ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG
----------o0o----------

BÁO CÁO BÀI TẬP VỀ NHÀ LẦN 4

THIẾT KẾ

SOC

SVTH : Trần Hồng Sơn

MSSV: 20200331

I. YÊU CẦU

Bài tập tích hợp lọc FIR vào hệ thống SoC:

Yêu cầu:

- Phát triển/tích hợp Master Write và Master Read (DMA) vào lọc FIR vào hệ thống SoC.
- File C code trên Eclipse để chạy phần mềm.
- Kết quả mô phỏng cả hệ thống, phân tích và giải thích dạng sóng trong mô phỏng.

Nộp file nén bao gồm:

- Code verilog.
- File báo cáo bằng pdf/word.
- File C code.

II. MÔ TẢ

2.1 Yêu cầu kỹ thuật chung

- Cấu hình bộ DMA để đọc và ghi giá trị.
- Mô đun FIR từ bài trước là 8 hệ số lọc, giá trị dữ liệu và hệ số là 8 bit
- Dùng FIFO để kiểm soát đọc ghi giữa Write / Read Master và FIR.

2.2 Cấu trúc

DMA FIR

- Master Write/Read để đọc ghi từ bộ nhớ ngoài (On-chip Memory)
- Kích thước đường dữ liệu vào ra là 32 bit

FIFO

- 2 bộ FIFO là FIFOi (Từ Master Read vào FIR) và FIFOo (FIR ra Master Write)

FIR_Wrapper:

- FIR_Core
  - ✓ Thực hiện chức năng lọc FIR.

- ✓ Có thêm tín hiệu Wait để chờ đến khi đưa vào đủ các hệ số lọc mới bắt đầu tính toán.
- ✓ Thêm tín hiệu Write để chỉ thực hiện dịch giá trị khi đọc vào giá trị mới (khi Write lên 1).

- FIR_Csr: chứa các tín hiện điều khiển
  - ✓ ChipSelect: để bật tắt việc ghi và đọc giá trị từ ReadData, WriteData.
  - ✓ WriteData: thanh ghi Avalon giá trị các hệ lọc đưa vào. Do có 8 giá trị 8 bit. Mà WriteData tối đa 32 bit nên chia ra đọc vào 2 lần, mỗi lần 4 hệ số.
  - ✓ Write: điều khiển việc viết giá trị từ thanh ghi WriteData vào module Csr.
  - ✓ Read: điều khiển việc đọc kết quả từ Csr ra thanh ghi ReadData.
  - ✓ Yn: giá trị được FIR_Core tính được
  - ✓ ReadData: thanh ghi chứa các kết quả ra được tạo từ Yn.
  - ✓ Cuối cùng là các hệ số lọc đưa vào FIR_Core.
  - ✓ Sử dụng biến Address điều khiển việc ghi dữ liệu từ thanh ghi WriteData. Khi Address là 2'b00 thì ghi H3-H0, Address là 2'b01 thì ghi vào H7-H4, khi Address là 2'b10 thì ghi vào giá trị X (ở cả 2 trường hợp ghi giá trị hệ số lọc H thì cho biến Wait lên 1 để chờ đến khi ghi hoàn tất).

