Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

PART I Objective Design SRAM USING VHDL

An SRAM retains data bits in its memory as long as power is being supplied to the device. You will create a static random-access memory (SRAM) using D-flip flops. You will be able to store information to different locations within the SRAM and read from it.

You should be familiar with the operation and behavior of D-flip flops before completing the SRAM. As a refresher, D-flip flops will be reviewed.

NOTE: If you already know how to create a D-flip flop, skip the review and move onto creating the Tri-State Buffer.

What is a Flip Flop?

A flip flop is a TWO STATE device (STATE depends on its current input and past sequence of inputs) that changes its STATE upon a change in the clock signal (rising or falling edge of the clock). This means that the device is changing state at a negative or positive edge of the clock signal.

D Flip Flop

A single D Flip Flops stores 1 bit of information upon the rising edge of a clock. In the following image the clock is alternating between 0 and 1 during a certain period of time. We will be evaluating what happens to Q with respect to D when the clock is 1, 0, rising edge or falling edge.

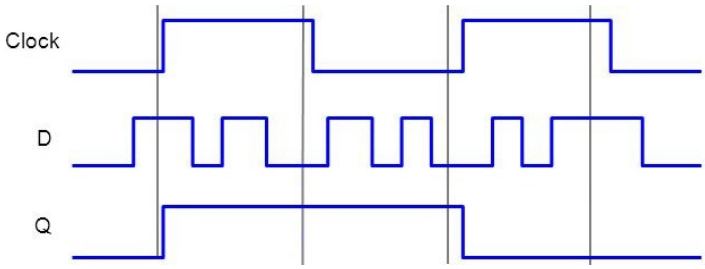


Figure 1: Timing diagram of D-flip flop.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

When the clock is 0 and D is 0 Q is whatever it previously was because Q is only affected by a rising edge of the Clock. When D is 1 and the clock is a rising edge Q will become 1. Although D is changing between 0 and 1, Q stays the same because again Q is only affected by D when there is a rising edge. In order to better illustrate figure 1 the following is a truth table explaining the different possible combinations between the clock signal, Input D and the stored value of Q.

Clock Signal	Input D	Value of Q	Description
Equal to 1	X	X	The value of D does not affect the value of Q because the clock value is 1.
Equal to 0	X	X	The value of D does not affect the value of Q because the clock value is 0.
Rising Edge	0	0	The vlaue of Q is reset to 0 regardless of what its previous value was.
Rising Edge	1	1	The value of Q is set to 1 regardless of what its prevoius value was.
Falling Edge	X	X	The value of D does not affect Q because clock is falling edge.
Falling Edge	X	X	The value of D does not affect Q because clock is falling edge.

Figure 2: Trnsition table for D Flip Flop.

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

```
--STATIC RAM CELL
library ieee;
use ieee.std_logic_1164.all;

entity SRAM1 is
        port(I, SEL, WR : in std_logic;
                    O : out std_logic);
end SRAM1;

architecture arch of SRAM1 is
        component tristate_buffer
                port(a, enable : in std_logic;
                            c : out std_logic);
        end component;

        signal D0 : std_logic;
        signal A : std_logic;

        begin
                A <= SEL and WR;

                --D Flip Flop
                process(A)
                begin
                        if rising_edge(A) then D0 <= I
                        end if;
                end process;

                TRI : tristate_buffer
                        port map (D0, SEL, O);

end arch;
```

A D-flip flop is essentially a one cell static RAM capable of storing 1 bit of information EXCEPT the SRAM utilizes a tri-state buffer.

The function *rising_edge* is a clock event detector of a *std_logic* or *std_ulogic* signal representative of the clock. It returns true when the signal changes from a low value (‘0’) to a high value (‘1’).

Figure 3: VHDL code of a D-flip flop

NOTE: If you already know how to create a Tri-State buffer, skip its review and move onto creating the SRAM.

Tri-state Logic buffer

You will need a Tri-State buffer for the SRAM. We want to control the flow of data going in and out of the circuit.

A tri-state buffer is a useful device that allows us to control when current passes through the device and when it doesn’t. It returns the value of the input when current is passed into the device: when its selector value is 1. When its selector value is 0, nothing is outputted.

