ECE532 Project Group Report

MD Simulation Performance Acceleration based MPI on FPGA Platform

Hao Jun Liu, Chao Yu
haojun.liu@utoronto.ca, chao.yu@utoronto.ca

1 Project Overview

1.1 Goals

The first goal of our project is to develop an MD(Molecule Dynamic) simulation engine on FPGA. The second goal is to add MPI as the synchronization protocol so that it is possible to run multiple simulation engines on FPGA. The ultimate goal is to measure the performance of the simulation engines and pin point the performance bottlenecks so that improvement of the engine can be made.

1.2 Project Background

MD simulation is a simulation technique that calculates atoms' positions based on the forces between them. The algorithm we have is implemented in C++ and we used ImpulseC compiler to compile the C++ code into VHDL. ImpulseC is a subset of C with additional libraries provided. The Impulse C code used in our design is hand written based on the C++ code we have. The ultimate goal is to use a cluster of FPGAs to run MD simulation faster compared with regular high performance machines with lower power.

1.3 System Block Diagram

Fig. 1 is the current block diagram of the simulation core.

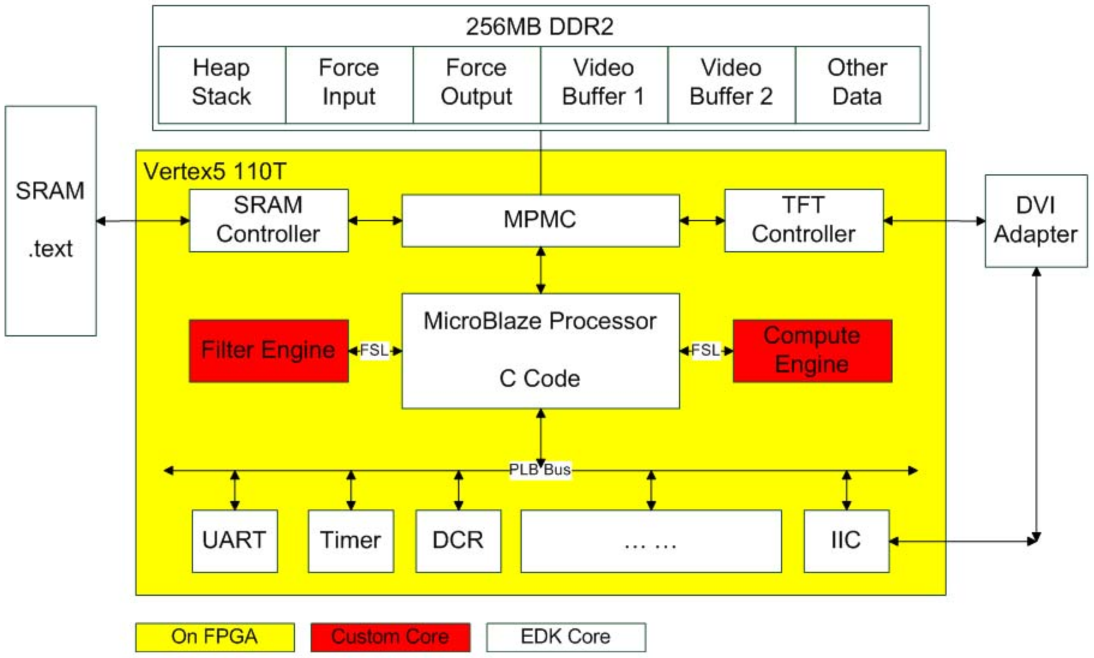
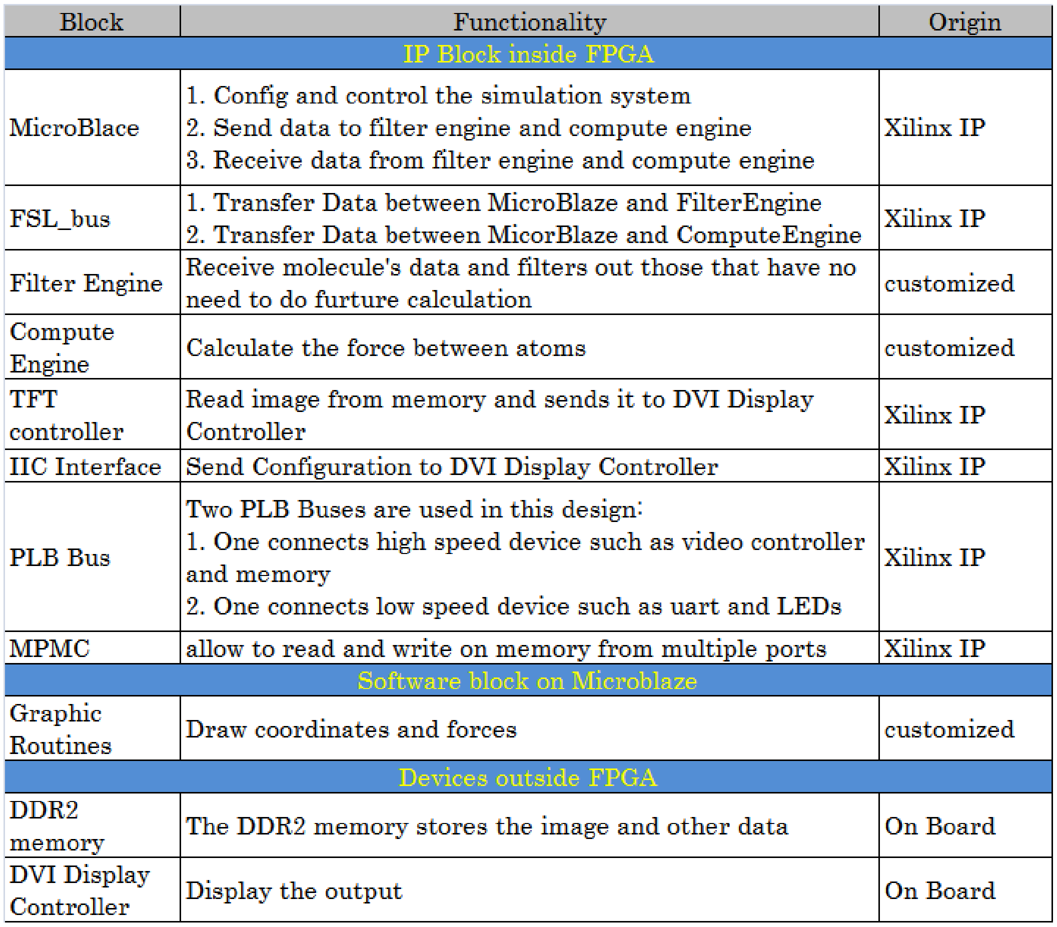


Figure 1: System Block Diagram