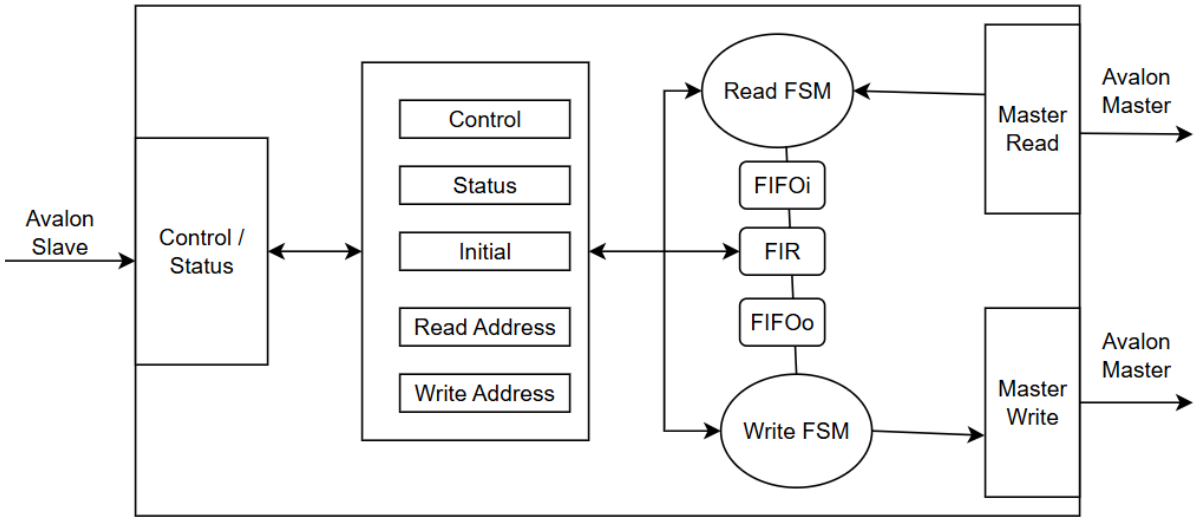
III. THIẾT KẾ

3.1 Các thanh ghi kiểm soát

Địa chỉ	Tên	Truy cập	Mô tả
0	Control	R/W	Thanh ghi điều khiển - Bit 0: Start DMA - Bit 1: fifoi_clear: Xóa data trong FIFOI - Bit 4: fifoo_clear: Xóa data trong FIFOO - Bit 9:8 : Địa chỉ thiết lập cho FIR. 0 thì ghi vào H0-H3, 1 ghi H4-H7, 2 thì ghi X
1	Initial	R/W	Thanh ghi chứa giá trị khởi tạo (hoặc thiết lập). Chứa giá trị của các hệ số lọc (H)
2	Read Address	R/W	Địa chỉ bắt đầu đọc
3	Write Address	R/W	Địa chỉ bắt đầu ghi
4	Length	R/W	Độ dài (byte)

5	Status	R	Thanh ghi trạng thái Bit 0: DMA Start: 1 thì bắt đầu, 0 thì ở IDLE Bit 1: Fifoi Clear Bit 2: Fifoi Empty Bit 3: Fifoi Full Bit 4: Fifoo Clear Bit 5: Fifoo Empty Bit 6: Fifoo Full Bit 7: DMA done lên 1 khi hoàn thành DMA Bit 9:8 địa chỉ cấu hình cho FIR

3.2 Sơ đồ hệ thống



3.3 Mô tả máy trạng thái hữu hạn

- Với máy trạng thái đọc kiểm tra reset, nếu reset thì quay về trạng thái khởi tạo, không thì chuyển trạng thái.

- Có 4 trạng thái:

Trang thái khởi tạo (r_idle) khởi động khi có start_trigger (bắt đầu DMA) và bắt đầu đọc địa chỉ khởi tạo (address_read_fetch = 1 thì địa chỉ đọc là địa đầu, là 0 thì cộng dồn địa chỉ đọc có sẵn thêm 4 và đọc).

Trạng thái r_fifo_read khởi tạo ghi giá trị từ FIR vào FIFOo

Trạng thái kế r_request thiết lập địa chỉ đọc, thực hiện yêu cầu đọc và kiểm tra nếu fifo đang đầy hoặc chờ thì lặp lại, khi đọc thành công thì qua trạng thái r_incr.

Trạng thái r_incr kiểm tra nếu đã đọc xong thì qua trạng thái khởi tạo, ko thì qua trạng thái r_fifo_write để lặp lại quá trình đưa dữ liệu từ FIR qua FIFO đến ReadMaster.

- Máy trạng thái ghi: Cũng kiểm tra nếu reset thì qua giá trị khởi tạo, không thì chuyển trạng thái

