Informatics and Mathematical Modelling
Technical University of Denmark

02207 : Advanced Digital Design Techniques

Low-pass Filter (2 x 1-D)

Examination Project

Group *dt07*

Markku Eerola (s053739)

Rajesh Bachani (s061332)

Josep Renard (s071158)

December 11, 2007

Contents

1 Introduction

The project that we have implemented is a 2x1D filter of size 3x3 for convolution of an image of size 256x256 pixels.

2 Design Architecture

The overall design of the filter unit can be seen in figure 1. More detailed architecture can be seen in figure 2.

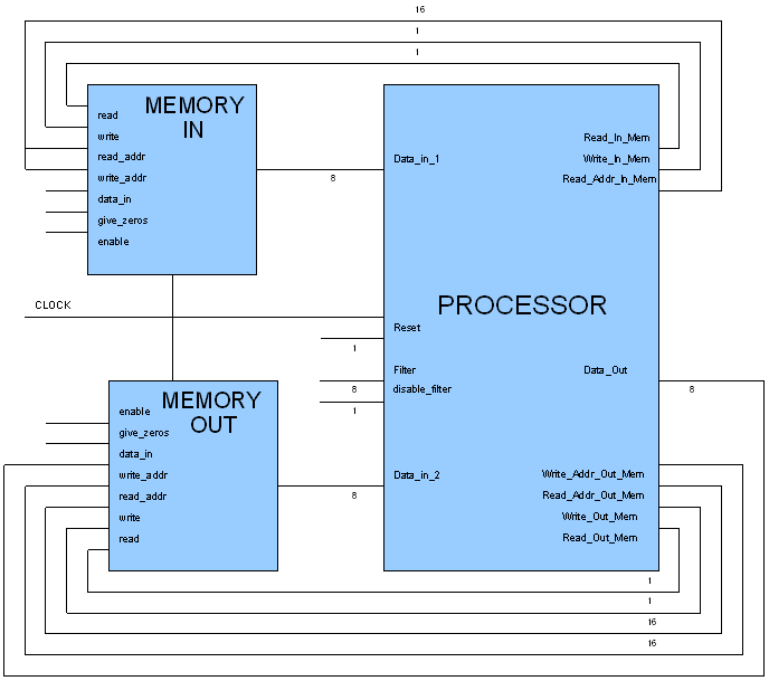

Figure 1: Filter unit design

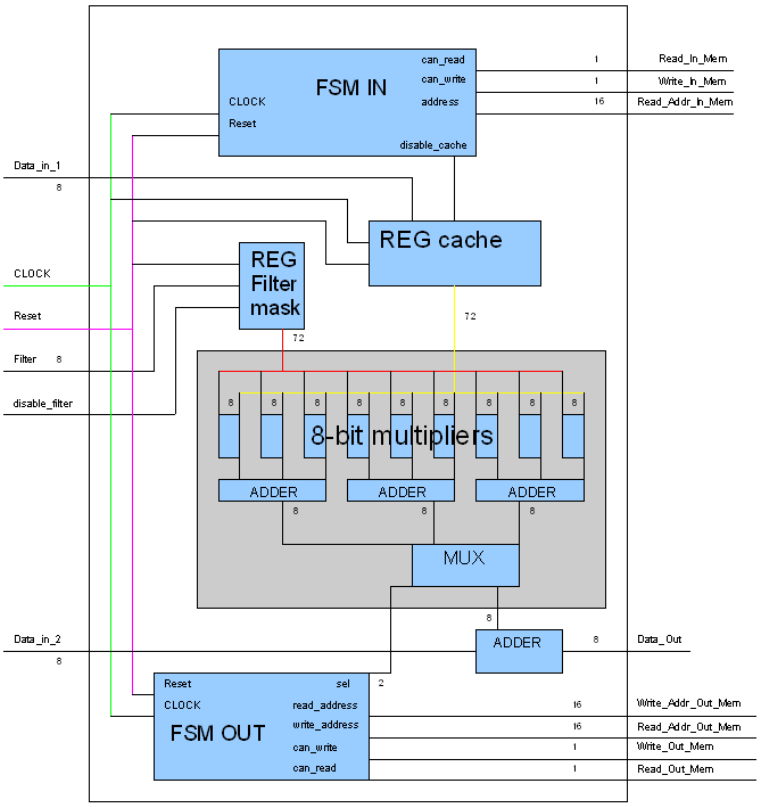

Figure 2: Processor architecture

3 Sequencing of Operations

3.1 Memory Initialization

In our case we have an image on format .hex, is the values of the image on hexadecimal, 2 values of 4 bits, 8 bits total, and we have to pass all the image to the memory so there are 65536 values on our image, so the processor are going to spend this amount on clock cycles, but we need too another memory to store the new values of the convolution for the next vertical filter pass. For this task we have a final state machine that is in charge to create an address and store the value on the memory with this address. So now we have to create two memories, the first one that we call memory in are going to be the values of the image to filter with their correspondent address, the final state machine generate the address and give to the memory the value of the image with their correspondent address, at the same time because are different state machine and memory, the space for the filtered image is created in a parallel way, but now the difference between the first one is that we want the memory empty, all with zeros, and we use the processor to make this operation, the final state machine for the output create the addresses and has the property to say the multiplexor of the final output which one wants, and one of the outputs is all zeros, so with that the processor knows that in this time period for initializing the memory out throws by Data_out a value 0 all the time. When the 65536 clock cycles are ended the memory in is with the values of the image and the memory out empty, all with values zeros.

3.2 Memory Read and Write by Processor

When the memory in and out are initialized, is time to the processor to make the operations, but how do the processor access to the memory for read and write?, very easy, we start with the premise that the final state machine out needs to read the values of the memory out, add the new value calculated and store it on the memory, this operation costs 3 clock period, but we have 3 values each convolution so the final state machine out needs 9 clock cycles for finish all his work. Now we have to synchronize the output with the input, so the final state machine in that read 3 values on 3 clock cycles has to wait 9 clock cycles for the next read for memory, and this all the time because he may allow the output to make their own calculations. So the input data is red on periods of 3 clock cycles and wait 9 for allow the output to generate the new value and don't overwrite some value that can be modify the output.