1.4 General Description of IP core used

This subsection provides a brief description of IP cores used in our design. Fig. 2 is the table that illustrates the IP cores used in the design.



Block	Functionality	Origin
	IP Block inside FPGA	
MicroBlace	1. Config and control the simulation system 2. Send data to filter engine and compute engine 3. Receive data from filter engine and compute engine	Xilinx IP
FSL_bus	1. Transfer Data between MicroBlaze and FilterEngine 2. Transfer Data between MicorBlaze and ComputeEngine	Xilinx IP
Filter Engine	Receive molecule's data and filters out those that have no need to do furture calculation	customized
Compute Engine	Calculate the force between atoms	customized
TFT controller	Read image from memory and sends it to DVI Display Controller	Xilinx IP
IIC Interface	Send Configuration to DVI Display Controller	Xilinx IP
PLB Bus	Two PLB Buses are used in this design: 1. One connects high speed device such as video controller and memory 2. One connects low speed device such as uart and LEDs	Xilinx IP
MPMC	allow to read and write on memory from multiple ports	Xilinx IP
	Software block on Microblaze	
Graphic Routines	Draw coordinates and forces	customized
	Devices outside FPGA	
DDR2 memory	The DDR2 memory stores the image and other data	On Board
DVI Display Controller	Display the output	On Board