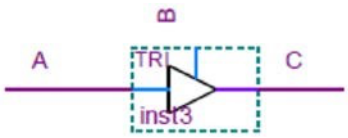


Figure 4: Diagram of Tri-State Buffer. A is the input that is sent to C only when B is ‘1’. B is this case is the selector.

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Input		Output
A	B	C
0	1	0
1		1
X	0	Z

Figure 5: Truth table of Tri-state Buffer.

```
library ieee;
use ieee.std_logic_1164.all;

entity tristate_buffer is
        port(a, enable : in std_logic;
                    c : out std_logic);
end tristate_buffer;

architecture arch of tristate_buffer is

        begin
                process (a, enable)
                        begin
                                if enable = '1' then c <= a;
                                elsif enable = '0' then c <= 'Z';
                                end if;
                end process;

end arch;
```

Figure 6: VHDL code for Tri-State buffer that you can use.

We will be using a tri-state buffer to read from the SRAM.

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

SRAM

Now that you understand how to store 1-bit of information, extend it to 4 bits.

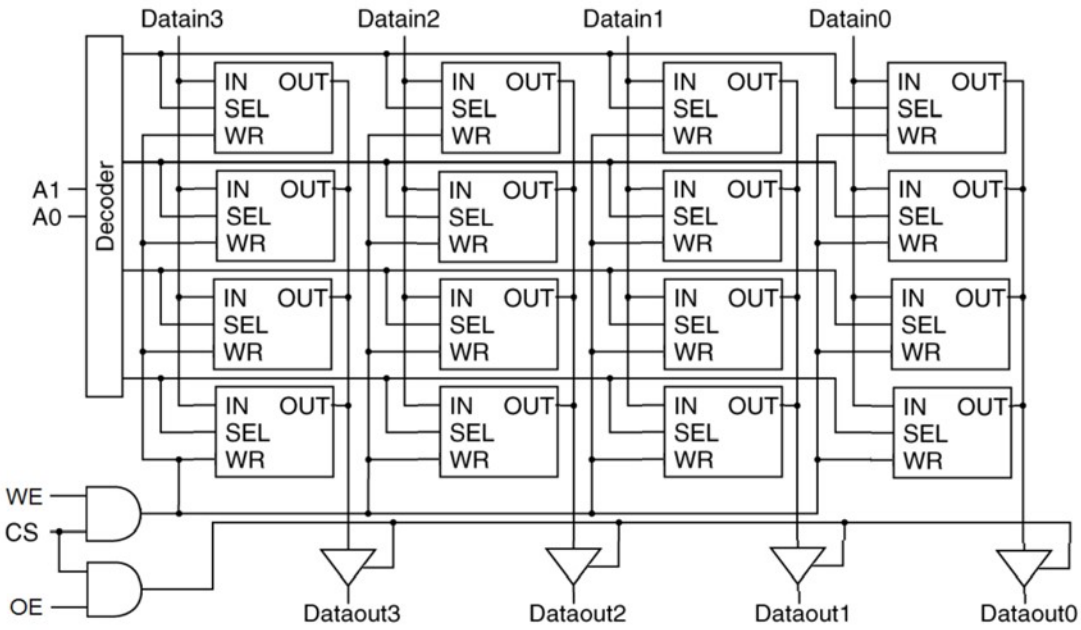


Figure 7: Diagram of 4x4 SRAM. 4 addressable locations each capable of storing 4 bits of information, a total of 16 static-RAM cells. The output of each column connects to the same output pin without the use of a multiplexer to separate the lines. As long as the tristate buffer cuts off the output of the other cells, the one cell that is on will be the only output that will be connected to the output pin.

The SRAM has several outputs:

- DataIn[3..0]: This is the input that will be written into the SRAM at a certain address.
- A[1..0]: This is the address input. This input determines which row of SRAM Cells will be turned on to write or read data. A standard 2-to-4 decoder is used to decode the address input and control the SRAM cells. -- **YOU WILL MAKE THIS A[3..0]**
- WE: The Write Enable input. This input tells the SRAM to write to the cells when it is high. If it is low, the chip will not allow any data to be written to any cell.
- OE: The Output Enable input. When this input is high, it will allow the SRAM to output the data at the chosen address. Otherwise, the tristate buffer will cut the output of the SRAM off.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

- CS: The Chip Select input. This input, when low, prevents any output from the SRAM and prevents any writing. It turns the SRAM off. Set this high to turn the SRAM on.

