FPGA **Digital Track** Training

Final Project: Implementing 3x3 **Matrix multiplier** using different implementations

(**FSMD**, **NIOS II SoC**, and **ARM Cortex M0**)

Submitted To:

Dr. Ihab Adly

Submitted By:

1. Ahmed Adel Abd Elmonem
2. Shady Shokri Ramzy
3. Mostafa Hassanien Ahmed

Table of Contents

Table of Figures

Tables

1 FSMD

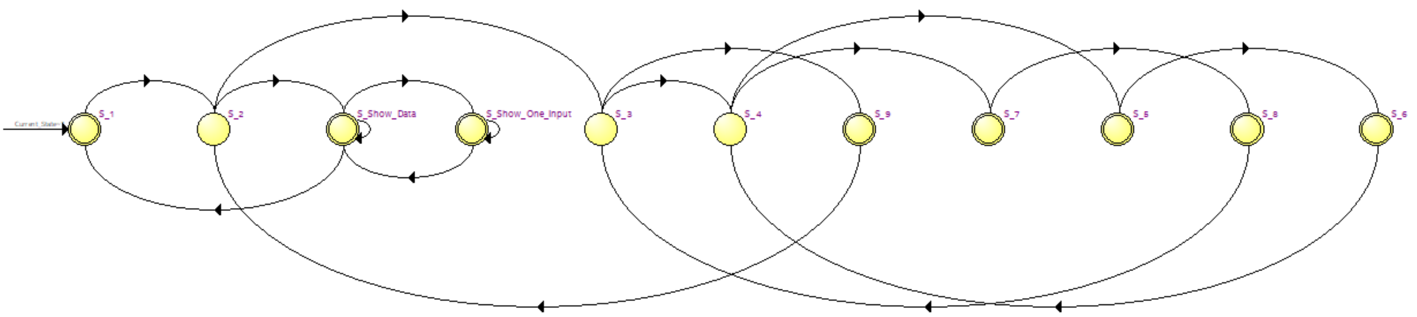

Figure 1: State Machine Diagram

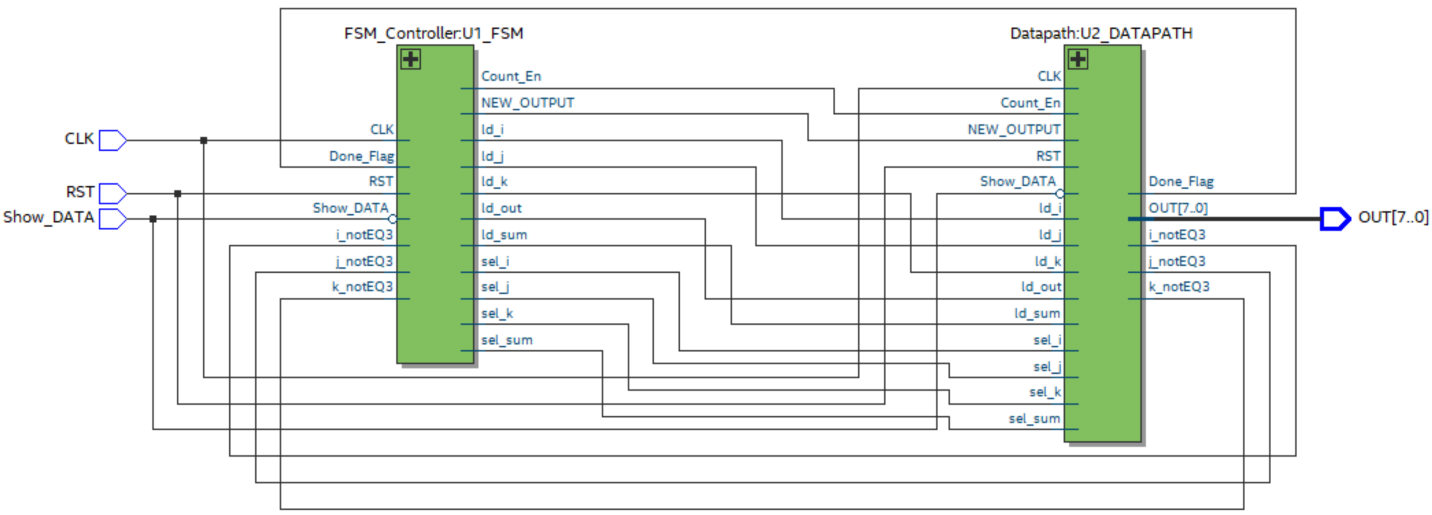

Figure 2: Top Level RTL Schematic