Figure 2: IP Core Used

1.5 Clock Domain

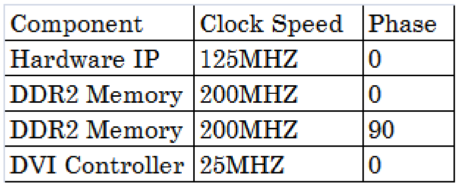
This subsection provides a brief description of different clock domains used in our design. Fig. 3 on the following page is the table that lists all the clock domains in our design.

The DDR2 memory runs at 200MHZ and DVI Controller runs at 25MHZ. The DVI controller helps to resolve the clock crossing domain issue with PLB bus which runs at 125MHZ.

2 Final Design Outcomes

2.1 Review of original plan

One of the features in the initial plan is to use MPI network to feed in data of molecules; these data will be used for calculating forces. However, due to the tight schedule, we decided to use MicroBlace processor to



Component	Clock Speed	Phase
Hardware IP	125MHZ	0
DDR2 Memory	200MHZ	0
DDR2 Memory	200MHZ	90
DVI Controller	25MHZ	0

Figure 3: Clock Domain

transfer the data to hardware cores via FSL bus.

Another important change in the project is to reduce the number of floating points calculation in the FilterEngine block and ComputeEngine block. This is because the float point calculation consumes over 300 DSP cores on FPGA. The FPGA we have contains only 64 DSP cores. The total number of required DSP cores exceeds the number available on the FPGA. Therefore we decided to reduce some floating points calculation that are not critical in the design.

2.2 Final Design

We have successfully build a system that is capable of performing MD simulation. The FilterEngine block can receive input data of molecules, send out desirable ones via FSL bus and discard the rest. The ComputeEngine block can receive data from the FilterEngine block and calculate forces between molecules. Both blocks are customized and implemented in hardware. Finally the result forces can be plotted on the monitor using DVI controller. The drawing function is implemented in software running on MicroBlaze processor. The requirement of this project is to have at least one processor and have at least one customized core in the design. We have successfully accomplished that.

In order to achieve better performance, our design requires significant improvememnts and this will be discussed in section 4.

3 Description of IP Blocks

3.1 MicroBlaze Processor

MicroBlaze is a built-in soft processor that is provided as part of the EDK.[5] The processor is added into the design through Xilinx EDK project wizard.[1] The processor has the following two functions:

- MicroBlaze contains the instructions for writing and read to and from system memory and FSL bus.
- A software program is running on the MicroBlaze which controls the entire system.

