Low Power FPGA Logic Design – Has It Been In Front Of Us All This Time?

S. Mizrahi, Y. Ovadya, E. Hatzroni, J. Lorch

Jerusalem College of Technology – Lev Academic Center, Jerusalem, Israel

Preview

As logic design engineers, we are often required to apply numerous simple modifications to our designs, such as clock gating and enable signals. Such modifications are primarily intended to improve the logical performance of the design. On the other hand, we must also keep an eye on our circuit's power consumption. What effect do these simple logical performance enhancement methods have on power consumption?

Introduction

In recent years, the need for low power electronic components has grown rapidly. While electronic devices and their power supplies are shrinking, the tasks they are required to execute are becoming more and more complicated and power consuming. Though there are numerous RISC processors on the market, each with its own focus (size, speed, functionality etc.), there are no vendor-specific low power FPGA based RISCs. In this paper, summarize our attempt to design such a RISC processor and present a power comparison between a regular and a low power version of the same RISC processor.

Background

Field Programmable Gate Arrays (FPGAs)

Among programmable logical devices, the FPGA is unique mostly in its flexibility, which is needed in order to allow field programmability. The architecture of an FPGA is such that an FPGA can be used to implement any digital logic function.

An FPGA is divided into a number of sections. Its core is the section which contains the logic elements that implement the requested logic design. It is built of Look Up Tables (LUTs), which are groups of logical gates. Each LUT is intended to implement a logical function. Each vendor defines its own basic logic group in the core (Logic Element / Logic Block), which will usually include one LUT, one D-Flip-Flop and one Multiplexer. The main differences between vendors are usually the number of inputs to the LUTs and Multiplexers.

FPGA Power Consumption

Due to the FPGA's flexibility, FPGA components consume more power than their equivalent non-field-programmable-gate-array, the ASIC (the Application Specific Integrated Circuit) [1].

Generally speaking, we consider two components of the power used by an FPGA: static and dynamic power. Static power refers to the power consumed regardless of the task being executed in the FPGA, while dynamic power is consumed in accordance with the tasks performed by the FPGA, resulting in changes in the transistors' states.

$$P_{TOTAL} = P_{STATIC} + P_{DYNAMIC}$$
$$P_{STATIC} = I_{CC} * V_{CC}$$
$$P_{DYNAMIC} = C * V^2 * f$$

List 1: power formulas

Numerous studies have been conducted in an attempt to minimize power consumption in FPGAs. Common recent models improve power consumption mainly by dealing with the hardware aspect of FPGAs. One approach offers scaling down the supplied power by changing the internal architecture [2]. Another approach offers a change in the routing architecture [3], and still another proposes a new interconnect voltage-scaling architecture [4].

Recent research shows that power efficiency can be achieved by smart system design [5].

FPGA Based RISC Processors

In the FPGA field there are numerous FPGA-specific RISCs. Each FPGA has its own specialty. The NIOS III by Altera is declared to be high functional, very flexible and user definable [6]. Xilnix's Microblaze is smaller and more modest than the NIOS and therefore more cost effective [7]. Numerous open-source RISCs are available, none of which are vendor specific. An engineer searching for a highly functional low-power FPGA-based RISC processor will need to compromise on what the market offers.

In this paper we present an innovative RISC processor, which was developed with both functionality and low power considerations. Low power principles were implemented during the design process, and numerous applications were tried on it.

An experiment was conducted during which we compared two versions of the same RISC processor. The first version did not take power consumption into account while the second version was designed as a low-power version of the RISC processor. We chose Lattice's iCE5 (iCE40 Ultra) as the target FPGA because it includes low power solutions and because of its small physical size: 2.08mm x 2.08mm x 0.45mm [8].

Control Signals

Logical blocks may be controlled by external control signals, allowing them to either perform their task or not. When a disabling control signal is asserted, the digital circuit remains in its current state. In addition, if the circuit is designed in such manner, it is possible to make all the logical gates and transistors in the block hold their current state and prevent toggling when disabled.

Two common control signal methods are **clock gating** and **enable signals**. In clock gating, the clock signal is disconnected from internal sequential logic blocks preventing the sampling of the block input and, therefore, forcing the blocks to maintain their state without toggling. Enable-signals disconnect the inputs of combinatorial logic and, therefore, cancel any toggling in such blocks. The difference between the two is the type of logic the method is implemented on – sequential or combinatorial.

