominant. Therefore an interesting option is to power down the core and to save the state to level 2 memory and restore it when the next batch of samples have to be processed. There are positive and negative contributions to the power dissipation. In those circumstances where the final net result is positive this is an interesting option. It means a hierarchical memory subsystem: small level 1 memories with a high number of accesses and larger level 2 memories with a very limited number of accesses. This is similar to a memory hierarchy in computer architectures but optimized for power dissipation instead of performance. Level 2 memory (or part of it) is also used for other purposes, e.g. to collect the samples that arrive while the core is down or to store multiple applications, which are not active simultaneously.

Let's apply this to the ECG example. The state includes not only data (1.2kB) but also the program (16kB). This data is used to retain the state of the filters and for several other variables such as the baseline drift. An important decision is the granularity of switching between modes. If we do this at a sample basis this can become quite expensive. Assuming a low power (level 2) SRAM memory in a 90 nm process and a size of 32kB the cost of an access is 0.875 pJ/B and the leakage equals 2.5µW. If the processor is powered down after every sample the cost is 28.8µW, the calculation is detailed in Table 5. This can be improved by grouping the samples in groups of 50, then the cost of saving and restoring is also reduced by a factor 50 which translates into an acceptable level of 3.0µW. This can even be further improved to 0.5 by using a non-volatile memory (flash).

The swapping between level 2 and level 1 memories can be done for complete applications but also for parts of an application. The Pan-Tomkins algorithm for ECG is a good example. As mentioned above it consists of 2 parts: the filtering and the feature extraction. Both parts have similar code size. The filtering is executed for every sample but the feature extraction is executed with a low probability (0.5%), i.e. only when a beat is detected, which is about once per

second. Therefore it is possible to reduce the level 1 code memory by a factor of 2, which reduces the access energy. The consequence is that the programmer or the compiler must be aware of this, e.g. to insert statements for code swapping.

Table 5: Level 1 to level 2 state save calculation

Cause	**Calculation**	**Result**
Granularity: 1 sample		
Leak		2.5µW
R_{pm} [a]	$16kB \cdot 8192 \cdot 0.875pJ \cdot 200/s$	22.94µW
W_{st} [b]	$1200B \cdot 8 \cdot 0.875pJ \cdot 200/s$	1.68µW
R_{st} [c]	$1200B \cdot 8 \cdot 0.875pJ \cdot 200/s$	1.68µW
	Total:	28.8µW
Granularity: 50 samples		
Leak		2.5µW
R_{pm}	$16kB \cdot 8192 \cdot 0.875pJ \cdot 4/s$	0.46µW
W_{st}	$1200B \cdot 8 \cdot 0.875pJ \cdot 4/s$	0.03µW
R_{st}	$1200B \cdot 8 \cdot 0.875pJ \cdot 4/s$	0.03µW
	Total:	3.02µW

[a] Read program memory
[b] Write state
[c] Read state

5.2 Results

Table 6 shows a system level overview of the different components of the power consumption in µW. Furthermore the application scope is widened. The first four rows show an ECG application with different assumptions. The first row shows the simple baseline ECG case with 1 channel as discussed above. The second row assumes 3 channels. The next one is again 1 channel but now a more complex algorithm for a more extensive analysis including extra parameters (such as Q&S peaks and average beat rate). The fourth one is the same as the previous one but now for 3 channels. The last two rows show FFT analysis on 1 and 10 channel(s) respectively. The different columns represent the different contributions to the power dissipation in µW. A 90 nm process is assumed. The second column represents the radio power assuming 150 nJ/bit. Columns 3 and 4 are related to the processor and show the dissipation when active and the leakage. The next column shows the dissipation due to state-saving and restoring in a 32 KB level 2 SRAM memory. The last 2 columns show the total dissipation for 2 different scenarios. The last column assumes level 2 memory is used and the processor put in power down mode. The previous column assumes the opposite.

We conclude for various use scenarios different components can have the largest contribution in power consumption. Therefore it is not easy to predict

and a careful analysis is needed for each situation. The data in Section 4 shows that the average power consumption constraint of 100µW is feasible.

Table 6: Power consumption with different assumptions, all numbers represent micro watts.

	P_{radio}	P_{active}	P_{idle}	P_{state}	$P_{tot\ L1}$	$P_{tot\ L1,L2}$
1ch	4.8	3.3	5.5	3.0	13.5	11.0
3ch	4.8	9.8	5.5	3.1	20	17.6
1ch+	9.6	4.6	5.5	3.3	19.6	17.5
3ch+	9.6	13.7	5.5	3.6	28.7	26.8
eeg1	2.16	2.1	5.5	3.0	9.8	7.3
eeg10	21.6	21.3	5.5	3.0	48.4	45.9

6 Conclusion

Power dissipation is the most important constraint for wireless sensor nodes for healthcare applications. This paper describes the different steps in the development of an architecture using a single channel ECG application as an example. It shows that a 100µW solution is feasible.

For minimum power dissipation there is an optimum balance between computation and communication. Transmitting raw data is usually not optimal. A significant reduction in the amount of transmitted bits is obtained via compression or feature extraction. As a consequence the bottleneck shifts towards the DSP. Static as well as dynamic dissipation must be tackled. Both components are reduced by tuning the core to the target domain (application specific instructions, proper memory sizes, etc.) In an optimized architecture the level 1 memories have a limited size due to the high number of accesses in active mode. When the processor is inactive it can be powered down while the state is saved in level 2 memory. This requires that the granularity is carefully chosen. Analyzing different ECG applications it is shown that optimizing the digital processing technology is important.

Therefore this is chosen as the focus of this paper. Using ECG as a driver and adopting a bottleneck-driven step-by-step approach a factor of 100 reduction of power dissipation of the DSP core was measured via simulations. This is a result of the following actions that span the different design levels.

- Algorithm level: optimization and simplification of the code.
- Architecture level: e.g. level 1 memory size reduction by a factor of 2 for instructions and a factor of 16 for data
- Gate level: e.g. clock gating.
- Technology with HighV_t.

References

1. Rangaraj Rangayyan: *Biomedical Signal Analysis*. USA: Wiley ©2002. ISBN 0-471-20811-6.
2. J. Pan and W.J. Tompkins: *A Real-Time QRS Detection Algorithm*. IEEE Transactions Biomedical Engineering 1985, BME-32(3): 230-236.
3. EP Limited *http://www.eplimited.com*
4. Silicon Hive *http://www.siliconhive.com*
5. Bert Gyselinckx. *Human^{++}: emerging technology for body area networks.*
6. T.R. Halfhill. *Silicon Hive Breaks Out*. Dec. 1st 2003, Microprocessor Report, `www.MPRonline.com`
7. True System-on-Chip with Low Power RF Transceiver and 8051 MCU, TI Datasheet CC1110, SWRS033A.
8. Low power DSP, TI MSP430F149, *www.ti.com*
9. Coolflux DSP, *www.coolfluxdsp.com*
10. Virantha N.Ekanayake, Clinton Kelly, IV and Rajit Manohar. *BitSNAP: Dynamic Significance Compression For a Low-Energy Sensor Network Asynchronous Processor*, Proc. ASYNC, pp.144-154, Mar.2005.
11. Brett A.Warneke and Kristofer S.J.Pister. *An Ultra-Low Energy Microcontroller for SmartDust Wireless Sensor Networks*, Proc.ISSCC, Feb. 2004.
12. H. Meyr, *System-on-chip for communications : The dawn of ASIPs and the dusk of ASICs*, Proc. IEEE Workshop on Signal Processing Systems (SIPS'03), Seoul, Korea, Aug. 2003.