**Journal**

**Exercise 4 HW/SW Co-design**

Date: **03/11-10**

Author: **Anders Hvidgaard Poder**

Stud.nr.: 19951439

# Introduction

In order to better understand the concept I will first discuss the concept and relate it to an example. Then I will talk about the problems and assumptions with this concept of estimation, before finally answering the assignment.

# Concept

The concept of Load Balancing mapping divides the different processes up based on computation requirements and then map them to PEs based on where there is room. In this assignment we also include a communication penalty, which is usually not part of load balancing, which only includes feasibility with respect to communication.

This assignment manually does the calculations to evaluate the mapping to the platform, just like the platform is known in advance. We can use these calculations for two purposes:

1. To determine if it is possible to map a given process to a given PE (does the PE have the capacity to do the required calculations in the allotted time).
2. To determine the total load for 1 sample through the system, including communication paths.

As there are big differences between mapping to HW and mapping to SW, the operations required to complete a process, is calculated for HW implementation and SW implementation separately (the book simply use a number for PE-speed, which is not realistic).

Furthermore the total delay through the system is artistic, as the processes may very often run in parallel, meaning that the total load through the system may be > 100% and it still being possible to get a functioning system. However if the total system load is < 100% we can be sure it is possible, as that means that we can fully process a sample before the next one arrive.

# Assignment 1

## General preconditions

|  |  |
| --- | --- |
| **Description** | **Value** |
| CPU speed (both 1 and 2) | fpe = 50MHz |
| HW speed (both 1 and 2) | fpe = 50MHz |
| HW <-> HW bus speed | Cs = 50MHz |
| CPU <-> CPU bus speed | Cs = 10MHz |
| HW <-> CPU bus speed | Cs = 25Mhz |
| 1 IIR on CPU | Ops/sample = 25 |
| 1 IIR on HW | Ops/sample = 5 |
| 1 LMS on CPU | Ops/sample = 2300 |
| 1 LMS on HW | Ops/sample = 256 |
| Input frequency | 96kHz |

Without considering the communication we can start by considering the possible mapping. This can be done by calculating the load that a given process will impose on a given PE.

To do this we use the formula: Load(PE) = fs \* (ops/sample) / fpe

|  |  |  |
| --- | --- | --- |
| **Description** | **Formula** | **Load** |
| 1 IIR on CPU | 96000 \* 25 / (50\*106) | 4,8% |
| 1 IIR on HW | 96000 \* 5 / (50\*106) | 0,96% |
| 1 LMS on CPU | 96000 \* 2300 / (50\*106) | 441,6% |
| 1 LMS on HW | 96000 \* 256 / (50\*106) | 49,152% |

From this we can quickly see that it is not possible to map the LMP to the CPU, and we need not consider this option anymore – as it is part of the assignment naturally we will.

Then we can calculate the Bus load for the different configurations:

To do this we use the formula: Delay factor = fs/Cs.

|  |  |  |
| --- | --- | --- |
| **Description** | **Formula** | **Delay factor** |
| HW <-> HW bus speed | 96000 / (50\*106) | 0,00192 |
| CPU <-> CPU bus speed | 96000 / (10\*106) | 0,0096 |
| HW <-> CPU bus speed | 96000 / (25\*106) | 0,00384 |

This allows us to calculate the total delay factor for every combination, if we so desire, yet as the assignment only mention 3 combinations we will focus on these. Remember, there are two IIR filters in serial.

## Assignment 1.1

The formula to use is: Total Load = SUM(Load(PE))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mapping** | **Platform** | **Mapping** | **Formula** | **Total Load** |
| 1 | Platform A, FPGA | LMS+IIR -> 2 FPGA | 0,49152 + 2 \* 0,0096 |  |
| 2 | Platform X,  Nios II | LMS+IIR -> 1 CPU | 4,416 + 2 \* 0,048 |  |
| 3 | Platform C,  HW/SW | LMS -> HW , IIR -> CPU | 0,49152 + 2 \* 0,048 |  |

From this it may be seen that we cannot make a purely CPU-based solution, neither with 1 CPU or 2, but we already knew that. We can also see that the pure FPGA solution is faster than the HW/SW solution.

## Assignment 2

When the Load exceed 1 it means that the given PE is loaded more than 100%, which means it is not able to perform the process in a timely manor.

## Assignment 3

As we have already determined that it is not possible to map the LMS algorithm alone to the CPU, then: No, it would not help.

|  |  |  |  |
| --- | --- | --- | --- |
| **PE/CE** | **Mapping** | **Formula** | **Total Delay factor** |
| CPU1 | LMS | 4,416 | 4,416 |
| CPU2 | IIR | 2 \* 0,048 | 0,096 |

## Assignment 4

Here it is a little complicated as there are several aspects to consider. First, should we include communication delay? Second, can the LMS and IIR execute in parallel? Are the mapping done to 1 FPGA or two?

We do this calculation based on this assumption:

* A single FPGA running two processes.
* We include the communication delay
* The two processes run in sequence (not parallel).

The formula is then: LOAD(PE) = fs \* 256 / (50\*106) + fs / (50\*106) + fs \* 5 / (50\*106) = 1

which is collected to: 1 = fs \* (0,00000512 + 0,00000002 + 0,0000001) => fs = 1 / 0,00000524 = 190840Hz

## Assignment 5

The formula to use is: Total Delay Factor = SUM(Load(PE)) + SUM(Bus Delay factor)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mapping** | **Platform** | **Mapping** | **Formula** | **Total Delay factor** |
| 1 | Platform A, FPGA | LMS+IIR -> 2 FPGA | 0,49152 + 2 \* 0,0096 + 0,00192 | 0,51264 |
| 3 | Platform C,  HW/SW | LMS -> HW , IIR -> CPU | 0,49152 + 2 \* 0,048 + 0,00384 | 0,59136 |

This delay factor cannot be used as simply as the Load, where we can simply state that if it is > 1 then it is not a possible mapping. This is due to the fact that if the processes can run in parallel a total delay factor > 1 is possible.

## Assignment 6

The Slack on the CPU running the IIR can be seen directly from the above table which states that the CPU is loaded 2 \* 4,8% = 9,6% from running the IIR filter, which means that the CPU has a slack of 100 – 9,6 = 90,4%

## Conclusion

From this assignment it is easy to see the difference in mapping to HW and SW – naturally this assignment does not take into how much longer it takes to map an implementation in HW compared to SW. This is a very important factor to include, as from the above one might wonder why on earth anyone would choose to map in both HW and CPU, when the HW can handle both. The answer to this is simple. If you have a relatively small number of units to produce, and you have to spend lets say 10 hours to develop it in SW and 100 hours in HW (not unrealistic), then at 600 kr/h you would have to save 90 \* 600 = 54000 on avoiding the CPU for it to make sense, and this is just a simple example that do not include the relative complexity of implementing changes to HW vs. SW later on.

All in all selecting between HW and SW is a complex issue, and the above methods are good to determine impossible mappings, which may then indicate a required HW implementation. It would be too much to say that one should only use HW implementation when SW is impossible, but it is a good starting point. A better version is, one should only use HW implementation when SW is impractical, i.e. always as a secondary option.

Naturally this is my own personal observation, and may certainly be subject to disagreements.