Bennion Redd and Toren Monson

Haar Wavelet Transform for Low Power Data Compression using Dynamic Body Biasing

***Abstract* — Wireless embedded processors are not only constrained by power but also bandwidth. The hardware implemented Haar Wavelet Transform is used to transform the input data for significant compression while using very little power or chip real estate. The design also contains a dynamic body biasing scheme to improve the efficiency of the individual pmos and nmos transistors within the separate sections. Using the a very simple but effective transform is hoped to be more efficient than a normal software solution. It is also hoped that the body biasing technique will control correctly pwells so that it may be implemented in the future with autonomous control over larger circuit structures.**

***Index Terms* — Haar Wavelet Transform, Low Power, Compression, Body Biasing, Microcontrollers, Neural Data.**

I. Introduction

The Haar transform block was is planned to be an add-on to the third-generation of the WIMS (Wireless Integrated MicroSystems) microcontroller. This microcontroller has a variety of sensor interfaces and could be used for a variety of applications. Our research group is also developing chemical and electrical sensors that interface with the microcontroller. The sensors and, indeed, this entire generation of the microcontroller are geared specifically towards recording neural data. The electrical and chemical sensors will eventually be used to record various types of neural spikes and this wavelet transform will be used to compress the data before it is wirelessly transmitted.

The neural transmitter will have many recording sites for both electrical and chemical signals, and will use an Ultra-Wide Band (UWB) transmitter to send data wirelessly off the chip. The amount of data that can be collected is limited by the bandwidth of this wireless link, so data compression will allow much more data to be collected. The wavelet transform will assist in easy spike detection and compression. The wavelet transform itself does not compress data, but it tends to make many of the numbers much smaller, so that many of the bits can be dropped off without losing accuracy. When coupled with a simple run-length encoder, it will be very useful for both compression and spike detection.

The rest of the paper will be organized as follows: Section II will discuss our motivation for designing the Haar transform block. Section III discusses the testing considerations we had to take into account while designing the block. Section IV discusses the body biasing circuitry used by the block and the motivation for including it. Section V lists the methods we used to verify that our final layout does exactly what we intended to do.

II. Motivation

Wavelet compression was selected as an excellent candidate for neural spike transmission because the chip area required was significantly less than used by comparable DSP blocks and had previously been examined for the application of transmitting neural data [2]. The previous work recommended neural compression as a good application for the wavelet transform. The two wavelet transforms we examined were the Haar transform and the Daubechies D4 transform. The Daubechies could achieve better compression and more accurate signal reconstruction because it took in four numbers and multiplied them by four coefficients while calculating each resulting number. The Haar merely added or subtracted one number from another one, and then performed a right-shift so that it was finding either the average or the difference between the two numbers.

When these transforms are implemented in software to run on a CPU which already contains a multiplier, and power consumption is not a major design objective, the Daubechies D4 or even a more complicated transform would be more likely to be the best choice. However, our application has low power as a primary design objective. An adder takes up far less space than a multiplier and uses far less power as well. Because the Haar transform doesn't use any multipliers and uses only two samples at a time per number generated, it will obviously use much less area and much less power. For these reasons, the design of the Daubechies D4 transform was abandoned in favor of the Haar transform.

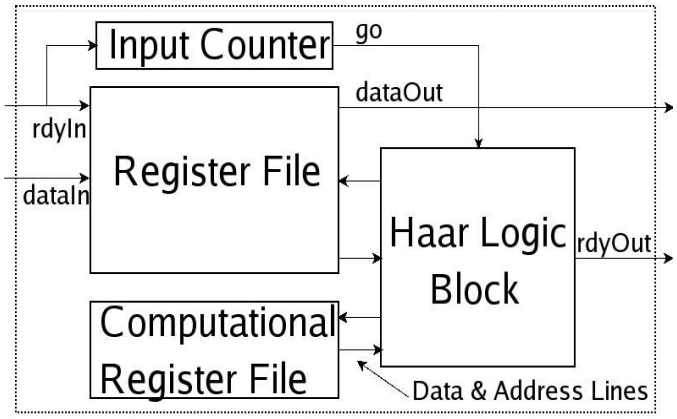

Figure 1. Block Diagram of the Haar Transform Design

While build out of one Verilog module, the Haar block can be thought of in terms of the block diagram in Figure 1. The Haar block can be thought of to be made up of register file of size 192 bits that contains the data that will be transformed, a computational register file of 96 bits used to hold temporary values created during the Haar transformation, the main logic for the transform, and a counter that sets the signals the Haar Logic block when to begin. Basically, the block takes in data as signaled by a ready flag until 16 12-bit data samples have been read in. Then, the Haar Logic block is signaled to begin accessing the data using other registers to store some of the values during computation. Finally, the ready out flag is signaled letting the outside world know that it will be transmitting out a series of 12-bit values.