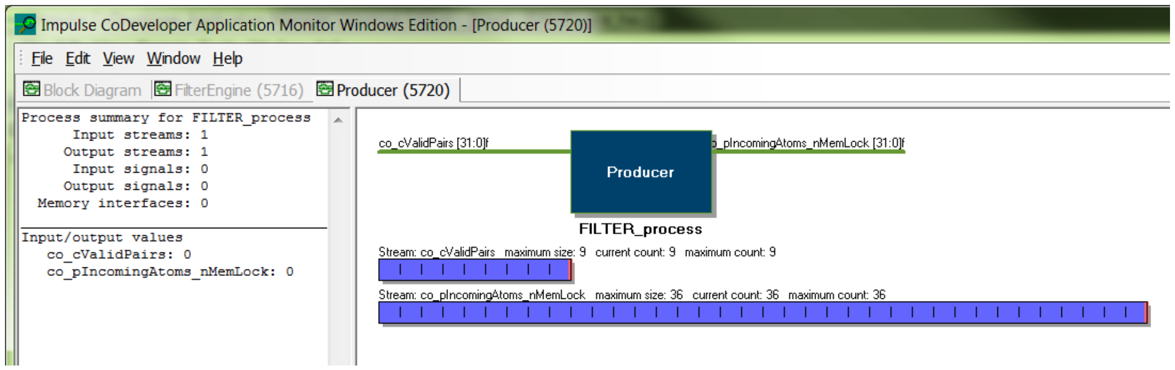
3.2 FSL Bus

FSL (Fast Simplex Link) bus is provided in the Xilinx IP catalog named as FSL_V20. The FSL bus is a uni-directional point-point FIFO-based communication channel bus. It is mainly used for transfering data from register file on the MicroBlaze processor to FilterEngine Block and ComputeEngine Block.[2]

3.3 Filter Engine Block

Filter Engine Block is a fully customized block. One molecule is surrounded by multiple molecules. Only molecules nearby contribute significant forces. Hence the function of this block is to determine which molecules are necessary for calculating forces. This block receives positions of molecules from MicroBlaze and writes back to MicroBlaze via FSL bus.

The Filter Engine is designed by first writing Impulse C code that is supported by the ImpulseC compiler. The compiler then generates HDL that can be used as an IP core in EDK. Two levels of simulations are performed to ensure the functionality of the core. The first one is a software level simulation performed by the CoDeveloper Application Manager, Fig. 4 gives an example of software simulation of the Filter Engine. We did another level of simulation, the behavioral simulation. Fig. 5 on the following page gives an example of behavioral simulation using Modelsim.


Impulse CoDeveloper Application Monitor Windows Edition - [Producer (5720)]
File Edit View Window Help
Block Diagram
FilterEngine (5716)
Producer (5720)
co_cValidPairs [31:0]
co_pIncomingAtoms_nMemLock [31:0]
Producer
FILTER_process
Stream: co_cValidPairs maximum size: 9 current count: 9 maximum count: 9
Stream: co_pIncomingAtoms_nMemLock maximum size: 36 current count: 36 maximum count: 36

```
Process summary for FILTER_process
      Input streams: 1
     Output streams: 1
      Input signals: 0
     Output signals: 0
  Memory interfaces: 0

Input/output values
   co_cValidPairs: 0
   co_pIncomingAtoms_nMemLock: 0
```

Figure 4: Filter Engine Application Manager Simulation

3.4 Compute Engine Block

ComputerEngine Block is another fully customized hardware block. The function of this block is to calculate forces between molecules. This block receives data produced by the FilterEngine block from MicroBlaze and writes back to MicroBlaze via FSL bus.

The Compute Engine is designed in the same way as the Filter Engine. Fig. 6 on page 6 gives an example of behavioral simulation using Modelsim.

3.5 DVI controller Block

DVI(Digital Visual Interface) controller is provided in the Xilinx IP catalog. The DVI controller reads data from a video buffer and displays the image on screen.[3]

3.6 Software Program

The software is capable of plotting coordinates and lines on the screen based on Bresenham's line algorithm. This program is running on MicroBlaze Processor.