- Có 4 trạng thái là

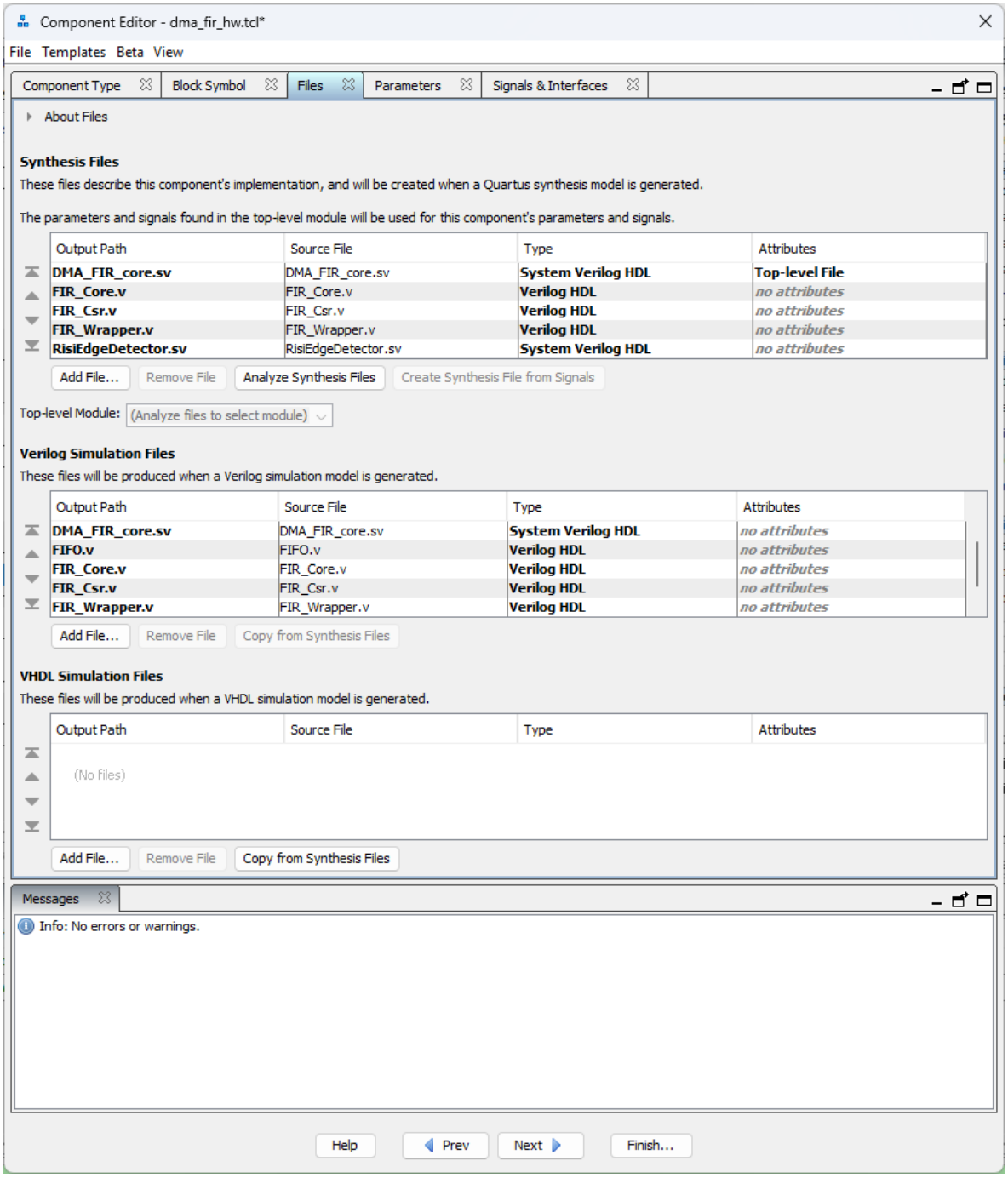
w_ilde : nếu fifoi không đầy thì cho address_write_fetch là 1 để địa chỉ ghi là địa chỉ bắt đầu, sau đó chuyển sang w_fifo_read

w_fifo_read: kiểm tra nếu fifoi có dữ liệu thì cho fifoi_read là 1 để đọc từ fifoi đưa vào FIR, sau đó sang trạng thái kế w_request