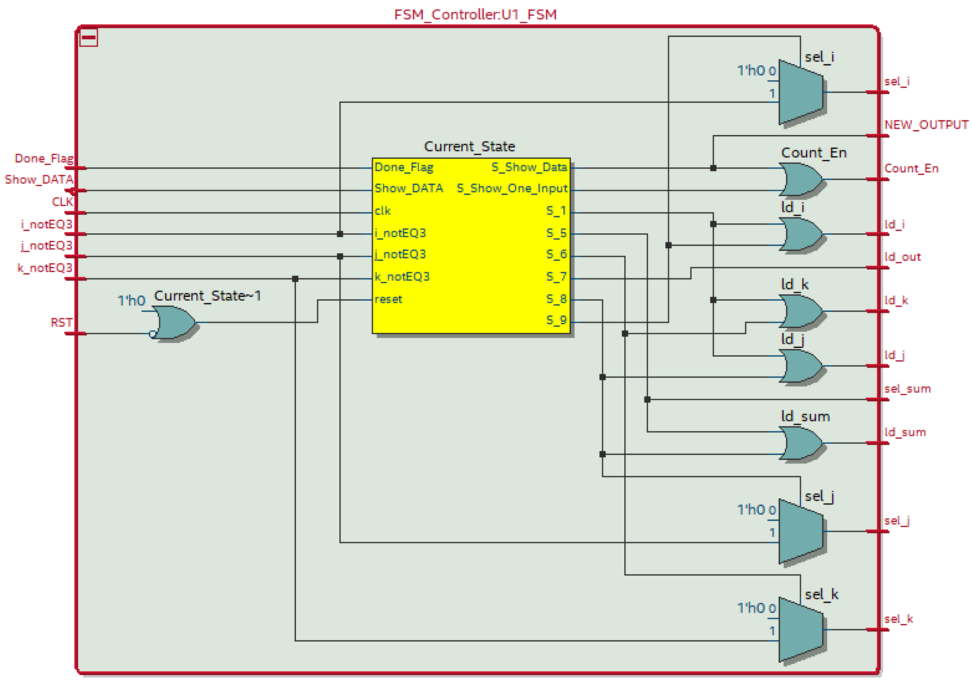

Figure 3: FSM RTL Schematic

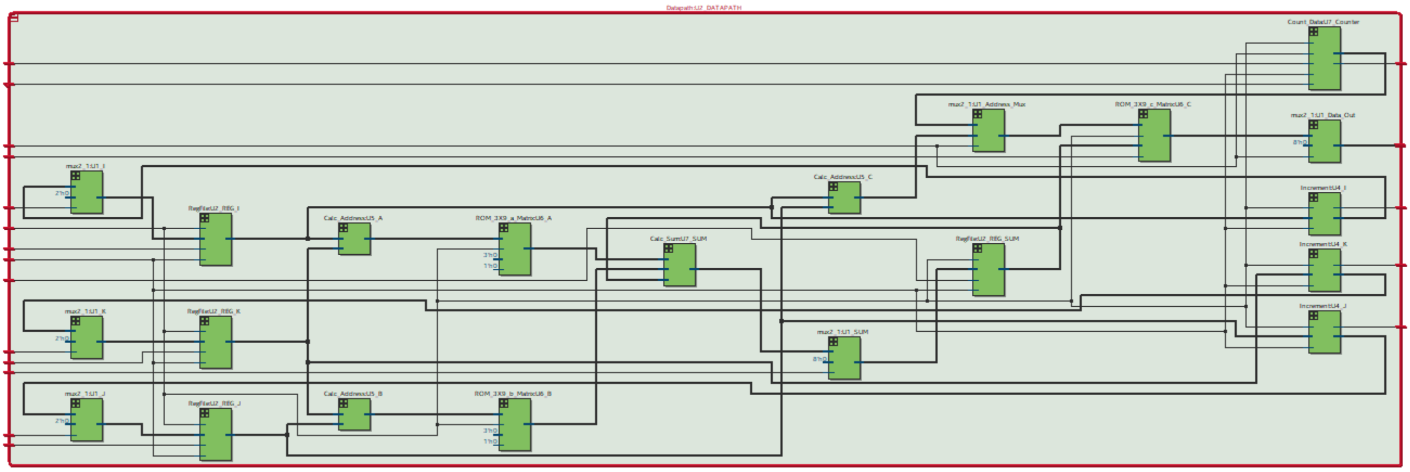

Figure 4: Datapath RTL Schematic

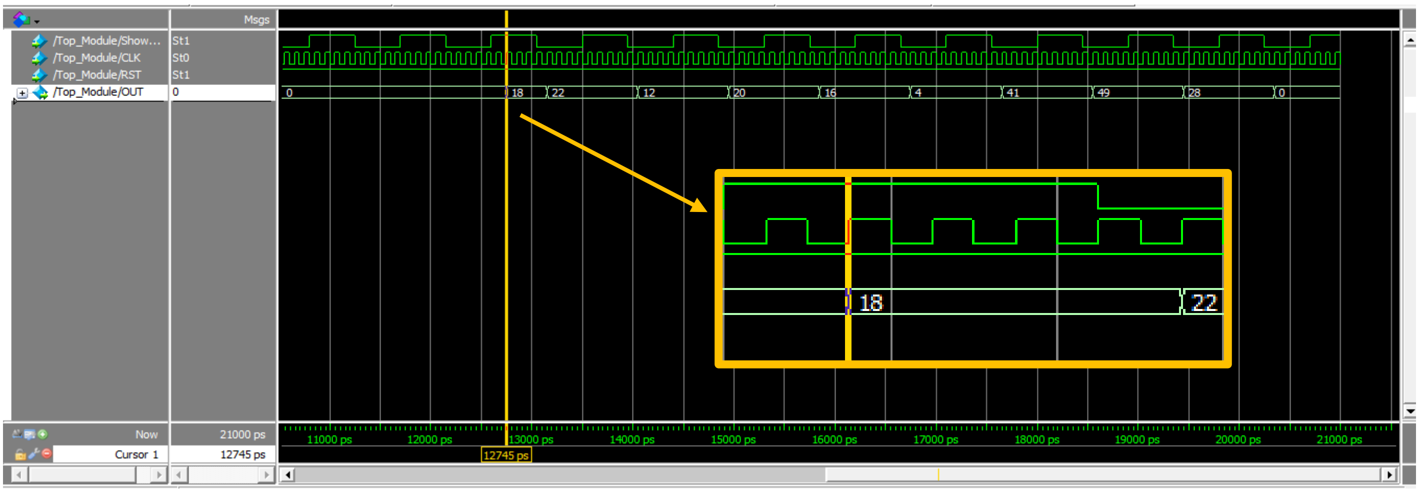

Figure 5: Functional Simulation

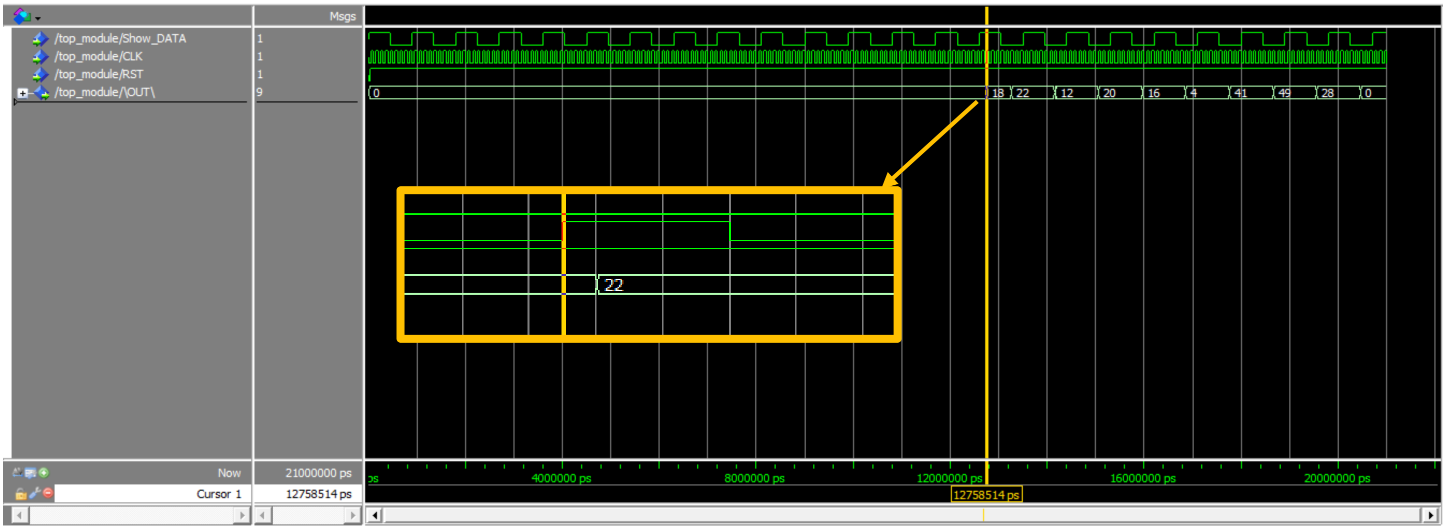

Figure 6: Timing Simulation

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Oct 04 21:11:49 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Top_Module
Top-level Entity Name	Top_Module
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	82 / 22,320 (< 1 %)
Total registers	39
Total pins	11 / 154 (7 %)
Total virtual pins	0
Total memory bits	128 / 608,256 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 7: FSMD Resources

Note:

- Only the output matrix RAM memory is inferred to BRAM resources of the FPGA. The other two input matrices RAM memories are uninferred due to inappropriate RAM size.
- Additionally, multipliers are uninferred to the embedded multiplier 9-bit elements due to inappropriate multiplier size.

