Embedded system, from software to hardware

(EDAN15 VT15 Final Report)

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

May 24, 2015

Abstract

This report looks at the use of software and hardware solutions on FPGA boards carried out in 5 sessions. The solutions are measured and discussed from three perspectives performance, utilization and power. It's generally known that hardware implementations are faster than pure software implementations. This is tested during the sessions. A significant improvment can be noted in performance when the solutions goes towards hardware solutions. Their is also an improvement in utilization and power depending on the solution. information about the rest of the document, but not too many details. Between 5–10 lines in this format.

1 Introduction

This is the lab report on the laboratory work in the Embedded Systems Design(EDAN15) course at LTH. The purpose of the laboratory is too use and implement an embedded system hands-on. Too see how different software and hardware solutions differ in relation to effiency, power and utilization. Starting off with pure software solution and slowly descend into hardware solutions. In addtion the purpose is to explore the framework and tool chains related to developing embedded systems. The embedded system used in the laboratory is an FPGA platform.

This part of the course is the practical part where the rest the course is more theorethical.

The rest of the report is organized as follows. Chapter 2 describe the experiments conducted throughout the 5 laboratory sessions. Chapter 3 presents the measurments and dicussions gained from the experiements in chapter 2. Chapter 4 concludes the report with a summary.

2 Experiments

The experiment was divided into 5 lab sessions. The goal was to evaluate software and harware solutions running on a Xilinix FPGA platform. The board used for testing was a Digilent Nexys-3. All software was developed using C.

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

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.

Third and fourth laboratory session was to choose a part of the gcd algorithm for N number and implement it in hardware. The hardware part was writen and simulated in VHDL using Xilinx ISE. The hardware should input/output data following a protocol specifed in the laboratory manual.

Last laboratory session was to integrate the hardware developed in the previous session into a larger system. The system should contain software to write the necessary communication and computations, to obtain a functioning hardware/software solution for the gcd of N-numbers problem.

2.1 Software algorithms

The software algorithm choosen was the Euclidean subtraction algorithm. This was choosen to avoid the need of calculations using division, modulo and recursion. The algorithm is very simple and well defined.

```
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. Software handled reading data from the serial port, calculate gcd and output it too the user. The gcd used is the one described above.

2.3 Dual processor

The dual processor system was implemented using two separate MicroBlaze cores. The communicated using two FSL-connections connected to each core. This was choosen over sharing memory between the two cores. Relation between the cores was divided

into master/slave where one core acted as the master. The software was divided onto the two cores accordingly with the master/slave principle. Software on the master core was responsible for fetching the input data, split it and send half of it to the slave core. The slave core contained only the code for communcating through FSL and calculating gcd. Both cores calculated gcd for their half the data set. Final caluclation was done on the master core after the slave had returned the gcd for the data set on the slave core.

2.4 Hardware accelerated

The part choosen for hardware acceleration was the gcd algorithm. More specific gcd for two numbers and not N numbers. The hardware runs on a custom IP-core on the same board as the MicroBlaze core running the software. The software reads input and output each element in the data set through FSL-communication to the hardware core.

Use pictures and timing diagrams, such as the one in Figure 1. Do explain every picture and diagram.

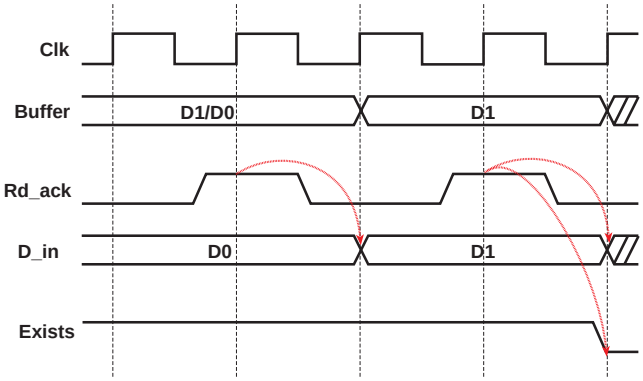

Figure 1: A figure example

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.

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

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.

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[REFERENCE:WIKI:The Art of computer programming]. 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.

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.

3.3 Power and Energy

The power and energy consumption are also important for a design. The XPower Analyser that comes with the Xilinx ISE helps you determine the power consumption for your designs. Have a look at the hierarchical breakdown of power consumption and identify the parts of your design that consume a lot of power. Also, as you know energy is the time integral of power:

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

How do your different solutions compare from the power and energy consumption point of view?

	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