Course of Experiment

The Lev RISC Processor

The RISC processor implements 44 independent assembly language instructions, each executed in a single cycle. Each instruction is compiled into a 16-bit data code. The processor contains a register file with 64 16-bit registers, and a 16-bit ALU which is capable of performing many 16 by 16-bit arithmetic operations (ADD, SUB, MULT). It includes a module which handles routine and sub-routine programs with a 16-bit program counter and an external stack. Besides the processor, the design includes a number of peripherals necessary for basic operation and functionality (e.g. two independent timers, each either 8 or 16-bit, data and program memories, each 4096 × 16-bit). The maximum operation frequency is 56-Megaherz while running on the Lattice iCE5 FPGA.

Lattice iCE5 FPGA

Lattice Inc. has a whole line of FPGAs targeting the low power programmable component market. The latest addition to this line is the iCE5 (also known as iCE40 Ultra), which consumes less power than its predecessors. This family comes in three device versions – the iCE5LP1k, the iCE5LP2k and the iCE5LP4k, the major difference between them being the number of LUTs offered by each.

Because of the size of our design (LUT-wise), we chose the iCE5LP4k device. This model includes numerous embedded systems, which were useful while performing the experiment – a PLL, RAM and ROM memories, and a DSP block, which we used for our MULT instruction. In addition, there are a few other features which can be interfaced with the RISC if necessary.

We performed our experiment with the iCE40 Breakout Board. The on-board components handle the data communication via a USB cable connected to a PC, with an on-board FTDI component. The USB cable is also responsible for providing power to the board and to the FPGA. The

board contains a 12-MHz oscillator. In addition the board features a dedicated 1Ω resistor through which the current supplied to the FPGA flows. This resistor is for measuring the power consumption by measuring the voltage across it.

The Experiment

We chose a digital filter as the application to run on the RISC processor and to benchmark the power consumption. The filter is a low-pass FIR equiripple filter, with 16 coefficients (15^{th}-order), a sampling rate of 12-kHz and a cutoff frequency of 500-Hz. We chose this application because a digital filter is a very common design element which is often implemented on FPGAs and digital circuits as part of a DSP system [9].

We designed the filter with the MATLAB FDATool, which calculated the filter's coefficients. Because the coefficients are less than 1 (in absolute value), we multiplied each of them by 1024 (equivalent to a 10-bit shift, which can easily be fixed by hardware after computation) and rounded off the result to get sixteen natural numbers that represent sixteen coefficients. This operation makes the filter work in a non-ideal fashion, but allows us to work with coefficients without needing to use fractions.

```
receive and store sample;
for (i in 0 to 15) do
        fetch sample and coefficient;
        multiply sample and coefficient;
        add result to accumulator;
end do;
send accumulator to output;
shift samples;
wait for timer overflow;
branch to beginning;
```

List 2: pseudo code of filter

Using a waveform generator, we generated a square wave, which was input to the filter at an amplitude suitable to the FPGA IOs (3.3 Volts-peak-to-peak), with an offset that sets the square wave at the RISC's DC level – an offset of 1.65 Volts. This signal was sampled by a timer-dependent-sample-mechanism, which counts one thousand 12-MHz clock cycles and gives a sample rate as was planned in MATLAB.

Because we worked with a square wave, the resulting sample could either be a logical '1' or a logical '0', and, therefore, only one bit of memory was needed to store each sample. These samples were stored in a 16-bit shift register.

After sampling, the filter checked the last sample in the 16-bit shift register containing the last 16 samples. If this bit was a logical '1' then the first coefficient was added to the accumulator and the checking could proceed. If this bit was a logical '0' then the checking could proceed without any coefficient added to the accumulator. After 16 checks the accumulator contained the final filter output for this iteration.

After the filtering is done, the samples are shifted in the register (making room for the next sample), the accumulator content is output via the FPGA's outputs, and the accumulator is reset. Now the RISC remains on standby until the next timer overflow brings a new sample.

Because we were using an 8-bit digital-to-analog converter (the AD260AN), we needed to shift the results left twice (as the result is 10-bit). In order to see the influence of the input frequency on the filter's output, the analog result of the conversion circuit is displayed on an oscilloscope together with the input signal from the waveform generator.