In this diagram, there is a 2 to 4 decoder used to select between 4 addresses.

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Figure 8: Truth table of 2 to 4 Decoder.

The decoder selects the address to read from or write to. You will need to extend this design to 16 4 bit addresses.

LPM RAM 1-PORT

The final product that you will be using is an LPM RAM 1-PORT. It serves the same purpose as the SRAM you are required design on your own, but it is provided to you by the LPM module. The following are the steps on how to properly create and use it.

To open the menu go to tools and then select Ip catalog.

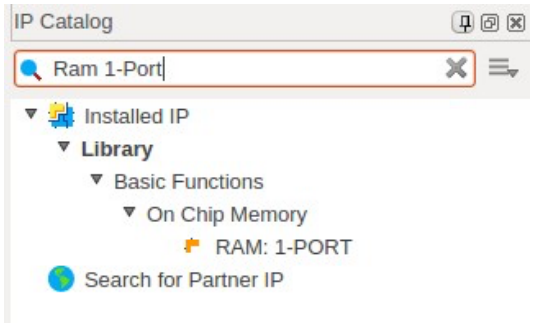


Figure 9: Dialogue window searching for RAM 1-Port design.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Figure 10: Make sure that when you are naming the file you select for it to be written in VHDL and not Verilog.

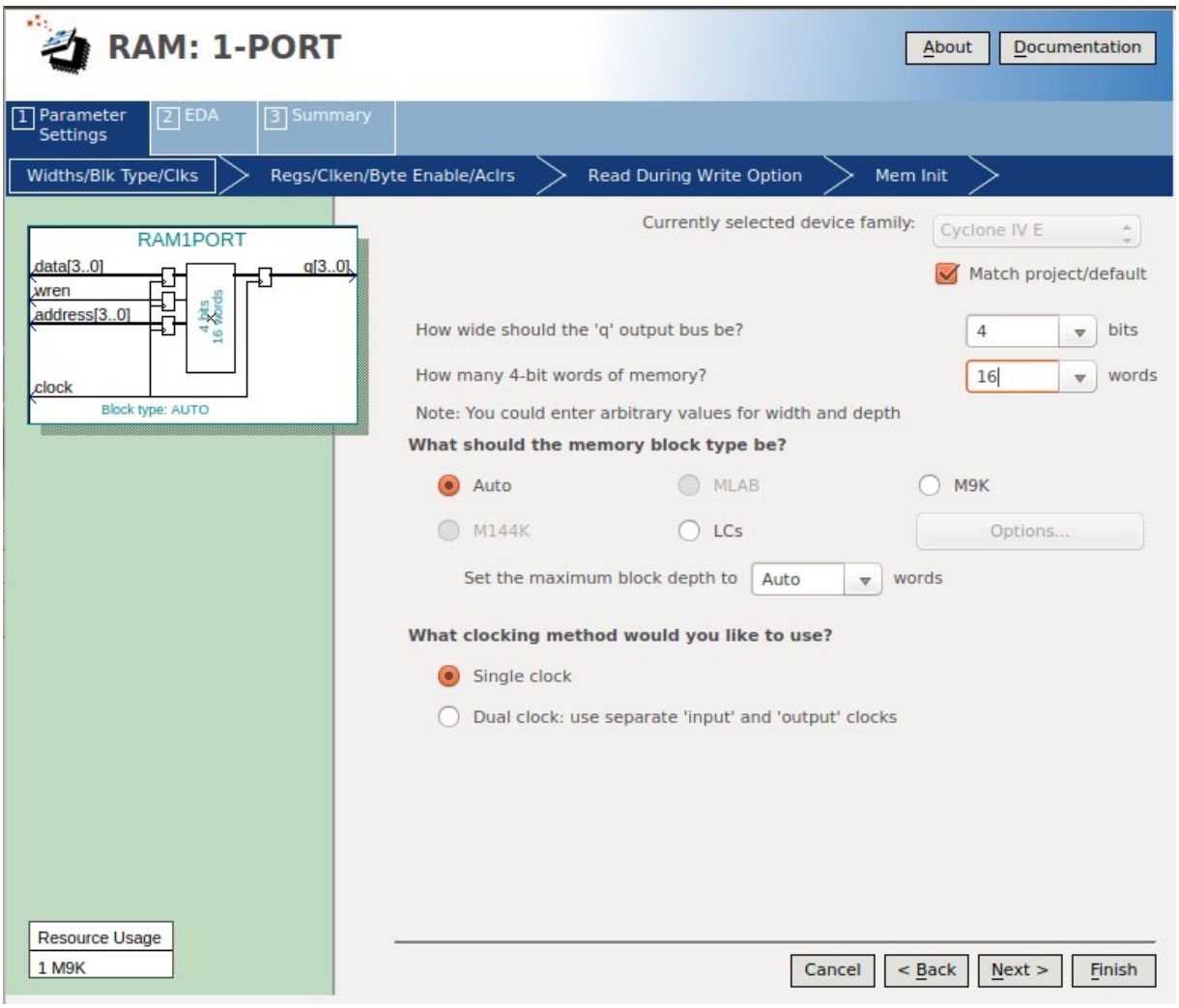


Figure 11: Make sure that you are using these settings when creating the RAM 1-Port, 4 bits wide, using 16 words, just like in the SRAM you are designing.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