III. Design for Test

The Haar compression block was developed as a separate component. Although the Haar transform block will eventually be integrated into the microcontroller's pipeline, the purpose of this tape-out is to test the blocks separately and make sure they work separately. It is being fabricated with other components on a chip, but it has its own inputs and outputs. This means that interfacing is very easy -- the Haar block reads in 16 12-bit data samples into its own registers through bi-directional pads, performs the transform, raises a signal to indicate that it's finished, and writes out the data through the same bi-directional pads. This will make the test vectors relatively easy to write.

The basic functionality of the block is not the only thing that we planned on testing. We also want to test the body bias cell ties. We have one voltage input tied to the p-well. This voltage input pad had to be surrounded by special pads to break the power rings in the padring. We will use this input to test the effect of biasing the p-well. Selecting the correct bias will minimize the power used by the block. We will change the bias slightly and monitor the power used by the chip to perform a certain number of transforms. This will allow us to verify that the well ties have been designed correctly. We will also be able to measure how much the p-well bias affects the power usage of the block.

We had previously hoped that the Haar block would have its own power domain so that power would be easily measurable. Unfortunately, due to the limited number of pads we could fit into the allowed space of the chip, we were not able to put the Haar in its own power domain. Instead, we plan on holding all of the inputs low and measuring the leakage power of the entire chip. Then we'd estimate how much of that leakage power is due to the Haar block based on the amount of area it takes up proportional to the area of the other blocks connected to the power ring. Then we'd measure the total energy used while running several transforms through the Haar. By subtracting the energy that we expect is due to leakage from other blocks, we can estimate the power usage of the Haar block alone.

IV. Body Biasing

Process variation affecting the Vth of pmos and nmos transistors is very significant in the resulting maximum frequency and power characteristics of chip. The transistors could have a high maximum frequency caused by a low Vth resulting in a chip section that always gets its results done early for the clock cycle compared with other sections of the overall chip. This would be a waste of power because a higher Vth would have still met the clock frequency while running with significantly less power. On the other side, the Vth could have been high enough that it would have forced the system clock to be slower even if the rest of the system could have operated at a lower clock rate. Now, this has demonstrated what happens given match nmos and pmos relative speeds. It is also possible for a fast pmos and a slow nmos which would meet the the clock frequency of the system but be leaking on a count of the pmos.

There are process designs that will take this in to account which have units that measure the frequency of different sections and change permanently the biasing of different sections. Still, these systems do not take into account differing pmos and nmos Vth values nor does do they take into account Vth changes over time due to transistor wear down or temperature. Using the slew rate of pmos and nmos designs, it is possible to detect not only changes that need to be made to both pmos and nmos together, but also changes that need to made individually to one well or the other. Also, such a design could continually check the values over the life time of the chip to constantly modify the changes that would

be made to one well section or the other. A design to do just this has been proposed by Amlan Ghosh [1], and part of the compression design has been to test out the effectiveness of body biasing. To do this, the cell library has been modified such that the nwells and pwells have been disassociated from vdd and gnd. Now, all the pwells for a given block are tied together for a given block and fed out as an inout pin such as vdd or gnd (as well as the nwells tied together). In future designs, these lines can be connected to a slew rate monitoring system as the outputs that will be controlled. Currently, the pwell is fed outside the circuit for dynamic control from the user while the nwell is tied to vdd. This way, the internal system will be tested while more time is made available to implement the rest of Amlan's design so that is design is tested incrementally.

V. Verification During Design

The first step in design is to create a Golden Model. Our Golden Model was a simple C program that performed the Haar transform on a set of numbers randomly generated within a specified range. Because the program is written in C, and is relatively simple, we feel very confident that it's output is correct. At every step in the design process, we verified that design operated in accordance with the original Golden Model.

After the golden model, we wrote the Verilog code for the Haar transform block and a ModelSim script to test it. We synthesized our Verilog code using Synopsys Design Compiler, and then ran our ModelSim script on it. The ModelSim script read the inputs used by the Golden Model, fed them into the Haar block, read its outputs, and compared the outputs of the Haar block to the results given by the Golden Model.

