IHA | ENGINEERING COLLEGE OF AARHUS

HSC1 – Exercise 4 – Design space exploration with the load balancing algorithm

Description:
In this exercise you will use the principles from the load balancing algorithm (LBA) and the longest processing time algorithm (LPTA) to explore the design space for mapping an application to a platform. Design space exploration is about exploring different system design solution for a given functionality constrained by performance metrics. In many real life situations the platform components are fixed, due either to legacy reasons or to availability. In this exercise we will continue with the LAVMU example described in exercise 2 and 3 mapping the adaptive filter and tone control to a FPGA based platform (DE2 board).

Goals:
When you have completed this exercise you will:

- have basic experience with design space exploration
- have tried to use the principle from the LBA and LPTA on a simple example
- have used metrics in estimating performance for a given functionality mapping to different platform solutions

Exercise 4:

First you have to read and understand the GSM encoder example described in chapter 4.4 of the [1]Embedded System Design book. The GSM encoder example is based on processing the content of a block of samples. In this exercise we will analyze the situation where we have a continuous stream of samples. We will analyze different mappings of functional blocks to either software or hardware constraint by the sampling rate of fs = 96 kHz. The figure below illustrates the audio path of the algorithm we want to map to the DE2 board. The tone controller can be realized as two IIR filters. We will use the adaptive LMS filter as modeled in exercise 3 to filter the input sound before the tone controller.

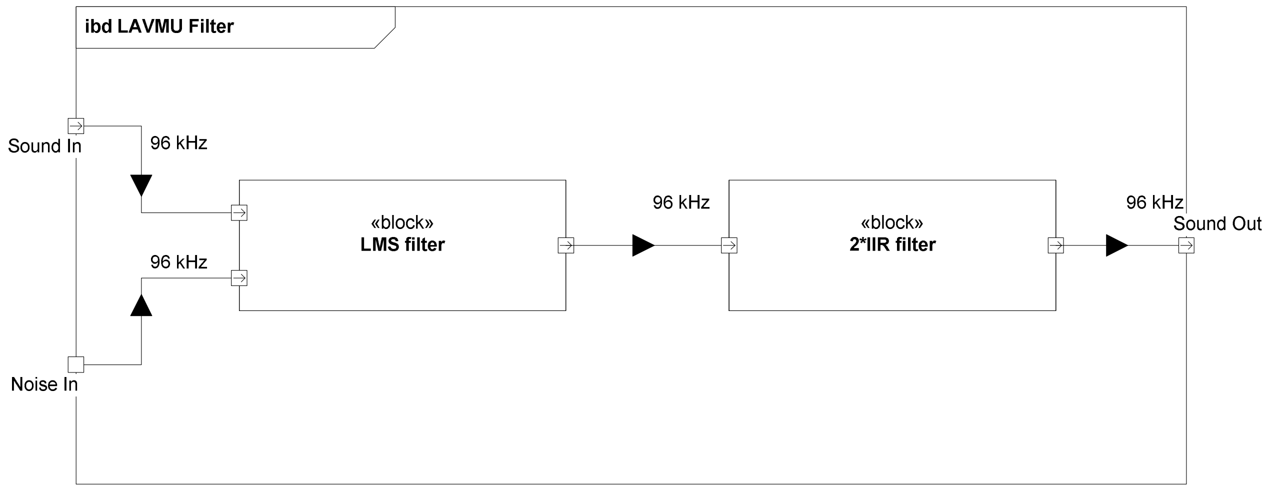

Figure 1 Internal block diagram of LAVMU filter

[1] Daniel D. Gajski, Samar Abdi, Andreas Gerstlauer, Gunnar Schirner, Embedded System Design, Springer 2009

By analyzing the two algorithms we can estimate the number of operations it takes to process one sample. The number of operations per sample for a sequential SW and parallel HW implementation are estimated by analyzing the SystemC model from exercise 3.

The estimates are given in the table below for a sequential software implementation running on the Nios II processor and a parallel pipelined optimized version running in FPGA hardware.

	Sequential (SW) **(operations/sample)**	**Parallel and pipelined (HW)** **(operations/sample)**
LMS filter (length 128)	**2300**	**256**
IIR filter	**25**	**5**

Our mapping platform consists of the NIOS II softcore processor and the FPGA running with a clock speed of fpe = 50 MHz. The communications speed (Cs) between the processing elements (50, 10 and 25) is given in mega words per second. (Word = 32bits) We assume one sample will be transmitted in one word without any overhead.

4.1 With a sample rate of 96 kHz, calculate the load of mapping to the below combinations of CPU (Nios II softcore) and HW. Calculate *Load(HW)* and/or *Load(CPU)* for each case.