3.3 Memory Access Sequence

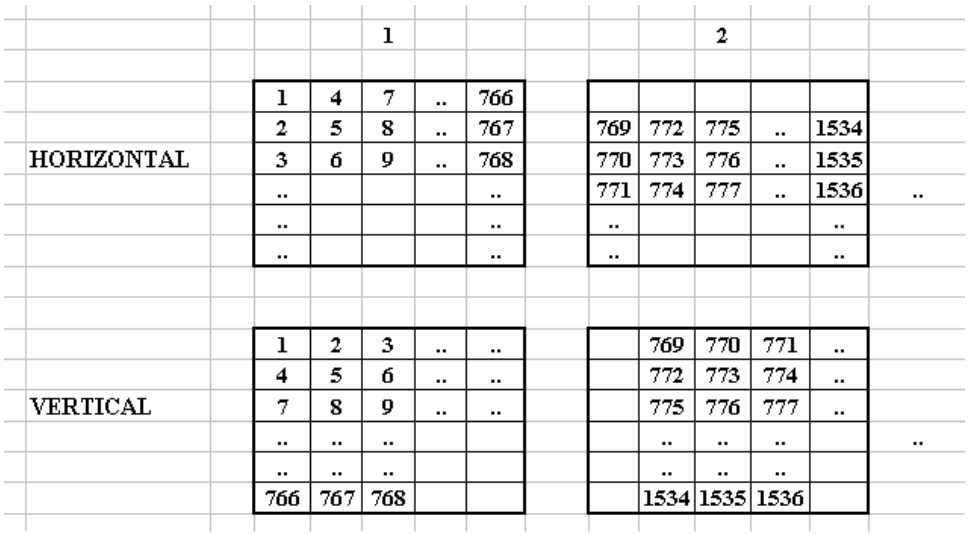

Figure 3: Sequence of Memory Address Access

For the memory sequence we use a particular way, we want to access to the memory the less time possible, so we have created one kind of memory cache inside the processor, the cache consist in one ShiftRegister for store 9 pixels, the number of pixels that we need to filter with our mask of 3x3, and on the beginning the final state machine take from the memory 9 values, for fill all the cache, but on the next clock cycle only take 3 values from the memory, the next column if we are on the horizontal pass (Figure 1(file horizontal pass)) or the next row if we are on the vertical pass (Figure 2(file vertical pass)) as we can see. But the sequence has a particular thing, when the values of the memory arrives to the end of the picture now doesn't work the shiftregister because we have to come back to the beginning and start again, so when the image values seized from the memory, last column on the horizontal pass and last row on the vertical pass, there are another time 9 lectures from the memory to fill up all the shift register and continue the same mechanism like l say before.

4 Finite State Machines

4.1 Input Controller

4.2 Output Controller

5 Synthesis

We synthesized the design using four different clock periods, namely 7ns, 5ns, 3ns and 2ns, and let Design Vision try to optimize the design for speed to get the fastest possible design. Turns out 2ns is the minimum clock period for our design, Design Vision was not able to synthesize a faster design even when we tried. To get meaningful power reports we simulated switching activity with the VSS Simulator and the activity was passed on to Design Vision. On top of power reports we also obtained area and timing reports from the design on all four clock cycles. The actual reports can be seen in the appendix, but a summary of the results can be seen in table 1.

Table 1: Summary of Design Vision reports

$\mathbf{T}_C[ns]$	$\mathbf{P}_{stat}[mW]$	$\mathbf{P}_{dyn}[mW]$	$\mathbf{P}_{tot}[mW]$	$\mathbf{A}_{comb}[\mu m^2]$	$\mathbf{A}_{tot}[\mu m^2]$	$\mathbf{T}_{cp}[ns]$
7	0.11	1.60	1.71	44067	53079	4.7
5	0.11	1.71	1.82	44067	53079	4.7
3	0.13	2.19	2.32	49595	58611	2.9
2	0.20	2.60	2.80	58668	67700	1.9

6 Results

Table 2: Summary

$\mathbf{T}_C[ns]$	**Critical Path** [ns]	**N. cycles** (256 x 256)	$\mathbf{E}_{pc}[mW/MHz]$	**AREA** $[\mu m^2]$
7	4.7	3121152	0.01197	53079
5	4.7	3121152	0.00910	53079
3	2.9	3121152	0.00696	58611
2	1.9	3121152	0.00560	67700

7 Discussion

8 Limitations

We have designed and implemented the architecture for an image filtering processor which uses a 3x3 filter to perform a convolution on the image of size 256x256. Though the results

of the convoluted image look promising, as shown in section 6, we are aware of some of the limitations of the work. Given more time, we would have liked to add the following missing aspects into the project.

1. We have just been able to implement the 3x3 filter for the convolution. Though, it was proposed that we would implement the higher dimension filters as well, including 5x5, 7x7 and 9x9, we were not able to do so, due to the initial problems we faced in the implementation of the 3x3 filter itself. We believe the results obtained in the section 6, in the form of the convoluted image could be better if the dimension of the filter is higher. In those cases, the blur effect on the image would be clearly evident, as compared to the case of the 3x3 filter. We would like to briefly mention how the design of the processor would be modified if we wish to convolute the image using higher dimension filters. If we consider the dimension of the filter as nxn, then we have the following:

   - Number of Adders = n^2
   - Number of Multipliers = n
   - Size of the Cache Shift Register = n^2
   - Multiplexer would have n inputs and 1 output.
   - Select signal from the Output Controller would be 3 bits for 5x5 and 7x7, and 4 bits for 9x9 filter.

   In addition the synchronization of the Input and the Output Controllers would change due to the number of clock cycles required to get n new pixels from the memory and compute the output for n pixels at a time. This means the cases described in section 3.2 would now be the following:

2. It is assumed that the filter is symmetric along the two dimensional x and y axes. We need this since the indices of the filter which need to be multiplied with the image pixels would change in horizontal and vertical movements. For simplicity therefore, we have made this assumption. The solution to this problem is quite simple though. We just need to have separate caches in the processor which hold the filter values. For horizontal movement we would use one cache, while the other one would be used for the vertical movement.

3.