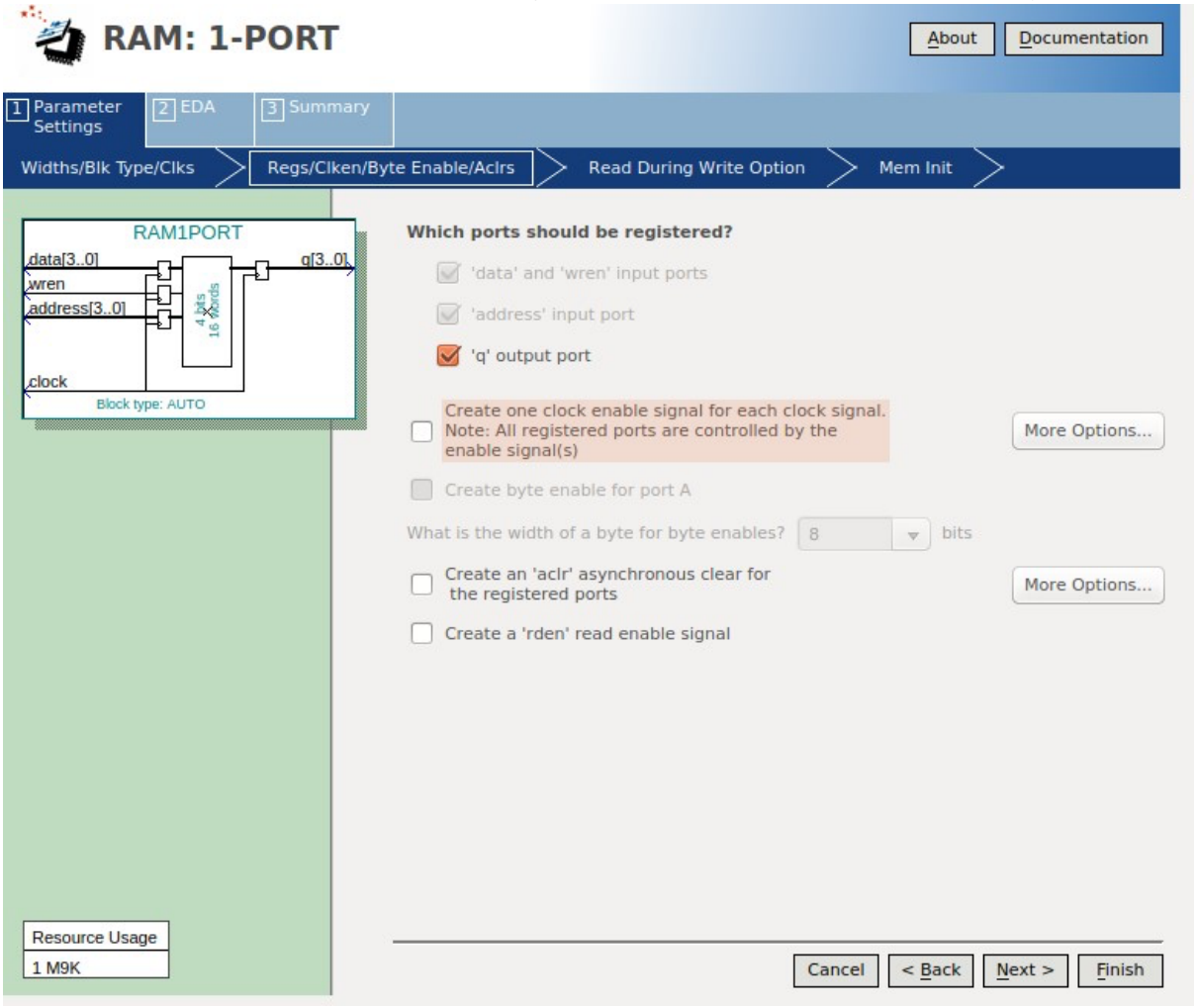


Figure 12: Do not change anything on the next screen just press next.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