The load can be calculated as: **Load(PE) = fs * Sum (ops/sample) / fpe**

1. Mapping both filters to HW
2. Mapping both filters to the same CPU
3. Mapping LMS filter to HW and IIR filter to CPU

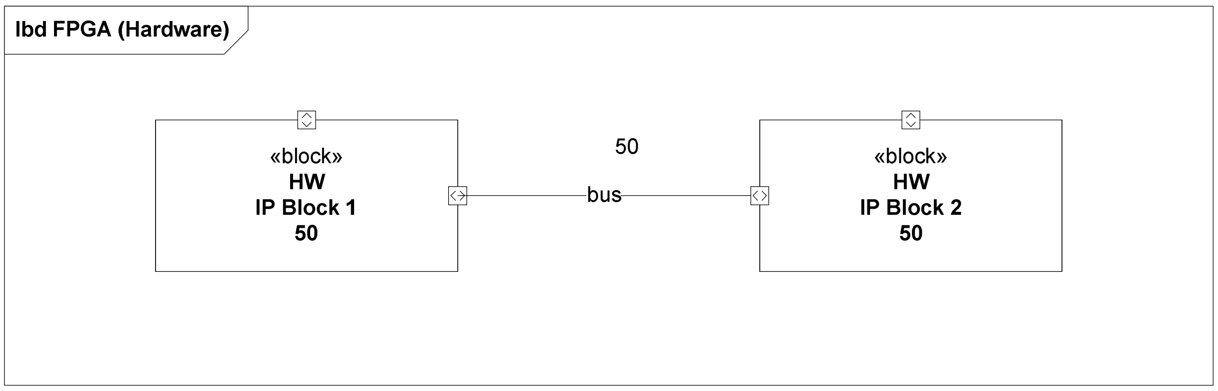

Figure 2 Hardware platform A

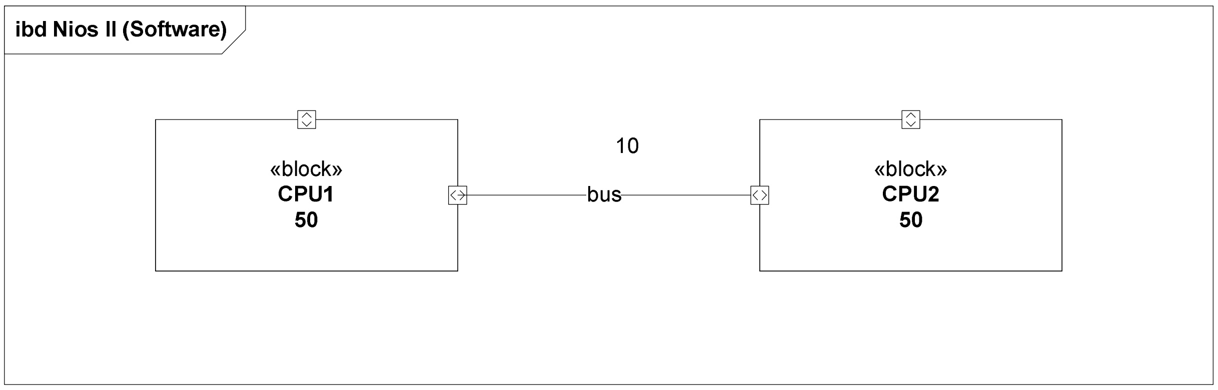

Figure 3 Hardware platform B

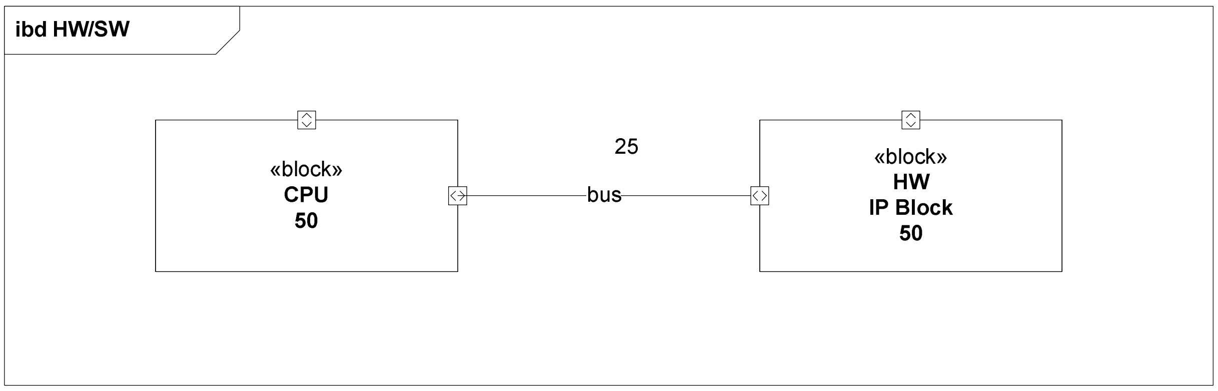

Figure 4 Hardware platform C

4.2 What does it mean in our case if the load exceeds 1?

4.3 Which mappings and platforms fulfill the design constrains for a sample rate of 96 kHz? Would it help for situation 2 to used hardware platform B?

4.4 What is the theoretical maximum sampling rate mapping both filters to HW platform A?

4.5 Calculate the communication cost in mapping to platform A and C using the longest processing time algorithm by calculating: ***E(p,PE)*** **= Load(PE) + fs/Cs**.

4.6 Calculate the Slack(PE) = Capacity(PE) – Sum Ops(p) * fs for the CPU in platform C in 4.5.