2 NIOS II SoC

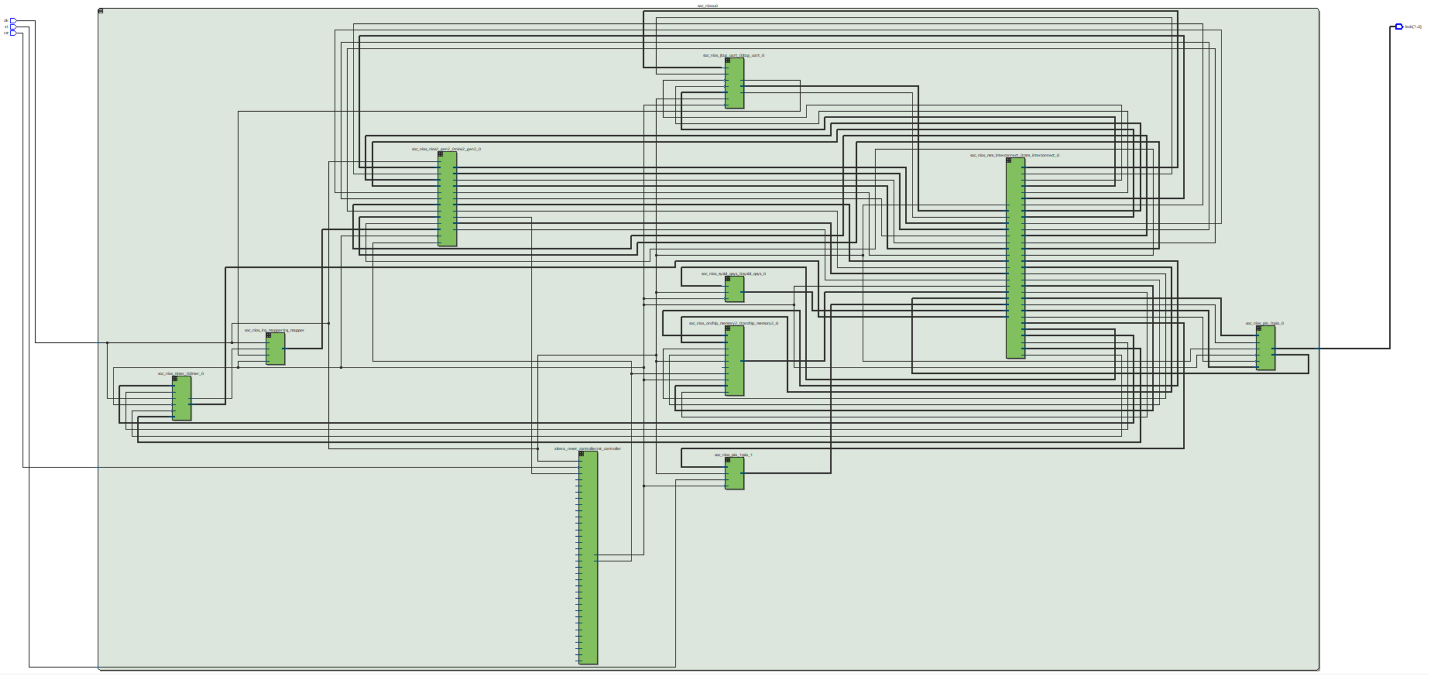

Figure 8: NIOS II Microcontroller Schematic

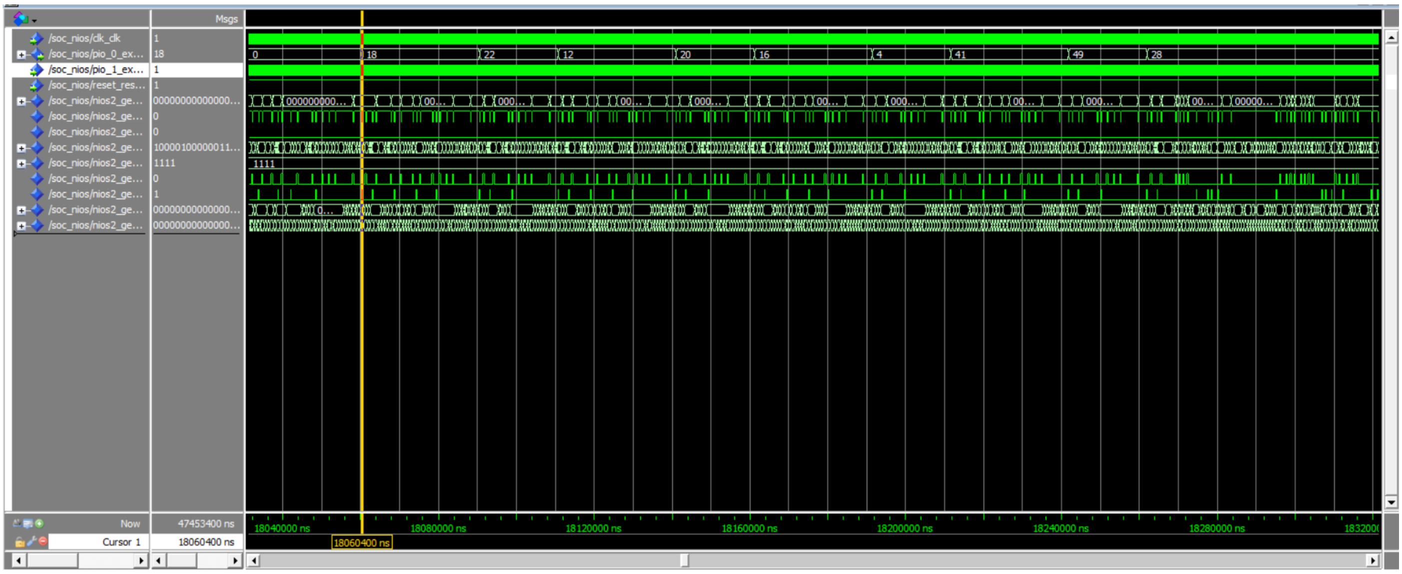

Figure 9: Functional Simulation

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Oct 04 21:25:57 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	soc_nios_fpga
Top-level Entity Name	soc_nios_fpga
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	1,876 / 22,320 (8 %)
Total registers	1046
Total pins	11 / 154 (7 %)
Total virtual pins	0
Total memory bits	404,480 / 608,256 (66 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 10: NIOS II SoC Resources