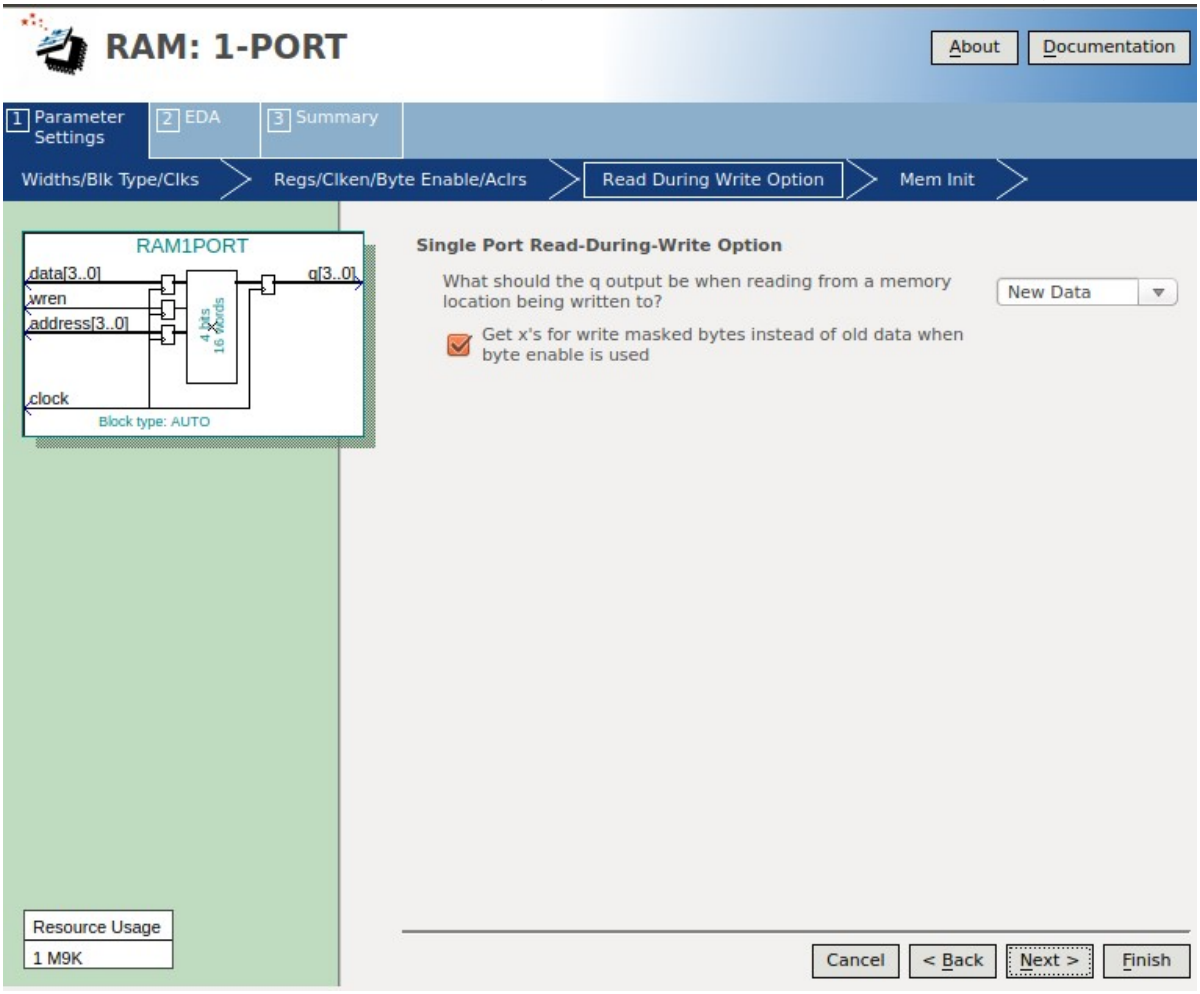


Figure 13: Make sure that this dialogue is set to true.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