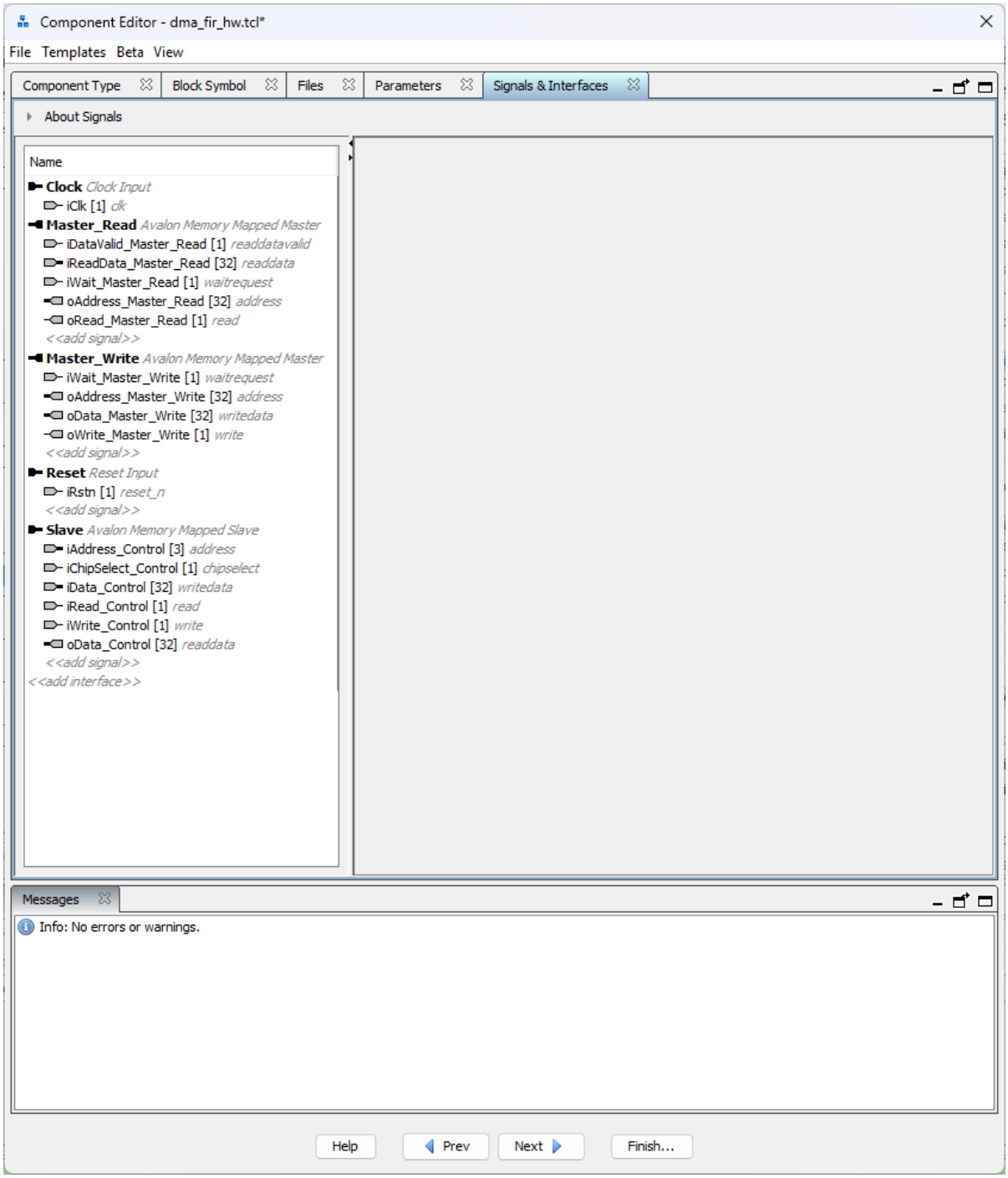
w_request: bật bit điều khiển ghi từ Master (oWrite_Master_Write), gán địa chỉ ghi và kiểm tra nếu đang chờ ghi hoặc fifoi đầy thì chờ, tới khi xong thì bật bit tăng địa chỉ ghi và chuyển sang w_incr

w_incr: kiểm tra nếu đã ghi xong tất cả dữ liệu thì cho done_trigger lên 1 để xác định biến done (hoàn thàn DMA), không thì quay lại w_fifo_read.