3 ARM Cortex M0

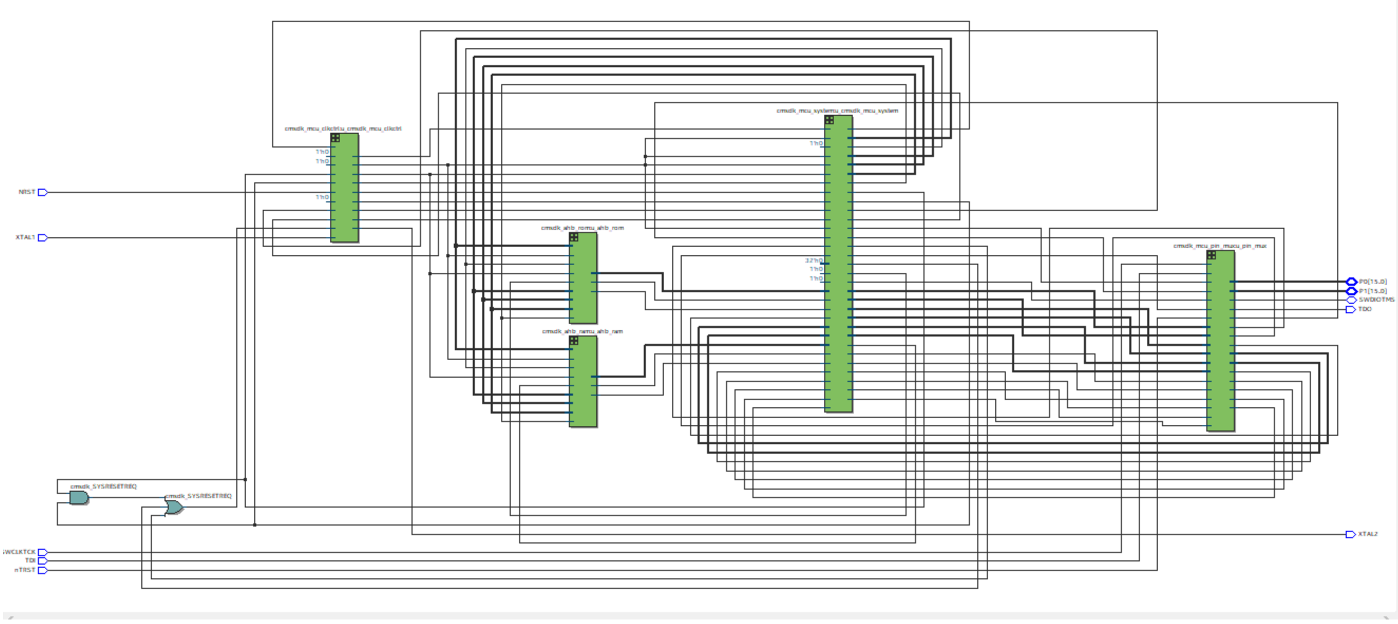

Figure 11: ARM Cortex M0 Schematic

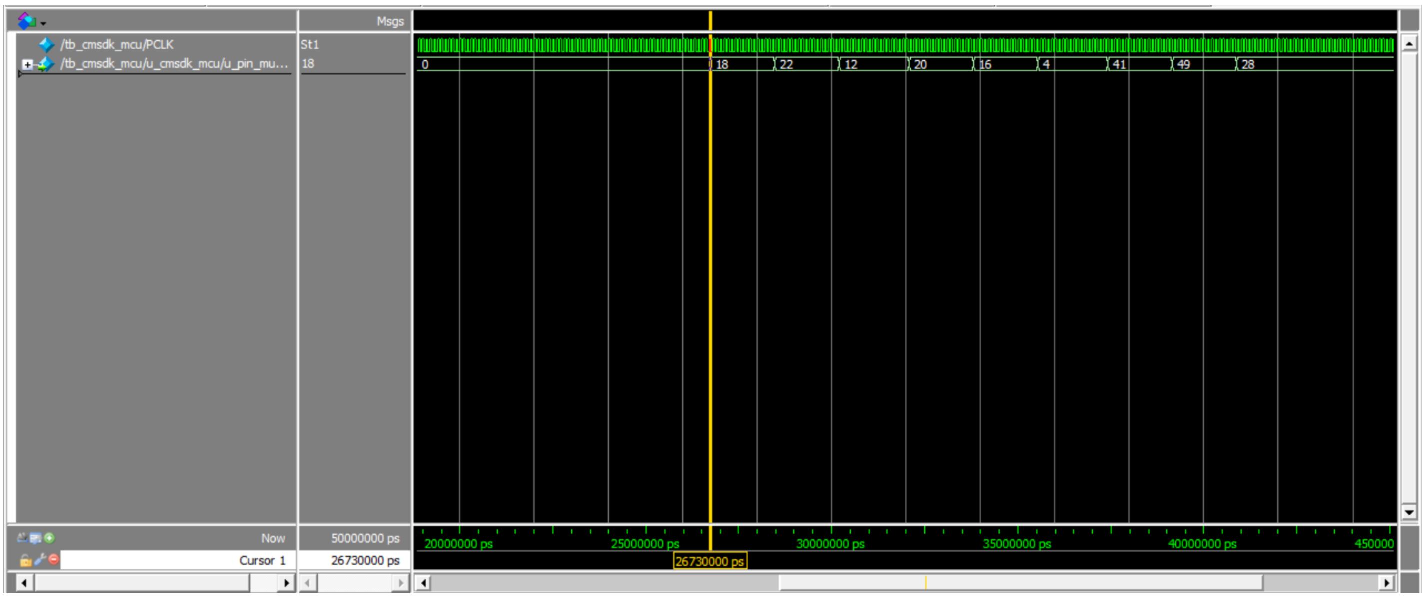

Figure 12: ARM Cortex M0 Functional Simulation

Flow Summary

<<Filter>>

Flow Status	Successful - Tue Oct 05 00:15:47 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cmsdk_mcu
Top-level Entity Name	cmsdk_mcu
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	18,711 / 22,320 (84 %)
Total registers	10614
Total pins	40 / 154 (26 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	6 / 132 (5 %)
Total PLLs	0 / 4 (0 %)

Figure 13: ARM Cortex M0 Resources

4 Observation

	FSMD	NIOS II	Cortex M0
Number of Cycles	128	180, 604	268
Total Logic Elements	82	1,876	18,711
Total Registers	39	1,046	10,614
Total Memory bits	128	404,480	0
Embedded Multiplier 9-bit elements	0	0	6

Table 1: Comparison between different implementations: FSMD, NIOS II, and Cortex M0

Obviously, FSMD is the fastest implementation regarding execution time followed by ARM Cortex M0. In contrast, NIOS II is the slowest implementation. Additionally, Cortex M0 uses a lot of resources and this why it has less execution time than NIOSS II processor.

	FSMD	NIOS Func	NIOS HW	Cortex Func	Cortex HW
Group 10	Done	Done	Done	Done	Done

5 Hardware Test Outputs

Figure 24: Hardware Test Output 1

Figure 35: Hardware Test Output 2

Figure 46: Hardware Test Output 3

Figure 57: Hardware Test Output 4

Figure 68: Hardware Test Output 5

Figure 79: Hardware Test Output 6

Figure 20: Hardware Test Output 7

Figure 21: Hardware Test Output 8

Figure 22: Hardware Test Output 9

6 Appendices

6.1 NIOS II C Code

```c
# include <system.h>
# include <altera_avalon_pio_regs.h>
#include <stdio.h>

int main (void){
    //inputing the two 3x3 matrices