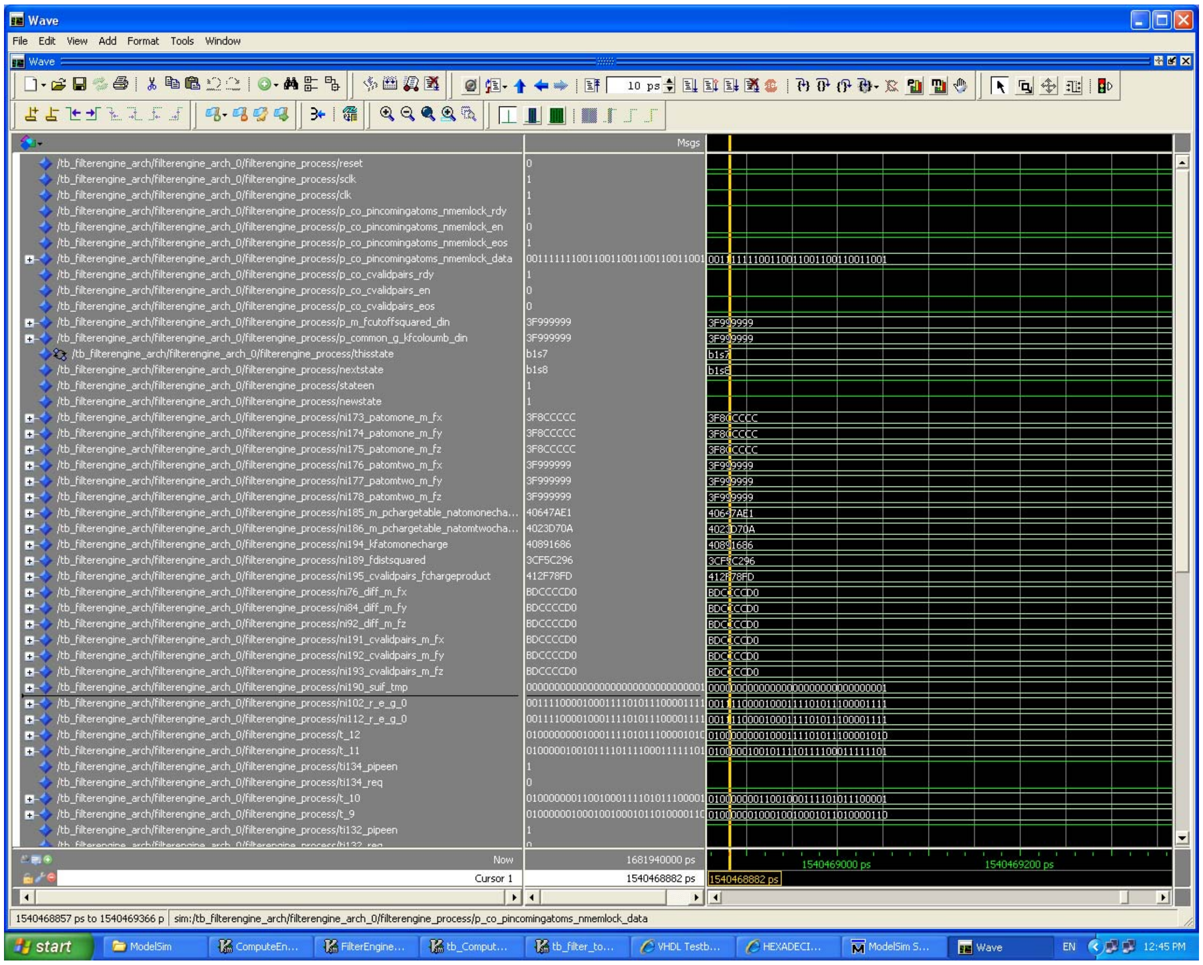
4 Design Methodology

4.1 Impulse CoDeveloper

The customized blocks Filter Engine and Compute Engine are generated by impulse C software-to-hardware compiler.[5] This compiler is included in the Impulse CoDeveloper. In the design process, we used this tool to compile impulse C code to VHDL code. It has been demonstrated that the generated VHDL code is reliable and also is synthesizable by Xilinx ISE. By using this tool, the IP core design time is reduced significantly.

Here is a list of advantages and disadvantages or space of improvement of the Impulse CoDeveloper Advantages:

- Much lower hardware development time if the designers understand the limitation of this compiler



```
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/reset                                0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/sclk                                 1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/clk                                  1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_pincomingatoms_nmemlock_rdy     1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_pincomingatoms_nmemlock_en      0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_pincomingatoms_nmemlock_eos     1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_pincomingatoms_nmemlock_data    00111111100110011001100110011001
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_cvalidpairs_rdy                 1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_cvalidpairs_en                  0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_cvalidpairs_eos                 0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_m_fcutoffsquared_din               3F999999
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_common_g_kfcoloumb_din             3F999999
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/thisstate                            b1s7
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/nextstate                            b1s8
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/stateen                              1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/newstate                             1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni173_patomone_m_fx                  3F8CCCCC
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni174_patomone_m_fy                  3F8CCCCC
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni175_patomone_m_fz                  3F8CCCCC
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni176_patomtwo_m_fx                  3F999999
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni177_patomtwo_m_fy                  3F999999
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni178_patomtwo_m_fz                  3F999999
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni185_m_pchargetable_natomonecha...  40647AE1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni186_m_pchargetable_natomtwocha...  4023D70A
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni194_kfatomonecharge                40891686
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni189_fdistsquared                   3CF5C296
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni195_cvalidpairs_fchargeproduct     412F78FD
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni76_diff_m_fx                       BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni84_diff_m_fy                       BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni92_diff_m_fz                       BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni191_cvalidpairs_m_fx               BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni192_cvalidpairs_m_fy               BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni193_cvalidpairs_m_fz               BDCCCCD0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni190_suif_tmp                       00000000000000000000000000000001
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni102_r_e_g_0                        00111100001000111101011100001111
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ni112_r_e_g_0                        00111100001000111101011100001111
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/t_12                                 01000000001000111101011100001010
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/t_11                                 01000001001011110111100011111101
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ti134_pipeen                         1
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ti134_req                            0
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/t_10                                 01000000011001000111101011100001
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/t_9                                  01000000100010010001011010000110
/tb_filterengine_arch/filterengine_arch_0/filterengine_process/ti132_pipeen                         1

Now        1681940000 ps
Cursor 1   1540468882 ps

1540468857 ps to 1540469366 p | sim:/tb_filterengine_arch/filterengine_arch_0/filterengine_process/p_co_pincomingatoms_nmemlock_data
```

Figure 5: Filter Engine Modelsim Simulation

- Easier to add functionality into a hardware block

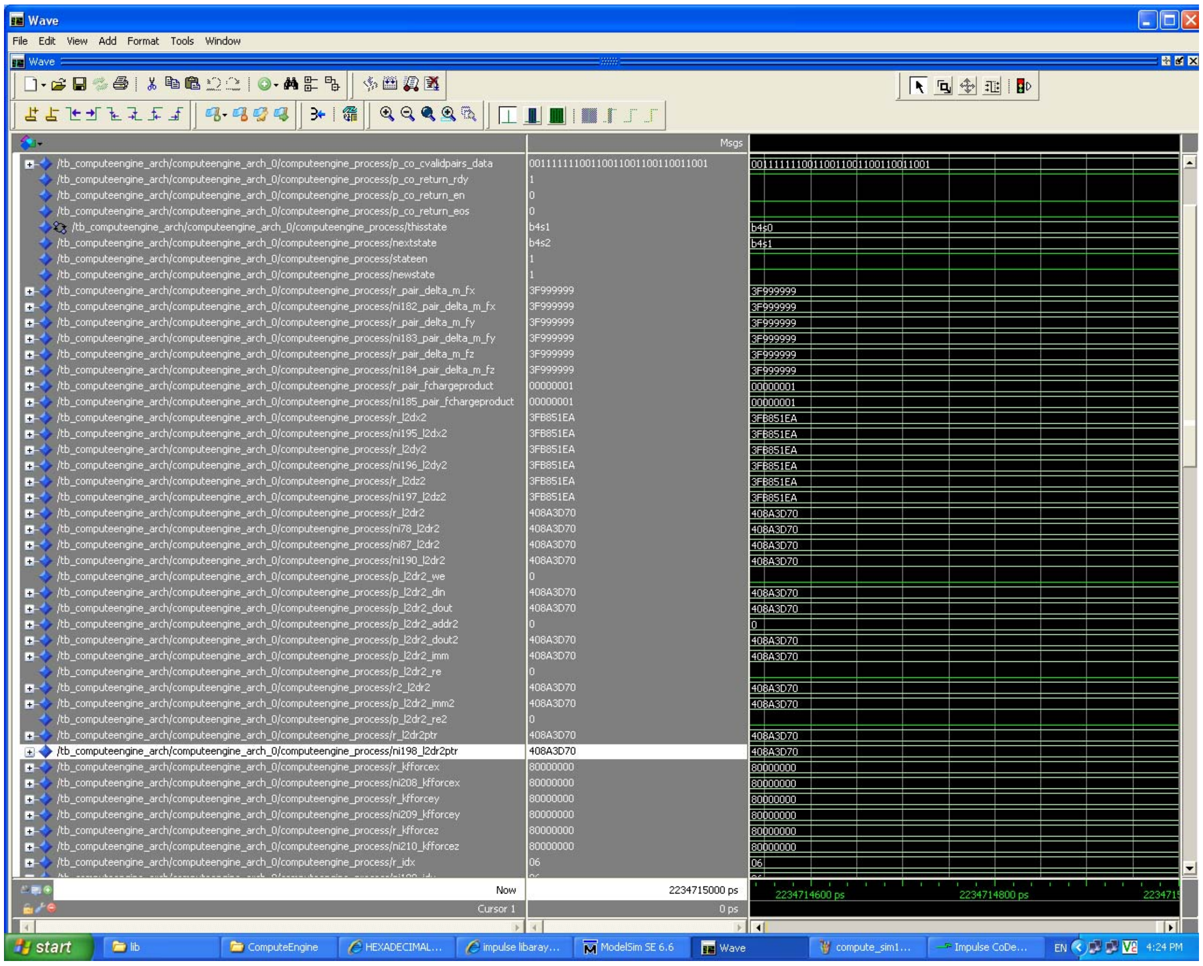
Disadvantages:

- A long learning process of the limitations of this tool
- No widely available technical support and discussion forum
- Programming Model does not support external memory via PLB bus

4.2 Future improvement and suggestion

Our Project is only a preliminary step in developing an MD simulator running on FPGA using MPI protocol. First, the MPI protocol is not added in at this point due to the tight schedule. Second, mapping the entire software algorithm is almost impossible on FPGA. The chip area is not large enough to hold all the functions. Moreover, the traditional memory model used in software does not works well in hardware. To make things even worse, control flow instructions can dramatically reduce area efficiency thus reduce overall system performance. Therefore, new programming paradigm in designing accelerators on FPGAs has to be devised.

Since Hao Jun is going to continue on this project with Prof. Chow during the summer, he could improve the design in the following aspects:



```
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_co_cvalidpairs_data   00111111100110011001100110011001
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_co_return_rdy         1
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_co_return_en          0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_co_return_eos         0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/thisstate               b4s1
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/nextstate               b4s2
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/stateen                 1
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/newstate                1
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_pair_delta_m_fx       3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni182_pair_delta_m_fx   3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_pair_delta_m_fy       3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni183_pair_delta_m_fy   3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_pair_delta_m_fz       3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni184_pair_delta_m_fz   3F999999
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_pair_fchargeproduct   00000001
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni185_pair_fchargeproduct 00000001
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_l2dx2                 3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni195_l2dx2             3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_l2dy2                 3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni196_l2dy2             3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_l2dz2                 3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni197_l2dz2             3FB851EA
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_l2dr2                 408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni78_l2dr2              408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni87_l2dr2              408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni190_l2dr2             408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_we              0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_din             408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_dout            408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_addr2           0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_dout2           408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_imm             408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_re              0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r2_l2dr2                408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_imm2            408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/p_l2dr2_re2             0
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_l2dr2ptr              408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni198_l2dr2ptr          408A3D70
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_kfforcex              80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni208_kfforcex          80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_kfforcey              80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni209_kfforcey          80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_kfforcez              80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/ni210_kfforcez          80000000
/tb_computeengine_arch/computeengine_arch_0/computeengine_process/r_idx                   06
Now        2234715000 ps
Cursor 1   0 ps
```