We then placed and routed the synthesized netlist using Encounter. We read the placed and routed design into Cadence, and ran the Design Rule Check (DRC) and Layout-versus-Schematic (LVS) check in Calibre. This verified that our design followed the process's design rules and that our layout matched our schematic. We these checks in place, we feel confident that our final layout matches our original Golden Model.

Total Dynamic Power	665.3306 uW
Cell Internal Power	619.6840 uW (93%)
Net Switching Power	45.6466 uW (7%)

Table 1. Power statistics for the Haar transform block

Total cell area	8410.049805 um
Combinational Area	6086.700095 um
Noncombinational Area	2323.349953 um

Table 2. Area statistics for the Haar transform block

VI. Results

While our area estimates are very good as shown in Table 2, and we are very satisfied with the area taken up by our block, there appears to be a problem with the power estimations given by Design Compiler. The cell internal power is 93% of the overall dynamic power in the design, and only 7% is the net switching power shown in Table 1. In a normal design, the cell internal power should be a much smaller fraction of the total switching power. This indicates that for some reason, nets are changing much more slowly than they ought to be. If Design Compiler is re-run pointing at even the fastest timing of our libraries that we have available, these numbers do not significantly change. We examined the Synopsys Design Compiler script and discovered that we are instructing Design Compiler not to touch the clock pin. This is because Encounter will build the clock tree later. It may be that the clock pin is driving such a large load that it changes unusually slowly, and this causes the internal cell power to be much higher than normal.

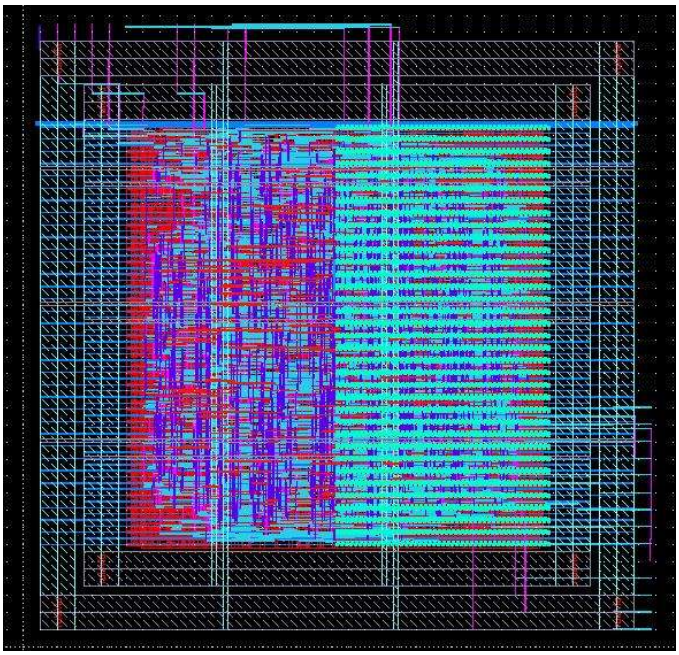

Figure 2. Haar Transform Layout

VII. Conclusion

The design was a success. The Haar transform was simulated thoroughly in a HDL and its correctness was verified. The design passed all the checks and was sent to fabrication in the 65 nm process with an output pin that controls the body bias of the pwell.

Now, there is much future work pending on the design as well as other future work that can be done during fabrication. First, there are other wavelet transforms that can be explored with the possibility of better compress for the power it uses. Also, the fabricated chip will need to be tested for correctness and power efficiency for different body biases. Finally, the power usage of the chip will be compared with software compression that the MIPS microcontroller could provide. It is possible that the MIPS processor with a bit of instruction set additions and/or register modifications could output perform the external compression block or at least come close enough to make the block unnecessary. Either way, the block will provide a good test candidate for the body biasing setup purposed by Amlan.

References

[1] Amlan Ghosh, Rahul M. Rao, Jae-joon Kim, Ching-Te Chuang, Richard B. Brown, "On-Chip Process Variation Detection Using Slew-Rate Monitoring Circuit," *vlsi-design* , pp. 143-149, 2008.

[2] Oweiss, K.G.; Mason, A.; Suhail, Y.; Kamboh, A.M.; Thomson, K.E., "A Scalable Wavelet Transform VLSI Architecture for Real-Time Signal Processing in High-Density Intra-Cortical Implants," *Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]* , vol.54, no.6, pp.1266-1278, June 2007