Embedded system, from software to hardware

(EDAN15 VT15 Final Report)

Anton Eliasson, dat11ael@student.lu.se
Daniel Lundell, ada10dlu@student.lu.se

May 25, 2015

Abstract

This report looks at the use of software and hardware solutions on FPGA boards carried out in 5 sessions. The solutions are profiled and discussed from three perspectives: performance, utilization and power. It's generally known that hardware implementations are faster than pure software implementations. This was tested during the sessions. A significant performance improvement can be observed when the solutions go towards hardware solutions.

1 Introduction

This is a report on the laboratory work in the *Design of Embedded Systems (EDAN15)* course at LTH. The purpose of the laboratories is to use and implement an embedded system and to observe how different implementations differ in relation to effiency, power and hardware utilization. The assignment is first solved with a pure software solution which is later parallelized and then transformed into a hardware accelerated solution. In addtion the purpose is to explore frameworks and tool chains used in development of embedded systems. The embedded system used in the laboratory is a Xilinx FPGA platform.

The report is organized as follows. Chapter 2 describes the experiments conducted throughout the five laboratory sessions. Chapter 3 presents and discusses the measurements obtained during the experiments. Chapter 4 concludes the report with a summary.

2 Experiments

The experiments were carried out over five lab sessions. The goal was to evaluate software and hardware solutions running on a Xilinix Digilent Nexys-3 FPGA platform. All software was developed in C. In every solution the performance (run-time), FPGA area utilization and power was recorded for later comparison.

The first laboratory was to implement and compare two pure software solution of a Greatest Common Divisor (GCD) algorithm for N numbers. Furthermore, the software has to run on a single processor system.

The second laboratory session was to implement and evaluate again a pure software solution of a GCD algorithm for N numbers. This time the architecture should use a multi-processor system. The system uses two MicroBlaze processors, working on the same data set.

The third and fourth laboratory session was to select a part of the GCD algorithm for N numbers and implement it in hardware. The hardware core is first written in VHDL and simulated using Xilinx ISE. It communicates with the processor using a Fast Simplex Link interface[**?**].

The last laboratory session was to integrate the hardware developed in the previous sessions into a larger system. It has software to communicate with the operator over a serial link, similar to the uniprocessor solution, but uses the hardware core for the actual computations.

2.1 Software algorithms

The algorithm chosen for computation of the greatest common divisor is the Euclidean algorithm. This was chosen because it is both simple to implement, and because it, in its simplified form that is only valid for positive integers, avoids mathematical operations that are difficult to implement in hardware such as division and modulo. The algoritm in pseudocode to calculate the GCD for two integers is shown in listing 1. To calculate the GCD for an arbitrary number of integers, the fact that the GCD is an associative function is used, so for every a, b and c, $gcd(a, gcd(b, c)) = gcd(gcd(a, b), c)$ holds.

```
function gcd(a,b)
        while(a != b) {
                if a>b
                        a = a-b
                else
                        b = b-a
        }
        return a
}
```

Listing 1: Euclidean subtracion algorithm

2.2 Single processor

The single processor system was implemented using a single MicroBlaze core. A simple software program reads data from the serial port, calculate the GCD and outputs it to the user. The GCD used is the one described above.

2.3 Dual processor

The dual processor system was implemented using two identical MicroBlaze cores with separate memory. The two cores communicates using two unidirectional Fast Simplex

Links (FSL), creating a bidirectional interface. A master-slave model of communication between the cores was used. The master core receives data from the serial link, splits the data set into two halves and sends one half over FSL to the slave core. The cores calculate the GCD for their own halves separately and in the end the slave sends its result back over FSL. The master computes the GCD for the final two values and sends the result back over the serial link.

2.4 Hardware accelerated

The part chosen for hardware acceleration was the GCD algorithm for two numbers. The hardware runs on a custom IP core on the same board as the MicroBlaze core running the software. The software reads the input over serial and outputs each element in the data set over FSL to the hardware core. The GCD is computed in hardware and sent back to the MicroBlaze core over FSL.