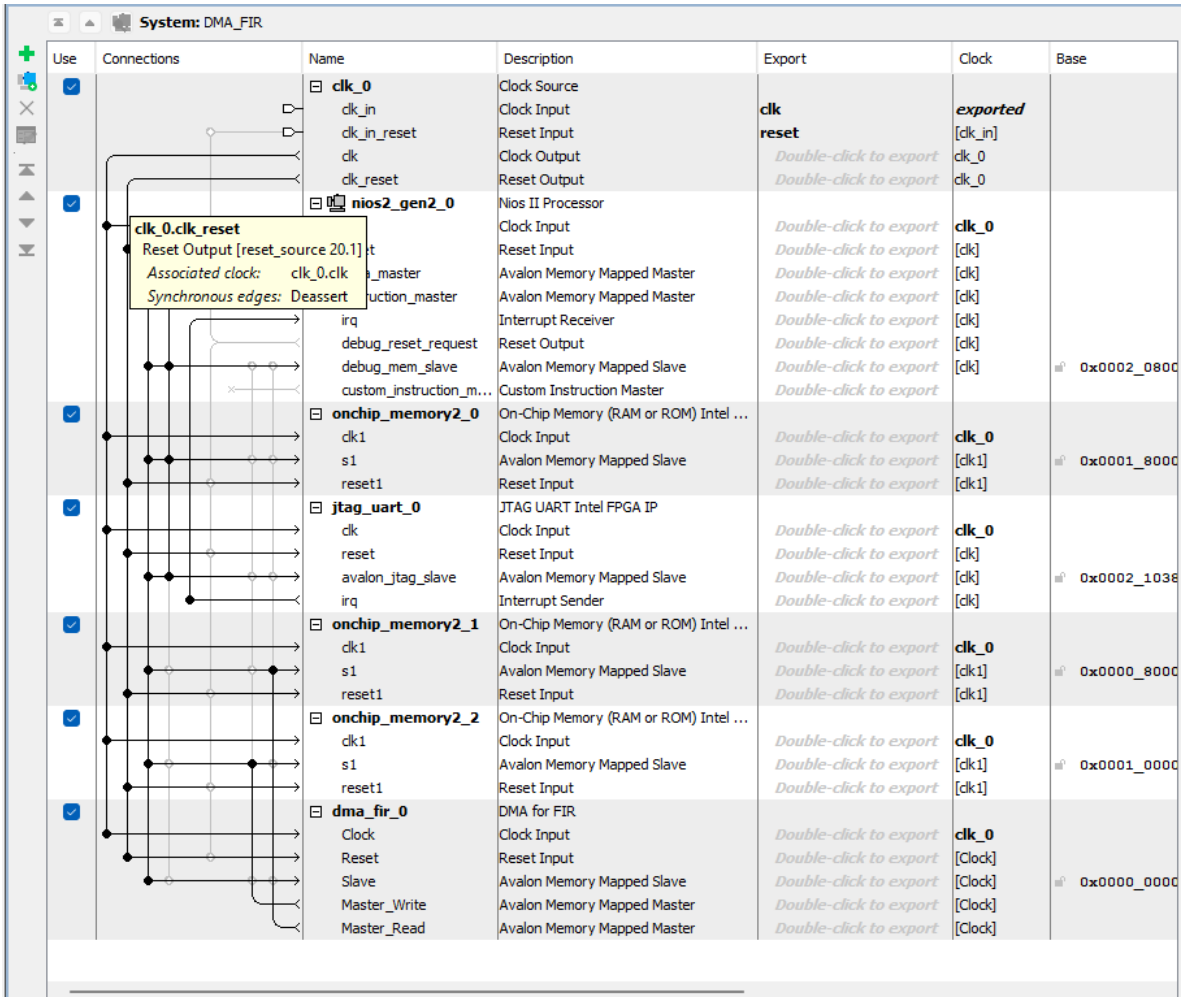
3.3 Mô hình system



Các file dùng để chạy và giả lập



Thiết lập phương thức giao tiếp cho các tín hiệu



Cách thành phần và cách kết nối của hệ thống

IV. XÂY DỰNG PHẦN MỀM VÀ MÔ PHỎNG

4.1 Mã C kiểm tra

```
#include <stdio.h>
#include <math.h>
#include "system.h"
#include <io.h>
#include "sys/alt_stdio.h"

int init_mem1(){
    volatile int* mem_ptr = (int *) ONCHIP_MEMORY2_1_BASE;
    unsigned int inputs[] = {0x11, 0x13, 0x0E, 0x0B, 0x0F, 0x11, 0x0E, 0x17,
            0x0E, 0x16, 0x14,0x11, 0x19, 0x10, 0x0D, 0x04, 0x02};
    int numInputs = sizeof(inputs) / sizeof(inputs[0]);
    int i = 0;
    for (i = 0; i < numInputs; i++){
        *(mem_ptr + i) = inputs[i] ;
    }
    return numInputs ;
}
int main() {
    volatile int* mem_ptr = (int *) ONCHIP_MEMORY2_1_BASE;
    volatile int* dmafir_ptr = (int *) DMA_FIR_0_BASE;
    volatile int* result_ptr = (int *) ONCHIP_MEMORY2_2_BASE;
    int length = init_mem1();