Input 50 Hz
Output 2.13Vpp

Input 700 Hz
Output 780mVpp

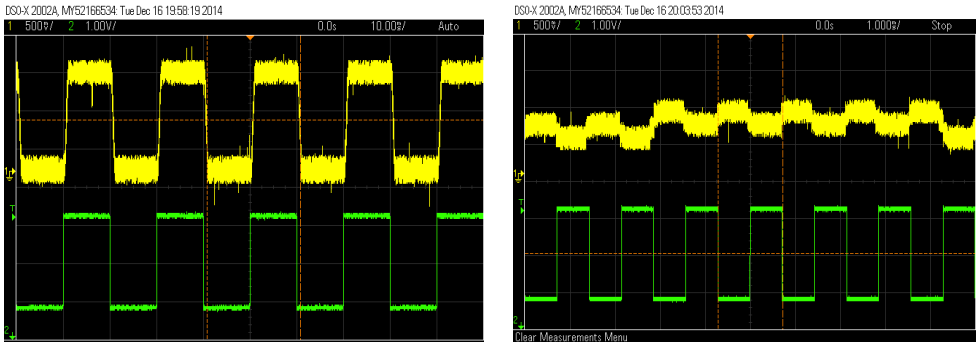

Figure 1: filter operation

In order to measure the power consumed by the FPGA, we measured the voltage across the dedicated 1Ω resistor. Since the power and current are very low, we had to amplify the signal representing the power by an amplification circuit consisting of an instrumentation amplifier. The gain of this circuit was 510. We displayed the measured power on the oscilloscope and attributed it to the filter output, as received by the DAC circuit.

As mentioned above, the experiment was executed with two different versions of the RISC processor – one which made use of low power design principles according to design structure and functionality, with reduced static and dynamic consumptions (figure 3), and another with reduced static but not dynamic consumptions (figure 2). The strategies implemented were clock gating and enable signals.

Where can I use it?

These global strategies can assist any FPGA engineer mostly of any modern FPGA vendor that wants to reduce dynamic power in his own design. The clock signal is a significant donor of capacity (see list 1) [5] due to length, therefore a targeted clock gating provides an appropriate method to deal with and can be easily implemented when using a dedicated FPGA clock network.

Regarding the enabling signals (data enable), they are very well understood at the system level and thus can effectively be defined and captured. Adding such a signal to a synchronous design block eliminates unrequired toggling also of the asynchronous parts. Global Clock signal is usually

Input 50 Hz
Output 2.13Vpp

driven from a dedicated built in global resource as a PLL/DLL element which distributes it to the entire design and is not problematic from timing aspects, obviously thanks to an absence of a clock skew that can be developed due to distributed clock signal in unequal paths. In addition, the two mentioned above will also minimize glitches [10]. Integrating the methods in the design together or separately will likely cause a designer to be more aware of his design from timing and functionality aspects but doing a judicious use of them will pay off.

Experiment Results:

We observed several interesting phenomena.

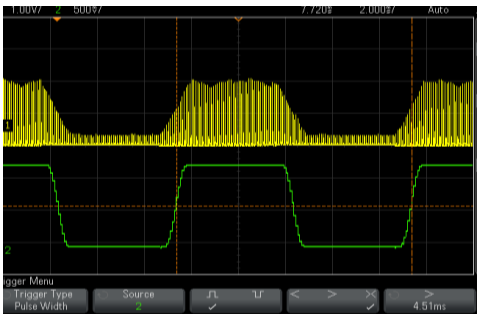

Figure 2: Top signal – power measurement. Bottom signal – filter output at 100-Hz

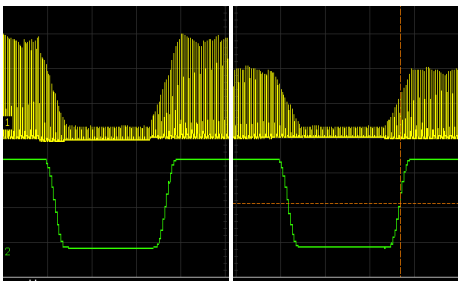

Figure 3: Comparison of power consumption in both RISC versions

As can be seen in figure 3, the output power was correlated to the filter output. The power consumed in order to produce a logical '1' was significantly greater than the power consumed to produce a logical '0'.

