Embedded system, from software to hardware

(EDAN15 VT15 Final Report)

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

May 28, 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.

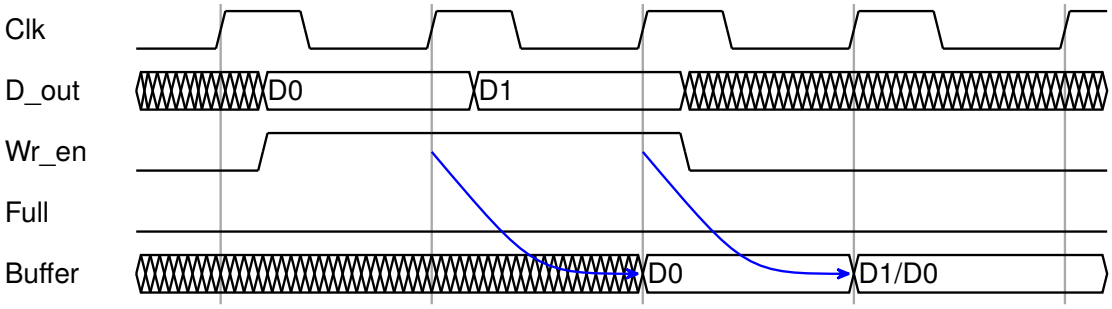

Figure 1: Timing diagram for an FSL write operation

3 Measurements and Discussion

This section describes the results obtained from the experiments. To get consistent results, the timer was configured to measure only the actual calculation and not the communication with the operator. In the hardware accelerated solution the timer was started just before the operands were sent to the hardware IP core and stopped just after the final result could be read back.

N	GCD	Single processor	Dual processor	Hardware accelerated
10	71	24978	12726	1595
30	3	1992704	928490	100623
50	6211	17097	9207	2485
70	128	108319	58745	7688
100	587	37876	21217	5129

Table 1: Performance numbers with the different solutions implemented

3.1 Performance

The performance figures are presented in table 1. In the single and dual processor solution the GCD calculation was done in software only. The results show that the dual processor solution takes just over half the time compared to the single processor solution in all datasets but one, where it takes slightly less than half. This was not suprising to us since each core can calculate the GCD for half the dataset independently, without having to wait for or synchronize with the other core until the end. The FSL transfers are included in the clock cycle counts, but the dual processor solution is still almost twice as fast. From this we can draw the conclusion that the Fast Simplex Link is very fast indeed. Using shared memory between the cores instead may yield even better performance, as no data would have to be exchanged at all.

The hardware accelerated solution is almost an order of magnitude faster than both the single and dual processor solution. The GCD calculation is moved completely into hardware. The software is only responsible for communication with the operator and communicating the operands and result with the hardware IP core. Even better performance would likely be obtained by hardware accelerating more parts, like the entire GCD for N numbers algorithm.

Table 1 shows the time taken (as measured in CPU clock cycles) to calculate the GCD for the sample data[**?**]. The table shows that the time increases when the final GCD decreases. This is because of properties of the euclidean algorithm. The time required is proportional to the number of divisions required, or in its simplified version, the number of subtractions.

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