      *(dmafir_ptr + 2) = (volatile int) ONCHIP_MEMORY2_1_BASE;
      *(dmafir_ptr + 3) = ONCHIP_MEMORY2_2_BASE;
      *(dmafir_ptr + 4) = length ;
      *(dmafir_ptr + 0) = 0; //Not start DMA, FIRadd = 0
      *(dmafir_ptr + 1) = 0x040E0C08;
      *(dmafir_ptr + 0) = 0x100;    //Not Start DMA, FIRadd = 1
      *(dmafir_ptr + 1) = 0x0B110D06;
      *(dmafir_ptr + 0) = 0x201;  //Start DMA, FIRadd = 10;
      while (1){
          int status = *(dmafir_ptr + 5);
          if ((status & 0x800) == 0x800){
              printf("Dma Done\n");
              for (int i = 0; i < length; i++){
                  printf ("mem1[%d] = %d\n", i, *(mem_ptr + i));
                  printf ("mem2[%d] = %d\n", i, *(result_ptr + i));
              }
              break;
          }
      }
        printf("Hello from System");
    return 0;
}
```

Khởi tạo các giá trị dữ liệu và trả numInputs là số data (length)

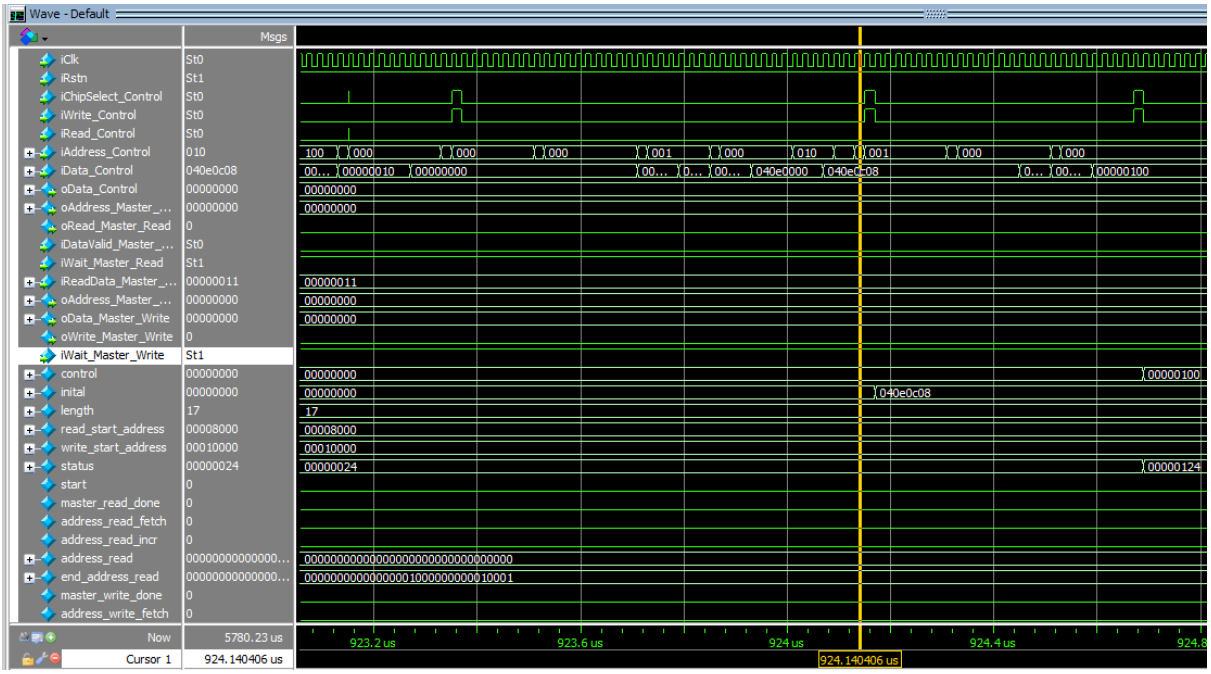
Ghi vào đĩa chỉ bắt đầu đọc, ghi và length