In addition, the power was distributed internally at a frequency of 12-kHz, which was the working frequency of the filter. During ninety percent of each cycle the RISC "*went idle*" waiting for the timer to overflow, and during another 10% it performed all the instructions for the filtering.

When comparing the two versions, we discovered that the amplitude of the amplified voltage measured across the resistor differs between the two. A visual comparison can be found in figure 3. In the basic version we measured 3.025 Volts at the peak during logical '1' production, which translates to an unamplified current of 5.9 milliamps through the resistor, and maximum momentarily power consumption of 7.08 milliwatts. The low power version showed 2.13 Volts, which translates to 4.17 milliamps and 5 milliwatts.

During logical '0' production we measured 0.435 Volts which translates to 0.86 milliamps unamplified and 1.03 milliwatts in the basic version and 0.325 Volts which translates to 0.64 milliamps unamplified and 0.77 milliwatts in the low power version.

Conclusions and Future Work:

Awareness of low power system design principles can be critical while designing logic circuits, even with applications that don't have strict power limitations. With a few basic low power methods, we were able to reduce total power consumption by 30%. The Lev RISC Processor is an example of a complex multi-task logical circuit, which shows the importance of power awareness while designing such logical circuits. It can be used to implement almost any task. Future improvements may be made to the processor in order to improve its performance.

There is still much work to be done in the area of low power system design. Our methods focused primarily on the capacitances in the dynamic power formula. The area of the frequency factor should be further explored. One option for influencing the frequency factor of the power formula

without harming the functionality of the circuit is frequency zone modulation. It includes identification of logical areas capable of working at different frequencies, dividing into different frequency zones, and distributing different clock signals to each zone.

Biography:

Josh Lorch – Is finishing a B.Sc. in electronics engineering from the Jerusalem College of Technology – Lev Academic Center, Jerusalem, Israel. His email address is joshlorch@gmail.com.

Elnatan Hatzroni – Is finishing a B.Sc. in electronics engineering from the Jerusalem College of Technology – Lev Academic Center, Jerusalem, Israel. His email address is elnatanh@gmail.com.

Acknowledgements

The authors thank Prof. S. Engelberg for his guidance while editing this article.

Sources:

iCE40 Ultra Breakout Board
Lattice Semiconductor | www.latticesemi.com

DSOX2002A Oscilloscope: 70 MHz. 2 Channels
Keysight Technologies | www.keysight.com

AD7801 8-Bit DAC, AD620 Instrument Amplifier
Analog Devices | www.analog.com

References

[1] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” IEEE Trans. on Computer-Aided Design, vol. 26, no. 2, pp. 203-215, Feb. 2007.

[2] Pooran Singh, Santosh Kumar Vishvakarma, “Device/Circuit/Architectural Techniques for Ultra-low Power FPGA Design,” Microelectronics and Solid State Electronics
2013; 2(A): pp. 1-15, 2013.

[3] S. Sivaswamy, G. Wang, C. Ababei, K. Bazargan, R. Kastner,and E. Bozargzadeh, “HARP: hard-wired routing patternFPGAs,” Proc. Int. Symp. on Field-Prog. Gate Arrays, pp. 21-25, 2005.

[4] S. Mondal and S.O. Memik, “A low power FPGA routing architecture, ” Proc. 2005 IEEE Circuits and Systems Conf., pp. 1222-1225, 2005.

[5] J. Lamoureux and W. Luk "An overview of low-power techniques for field-programmable gatearrays", Proc. IEEE NASA/ESA Conf. Adapt. Hardw. Syst., pp. 338 -345 2008.

[6] Altera Corp., "Nios II Processor Reference Handbook", 2014, Chapter 1.

[7] Xilinx Corp., "Microblaze Processor Reference Guide," Chapter 2, 2014.

[8] Lattice Semiconductor, "iCE40 Ultra Family Data Sheet," pp. 1-2, 2014.

[9] S.W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing," pp. 261-262, 1997.

[10] Naresh Grover, Dr. M.K.Soni “Reduction of Power Consumption in FPGAs - An Overview”, ijieeb 2012.