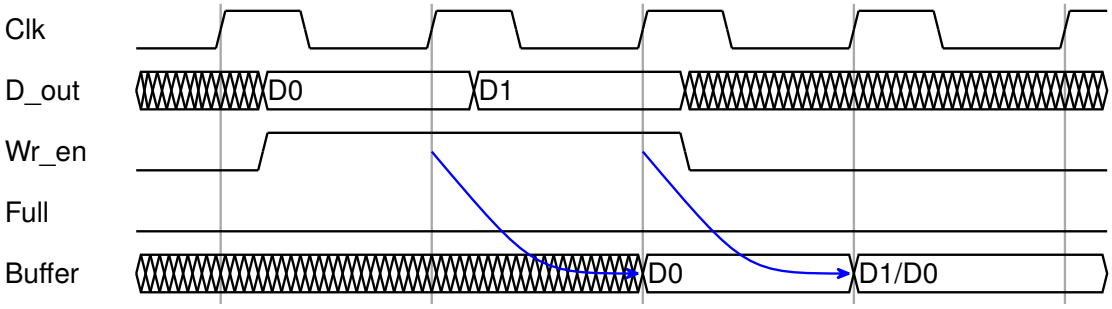

Figure 1: Timing diagram for an FSL write operation

3 Measurements and Discussion

The following chapter describes the results gained through the experiments. The clock cycles where measured when the core/cores started to share data and perform calculations. It does not include reading input or writing output through the terminal.

GCD	Single Processor	Dual processor	Hardware accelerated
10	24978	12726	1595
30	1992704	928490	100623
50	17097	9207	2485
70	108319	58745	7688
100	37876	21217	5129

Table 1: Clock cycles using the different solutions explained in chapter 2.

Say a few words about the complexity of the different solutions and how long did it take to reach a working design.

3.1 Performance

The performance figures are presented in Listing 1. When using the single and dual processor solution only software was used together with hardware support from the board. When comparing the single and dual processor solution its quite easy to see that the required cycles are almost half of the single processor. This includes sending data through the FSL, still its almost half as fast. Its not very suprising since each core only have to calculate half the dataset in the dual processor solution. This was not quite expected, we expected the FSL link to require more clock cycles. A faster solution could probably be found using direct memory sharing between the two cores. This would reduced the need for sending data over the FSL link but it might increase some time waiting when writing to memory to avoid overwriting data.

The hardware accelerated solution is significantly faster than both the single and dual processor solution. The custom IP core written in VHDL is significantly faster than the MicroBlaze core. The GCD algorithm is moved completley to a hardware solution. The MicroBlaze core is only responsible for FSL communication(and input/output but its not measured). A drawback with our solution is the lack of a GCD for N numbers, ours only provide a GCD for 2 numbers. This requires our VHDL core to wait for the MicroBlaze core each calculation. A faster solution could bave been implemented using a N number GCD in VHDL. Our expectations where quite right, we assumed the hardware solution would be faster but not that much faster. We also assumed we wouldnt get a fast result due to the 2 number GCD instead of the N number GCD.

In Listing 1 its easy to see that the clock cycles required to calculate the GCD with 30 numbers is very much slower than the rest. This is due to a drawback in the euclidean algorithm. The effciency is described by the number of divisions required. Our version uses the subtraction method so instead of division requried its described by the number of subtractions needed. In the sample data(found at cs.lth.se/edan15/labs/assignments) it can be seen that the GCD for the file contaning 30 numbers is 3 while the GCD for 100 numbers is 587. It can also be noted that the number range is quite similar in both the sample files. This requires our algorithm to do a significant more amount of subtractions in the file contaning 30 numbers than the one containing 100 numbers. This increases the number of clock cycles required to calculate the GCD.

3.2 Device Utilization

Give the FPGA resources consumed by each of your solutions. Explain how these relate to each other – e.g. whether a dual processor system has double the area of a single processor system and how do these relate to the hardware accelerated solution. Explain why or how using different algorithms influences or not the device utilization.

3.3 Power and Energy

The power and energy consumption are presented in Table 3. The power has been captured with the XPower Analyser that comes with the Xilinx ISE. The energy con-

Name	Single Processor	Dual processor	Hardware accelerated
Number of slice registers	24978	12726	1595
Number of slice LUT	1992704	928490	100623
Number of occupied slices	17097	9207	2485
70	108319	58745	7688
100	37876	21217	5129

Table 2: Clock cycles using the different solutions explained in chapter 2.

sumption has been calculated using (1).The clock cycles are choosen from the GCD with 50 numbers presented in Table 1.

$$E = P\Delta t = P * (frequency * clockcycles) \tag{1}$$

The power consumption are very alike in all solutions. The base value has to be the single processor solution since it uses as little of the boards resources as possible. There is a raise in the dual processor solution that comes with the added processor. In the hardware accelerated solution there is a slight raise, this is due to the added VHDL core. However in the energy consumption there is a significant difference. This is not unexpected since the amount of clock cycles are very different. From the equation(1) its quite easy to see how the time influences the energy consumption. The difference in the single and dual processor power doesn't influence much on the result since the clock cycles are halfed.

	Single Processor	Dual processor	Hardware accelerated
Power(W)	0.163	0.172	0.164
Energy	0.000027873	0.00001583604	0.0000040754

Table 3: Power consumption and energy calculated using the GCD for 50 numbers in Listing 1

4 Summary

It has become very clear during the sessions how software and hardware relate to each other in question of performance.

References