Chưa bắt đầu DMA và đưa vào FIR add để thiết lập

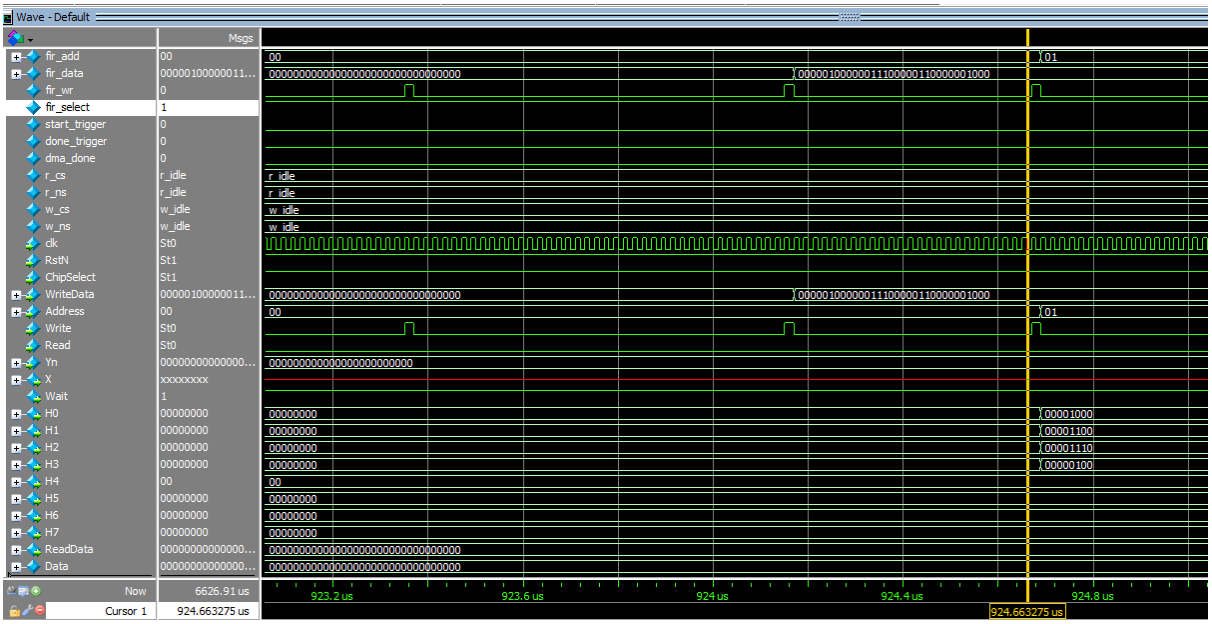
Đọc thanh status (Offset là 5) và kiểm bit thứ 8 để xem đã hoàn thành DMA chưa. Nếu rồi ngừng và xuất kết quả

4.2 Dạng sóng mô phỏng

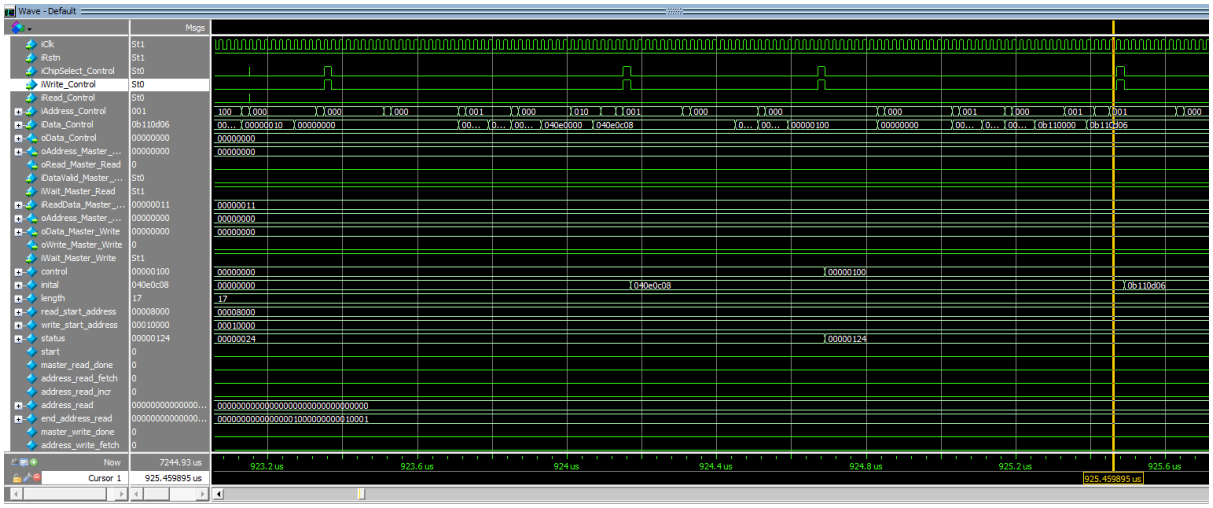
Khi iChipSelect_Control, iWrite_Control là 1, iAddress_Control là 1 thì ghi tổ hợp giá trị 32 bit H0-H3 vào initial.

