Electrocardiogram on Wireless Sensor Nodes

Lennart Yseboodt, Michael De Nil, Berekovic Mladen

Abstract

Wireless sensor nodes are applicable in a wide range of situations such as the medical, industrial or environmental domains, but the focus is on the biomedical domain. This paper presents the steps taken to develop a low power processor using Silicon Hive technology and mapping an electrocardiogram analysis algorithm on that processor. Today's energy-scavengers are able to deliver 100µW. This is the global power constraint of the sensor node. With a total power consumption of 16µW, the DSP processes the samples, compresses them into extracted parameters and the results are sent out by means of a radio.

1 Introduction

A new generation of biomedical monitoring devices is emerging. The main challenge for this kind of devices is low power dissipation. For example, on a 100mWh battery with efficiency 80%, a dissipation of 10mW leads to 8 hour operation, a level of 1mW corresponds to 3.3 days and 100µW translates into 33 days without scavenging (energy extracted from the environment). If scavenging is included the node can operate fully autonomously.

The goal of these sensornodes is programmability, we focus on a low duty cycle application that detects the contraction peaks in an electrocardiogram signal. This application was mapped and optimised to the target processor.

The application extracts parameters from the signal which makes transmission of the results less expensive. To transmit the full signal of an electrocardiogram measured at the minimum sample frequency of 200Hz would cost $200\text{Hz} \cdot 2\text{B} \cdot 150\text{nJ/b} = 480\mu\text{W}$. Using feaure extraction this is reduced by a factor of 50 to 9.6µW, which falls within our power budget.

2 Software optimisation

The algorithm is based on the Pan-Tomkins method of R peak detection. This R peak is situated in the QRS complex, which is a key artefact in the electrocardiogram as shown in Figure 1.

First the signal goes through a set of filters to remain only with the specific frequency of the QRS complex. The filters are, in order, lowpass, highpass, derivation, absolute value and integration of a moving window. After this stage an adaptive threshold detects R peaks.

On the Silicon Hive processor, called the PearlRay, the cost of analysing one sample was 650 cycles. The filtering stage was rewritten to do filtering on batches of 50 samples. Futhermore the division operations in the filtering stage

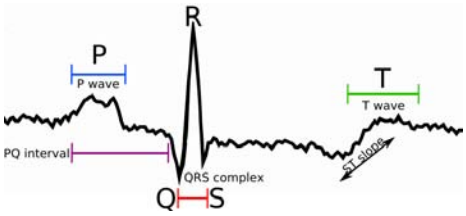

Figure 1: Electrocardiogram signal

Source	Power	Duration	Mean Power
Active	6.87mW	496µs	3.4µW
Idle	0.76mW	1s − 496µs	758µW
Leak	100µW	1s	100µW
Total power 861.4µW			

Table 1: Power results original PearlRay

were approximated by shifts and multiplies. These optimisations resulted in the same functionality using only 248 cycles.

3 Hardware optimisation

Silicon Hive processors are described in a high level object oriented language from which a processor and a C compiler for that processor can be generated. This way application specific processors can be designed in less time than usual.

The PearlRay processor is a 3 issue slot machine connected to a bus to which peripherals can be connected. Each issue slot has its own local register file and separate set of functional units. Issue slot 1 has two FIFO connections, issue slot 2 is connected to the local data memory and issue slot 3 has a master interface to the bus, which can be used for accessing external memory. The processor reads its instructions from a 128 bit wide program memory.

The power numbers of the PearlRay processor were extracted by gate-level simulation. As a first step, a default PearlRay processor is generated in VHDL containing 32kB of data memory and 32kB of program memory. This processor was then synthesized with Cadence RTLCompiler v06.10- s017 using NXP CMOS090 libraries. Chip layout was made using Cadence First Encounter v06.10-s018.

Running a power simulation on this processor yields the power results in Table 1. The processor uses 6.8mW while performing calculations, but on our 100MHz processor the duty cycle is only $248 \cdot 200/100 \cdot 10^6 = 0.05\%$ causing active power consumption to be only 3.4µW.

3.1 Optimising idle power

The dominant amount of power goes into the energy consumed due to switching while the processor is idle. Altough the PearlRay has an advanced network of clockgates internally, there is no top level clockgate. Using a top level clockgate means that when the processor is shut down an external piece of circuitry must

Source	Power	Duration	Mean Power
Active	6.87mW	496µs	3.4µW
Idle	0mW	1s − 496µs	0µW
Leak	100µW	1s	100µW
Total power 103.4µW			

Table 2: Power results with toplevel clockgate

revive it. When installing this top level clockgate the power results as shown in Table 2 are obtained. The power wasted due to switching while idle is reduced to 0.

3.2 Optimising leakage power

The leakage is now the dominant power consumer. When we examine what components leak the memories appear to be big leakers. The program memory (128bit wide, 32kB) takes 39% of the leakage, the datamemory (also 32kB, but 32bit wide) takes 50%. The core itself only consumes the remaining 11%.

We employ four methods to reduce leakage.

- **Reduce size data memory**
  Only 1.2kB of memory is required for this algorithm, therefore the data memory is reduced in size to 2kB. This causes leakage to drop with 34%.

- **Reduce with of program memory**
  The programsize is slightly less than 32kB so the program memory cannot be shrunk. By removing an issue slot and reducing the size of the immediates we can shrink the width of the instructions to 64 bit (now only 16kB is needed), thereby causing the anount of required cycles to increase to 314. This optimisation reduces leakage by 38%.

- **HighVt memory**
  The amount of leakage can be reduced further by using HighVt memory. This memory causes less leakage, but can only by synthesised at lower frequencies and with a higher active power consumption than its normal Vt counterpart. This optimisation has most effect, slashing down leakage by 84%.

- **Width of datapath**
  The samples are 16bit wide. When the processor is regenerated with a 16bit wide datapath the datamemory can again be halved. There are no measurements of this optimisations due to a bug in the tools.

With these techniques combined and an optimised floorplan for the processor the leakage was reduced to 5.5µW, a factor 18 improvement. The final results are in Table 3. The active power also went down due to the better floorplan, although the active time went up beacuse of the loss of an issue slot. The loss of this third issue slot also means that there is no possibility anymore to connect an external memory, something that might not be desirable in the final design.

Source	Power	Duration	Mean Power
Active	6.87mW	628µs	2.95µW
Idle	0mW	1s − 628µs	0µW
Leak	5.45µW	1s	5.45µW
Total power 8.4µW			

Table 3: Power results with all optimisations

4 Conclusion

This paper shows that a processor optimised for low-leakage can be generated without loss of generality. Basic feature extraction is possible with a low duty cycle algorithm. A full solution with processor and radio is possible for 18µW, well within the current power contraints of energy scanvengers.