    int a[9] = {1,2,3,5,0,1,2,3,7};
    int b[9] = {3,2,0,0,1,0,5,6,4};

    int c[9];

    int en = 1;

        int i;
        int j;
        int x;



        //multiplication loops
    for( i = 0;i < 3;i++){
        for( j = 0;j < 3;j++){
            c[3*i + j] = 0;
            for( x = 0;x < 3;x++){
                 c[3*i + j] = c[3*i + j] + (a[3*i + x] * b[j + 3*x]);
            }
        }
    }

    for ( int k = 0; k < 9; ++ k )
    {
      en = IORD_ALTERA_AVALON_PIO_DATA(PIO_1_BASE);
      while(en)
      {
          en = IORD_ALTERA_AVALON_PIO_DATA(PIO_1_BASE);
      }
      while ( en == 0 )
      {
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE, c[k] );
        en = IORD_ALTERA_AVALON_PIO_DATA(PIO_1_BASE);
      }
    }
    return 0;
}
```

6.2 ARM Cortex M0 C Code

```c
#ifdef CORTEX_M0

#include "CMSDK_CM0.h"
#include "core_cm0.h"
#endif

#ifdef CORTEX_M0PLUS
#include "CMSDK_CM0plus.h"
#include "core_cm0plus.h"
#endif

#include <stdio.h>

int main (void){
    //inputing the two 3x3 matrices
    int a[9] = {1,2,3,5,0,1,2,3,7};
    int b[9] = {3,2,0,0,1,0,5,6,4};

    int c[9];

        int i;
        int j;
        int x;
         int o;

        //initializing GPIO0
        CMSDK_GPIO0->OUTENABLESET = 0xFFFF;
        CMSDK_GPIO0->ALTFUNCCLR = 0xFFFF;

        //multiplication loops
    for( i = 0;i < 3;i++){
        for( j = 0;j < 3;j++){
            c[3*i + j] = 0;
            for( x = 0;x < 3;x++){
                 c[3*i + j] = c[3*i + j] + (a[3*i + x] * b[j + 3*x]);
            }
 //outputing value of the array element on LEDs connected to GPIO0 first 9 pins
                        CMSDK_GPIO0->DATA = c[3*i + j];
           for( o = 0;o < 50000000;o++){}
        }
    }

    return 0;
}
```