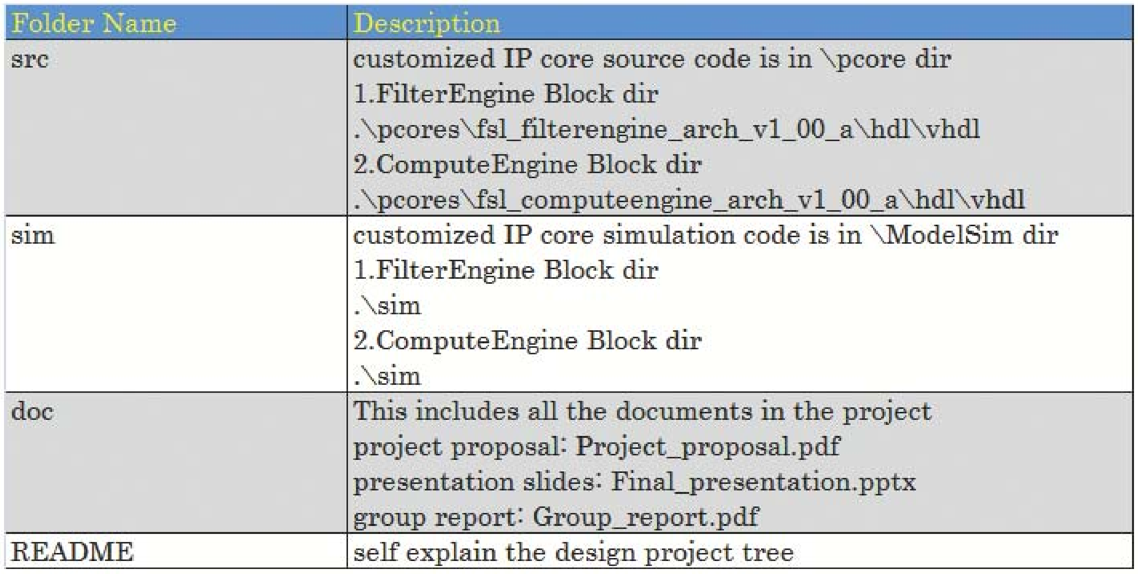
Figure 6: Compute Engine Modelsim Simulation

- implement a better algorithm to perform MD simulation on hardware with low number of DSP cores
- use MPI as the coherence protocol
- implement multiple hardware computing engines to improve the performance of the design

APPENDIX

A Description of Design Tree

The following table illustrates the content of each folder for the project. There is also a README.txt file in the the directory that provides similar information.



Folder Name	Description
src	customized IP core source code is in \pcore dir 1.FilterEngine Block dir .\pcores\fsl_filterengine_arch_v1_00_a\hdl\vhdl 2.ComputeEngine Block dir .\pcores\fsl_computeengine_arch_v1_00_a\hdl\vhdl
sim	customized IP core simulation code is in \ModelSim dir 1.FilterEngine Block dir .\sim 2.ComputeEngine Block dir .\sim
doc	This includes all the documents in the project project proposal: Project_proposal.pdf presentation slides: Final_presentation.pptx group report: Group_report.pdf
README	self explain the design project tree

Figure 7: Project Tree

B Reference

[1]. MicroBlaze_0 processor reference:
http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[2]. FSL data sheet:
http://www.xilinx.com/support/documentation/ip_documentation/fsl_v20.pdf

[3]. TFT controller data sheet:
http://www.xilinx.com/support/documentation/ip_documentation/xps_tft.pdf

[4]. Xilinx platform studio and EDK document:
http://www.xilinx.com/ise/embedded/edk_docs.htm

[5]. Impulse C co-developer reference website:
http://www.impulseaccelerated.com/