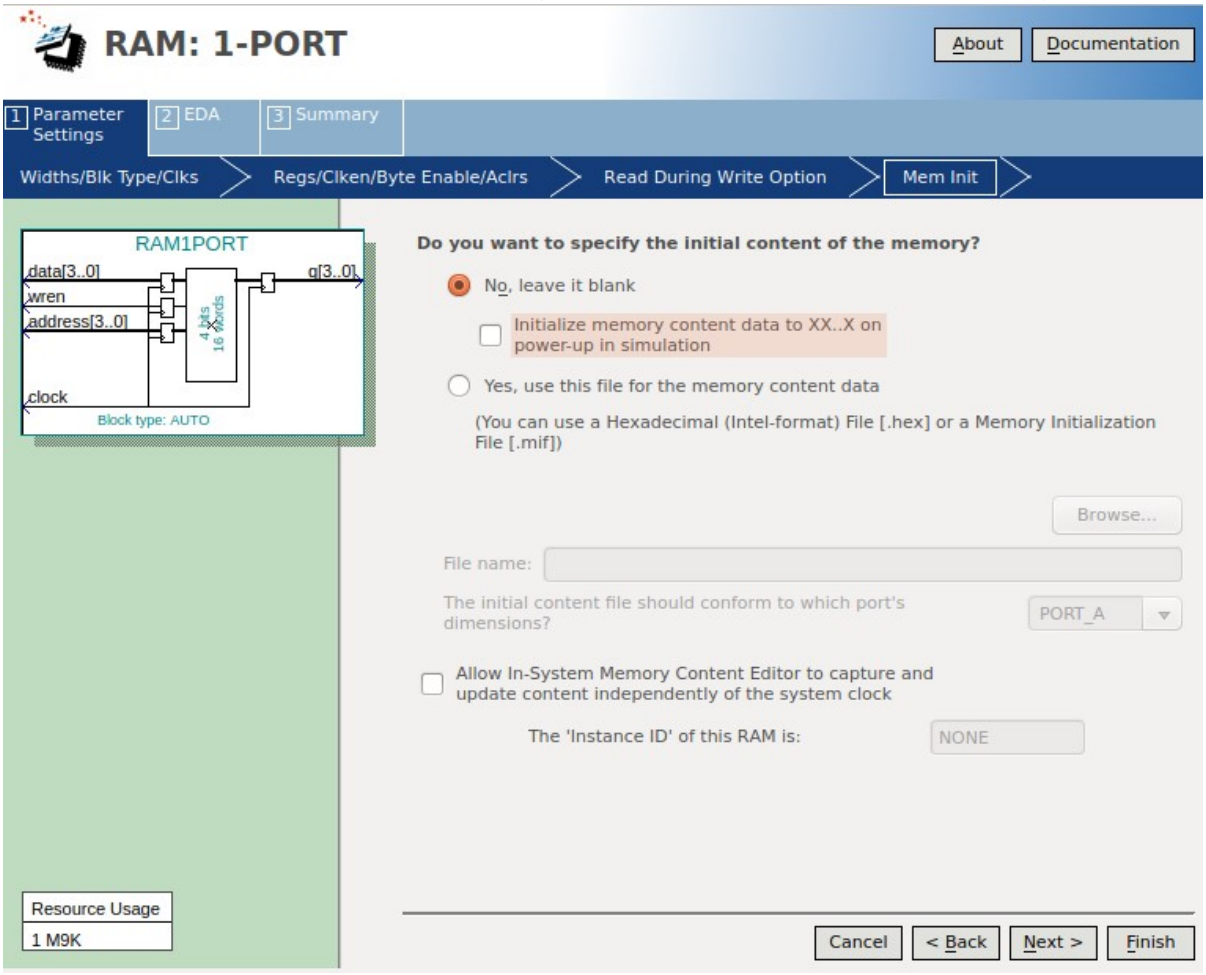


Figure 14: Select leave it blank and then continue by pressing next.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