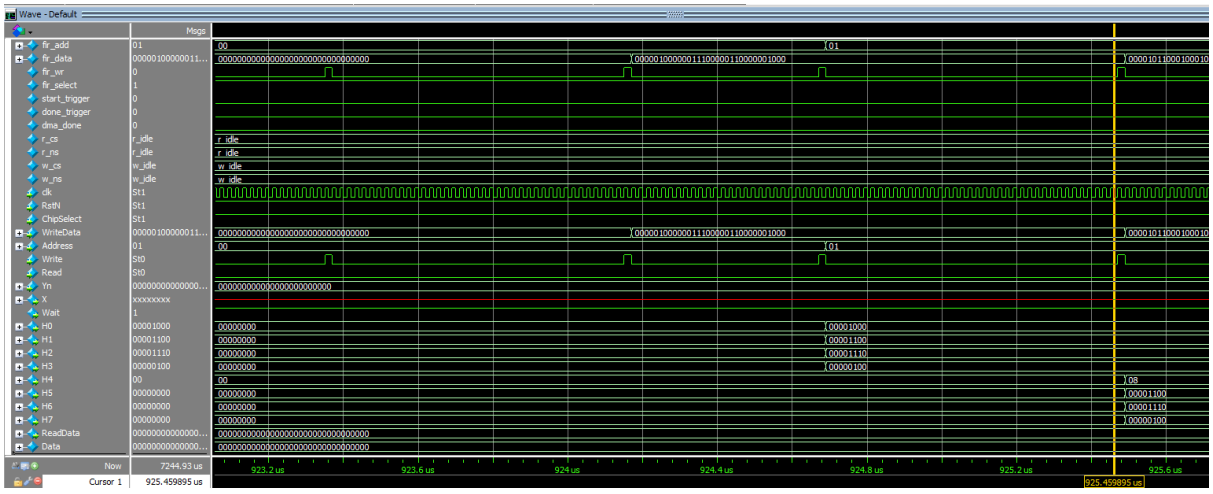


Khi fir_add đang là 0, ngay sau đó là thiết lập chi fir_select (ChipSelect) và fir_wr (Write) để ghi giá trị từ initial ngay ở xung lên chu kỳ clock kế.



Tương tự trên nhưng thiết lập fir_add là 01 để ghi vào từ H4-H7


Wave - Default
Msgs
iClk
iRstn
iChipSelect_Control
iWrite_Control
iRead_Control
iAddress_Control
iData_Control
oData_Control
oAddress_Master_...
oRead_Master_Read
iDataValid_Master_...
iWait_Master_Read
iReadData_Master_...
oAddress_Master_...
oData_Master_Write
oWrite_Master_Write
iWait_Master_Write
control
inital
length
read_start_address
write_start_address
status
start
master_read_done
address_read_fetch
address_read_incr
address_read
end_address_read
master_write_done
address_write_fetch
Now
7244.93 us
Cursor 1
925.459895 us
923.2 us
923.6 us
924 us
924.4 us
924.8 us
925.2 us
925.6 us


Wave - Default
Msgs
fir_add
fir_data
fir_wr
fir_select
start_trigger
done_trigger
dma_done
r_cs
r_ns
w_cs
w_ns
clk
RstN
ChipSelect
WriteData
Address
Write
Read
Yn
X
Wait
H0
H1
H2
H3
H4
H5
H6
H7
ReadData
Data
Now
7244.93 us
Cursor 1
925.459895 us
923.2 us
923.6 us
924 us
924.4 us
924.8 us
925.2 us
925.6 us

-HẾT--