A Learning Digital Computer

Bo Marr, Arindam Basu, Stephen Brink, Paul Hasler
Georgia Institute of Technology
Atlanta, GA
{hbmarr, arindamb, phasler}@ece.gatech.edu

ABSTRACT
The concept of learning digital hardware is presented here. A proof of concept of a circuit that can arbitrarily control the current, and thus the switching speed and power consumption, of a digital circuit is given. This control of current is directly tuned by the feedback from the digital circuit itself, thus a learning digital computer. An argument for a completely new paradigm in digital computing follows whereby an entire system of learning digital circuits is proposed.

Categories and Subject Descriptors
B.7.1 [Integrated Circuits]: Types and Design Styles—Advanced Technologies

General Terms
Design

Keywords
Floating Gates, Learning

1. INTRODUCTION
What if processors could learn? With all of the myriad applications that our embedded systems, general purpose processors, and reconfigurable arrays of hardware are required to run, we could benefit greatly if our processors could learn exactly what it was we wanted them to do and how we wanted them to do it. Better software is not the solution for this adaptability problem; after all, the ultimate performance of software is limited by the hardware itself. For low power processors, software only complicates the matter—the more software, the more instructions, and the more power is burned.

We propose to create a processor where the hardware itself learns. The hardware will learn which application it is running and adapt to create stronger circuits in the critical path of the application and will learn which paths are not critical and thus tune down the power in those areas. The processor will remember what it has learned so that even when the hardware is powered down and the application reloaded at some later time, the processor will go back to the optimal state it learned for that application. Hardware designers or synthesis algorithms would no longer have to spend hours tweaking designs when the designer doesn’t even know for which application the circuit will ultimately be tweaked. Microcontrollers and complicated dynamic voltage scaling (DVS) algorithms would become unnecessary. Software or firmware will not be what tweaks the processor, but the fabric of the circuits themselves.

Alas, many have proposed neuronal models of learning and implemented these in analog hardware [4], and some have even proposed this neuronal process in digital by implementing equations that model learning in FPGAs [2]. However, these methods all depend on spike-based neuron models using the spike time dependent plasticity (STDP) algorithm and cannot be of use to us for a general theory on a learning digital hardware.

2. A KEY CIRCUIT ELEMENT
In order for a processor, or digital circuits, to learn a novel circuit element would be introduced with a couple of key features. It would need to be able to

• Be dynamically programmable (during run-time).
• Control current flow arbitrarily in digital circuits.
• Remember or have a memory capacity.
• Be implemented with insignificant overhead to performance or power.

Since the flow of current is what ultimately determines the speed at which a digital circuit switches and its power consumption, a circuit element with the above characteristics would allow for digital circuits to tune their own performance and power. Such a circuit element is given in Figure 1.

Represented in Figure 1 is a floating gate transistor used to control the speed and power of a digital circuit. The gate of the pFET is floating as it has no DC connection and is only capacitively coupled to other nodes, which means it can hold an arbitrary charge on the node. For a faster digital circuit, more charge is allowed onto the floating node, opening up the FET allowing more current to flow. Charge can be taken off for a more power efficient digital circuit.
The voltage on the gate, $V_{fg}$, is reduced by charge injection (putting electrons onto the gate node) and is increased through Fowler-Nordheim electron tunneling [3], but otherwise does not change and remembers the current charge.

Recall that current through a transistor above threshold neglecting velocity saturation is

$$I_{ds} = \frac{W}{L} \frac{(V_{gs} - V_{th})^2}{2}$$

$$I_{transistor} \propto V_{fg}^2$$

Thus changing the charge on the floating node would change the current, and thus the speed, in the digital circuit quadratically. In subthreshold digital, of which there has been much interest of late, the relationship is $I \propto e^{V_{fg}}$, which allows for control in ultra-low power circuits as well [3].

An experimental chip has been fabricated showing this concept which will be released in early 2009, a prototype shown in Figure 2.

3. DATAPATH: A CASE STUDY

Now that we have our key circuit element, a case study of how a processor’s datapath would benefit is given. Take for example the image processing path of a digital signal processor (DSP). In many compression algorithms, video clips, and movie sequences the pixel data only changes for a very few pixels from frame to frame. Typical image data for an H.264 decoder yields repeated inputs to an FIR filter, made up of strictly of adders and multipliers, due to the repeated pixel values being generated by the movie.

Using this H.264 movie decoder example, Figure 3 shows there are 4 carry-overs needed in the addition of a pixel value, 39, being incremented by 1, and this is the critical path using a standard ripple-carry adder. If each 1-bit addition takes unit time, the critical path takes 4 time units to complete. Now, since pixel values are repeated 100s if not 1000s of times in a typical movie scene, it is likely this exact addition, or one close to it, would be repeated 1000s of times. Our datapath learns and strengthens the critical path (Figure 3b); the first 3 1-bit additions are sped up, and the critical path now takes only 2 time units total for a 2X speed increase. It has been shown that a 2X current increase, and thus this scenario, is quite plausible with floating gate technology [1].

4. SYSTEM OF THE FUTURE

In summary, the techniques presented here could be used to create an entirely new paradigm of a learning digital computer. The next steps to be taken are to determine the best feedback mechanism (asynchronous completion cells are one method), the range of current, speedups, and power gains that can be realized, and to fabricate a datapath with this technology.

5. REFERENCES