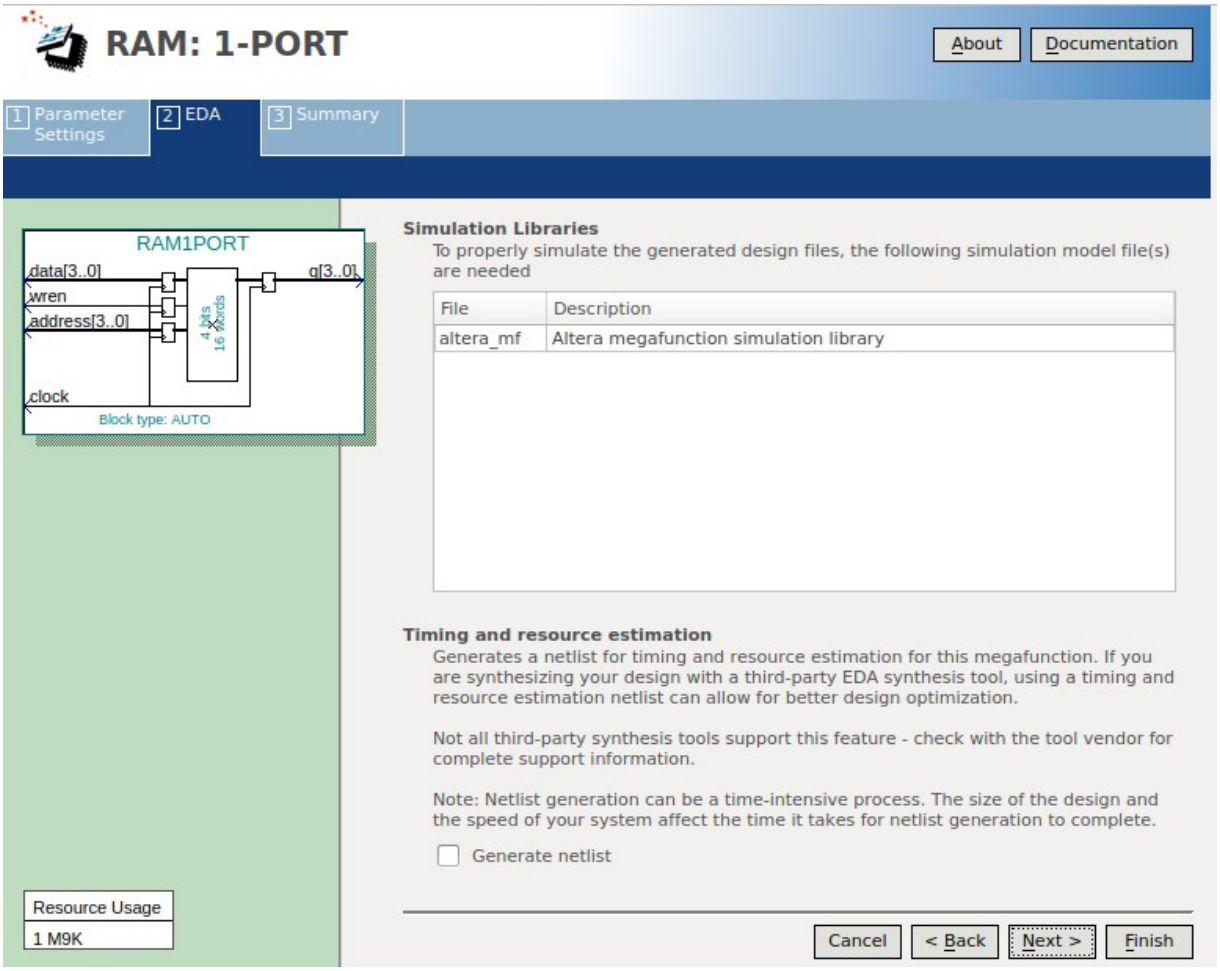


Figure 15: Continue onto the next screen.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

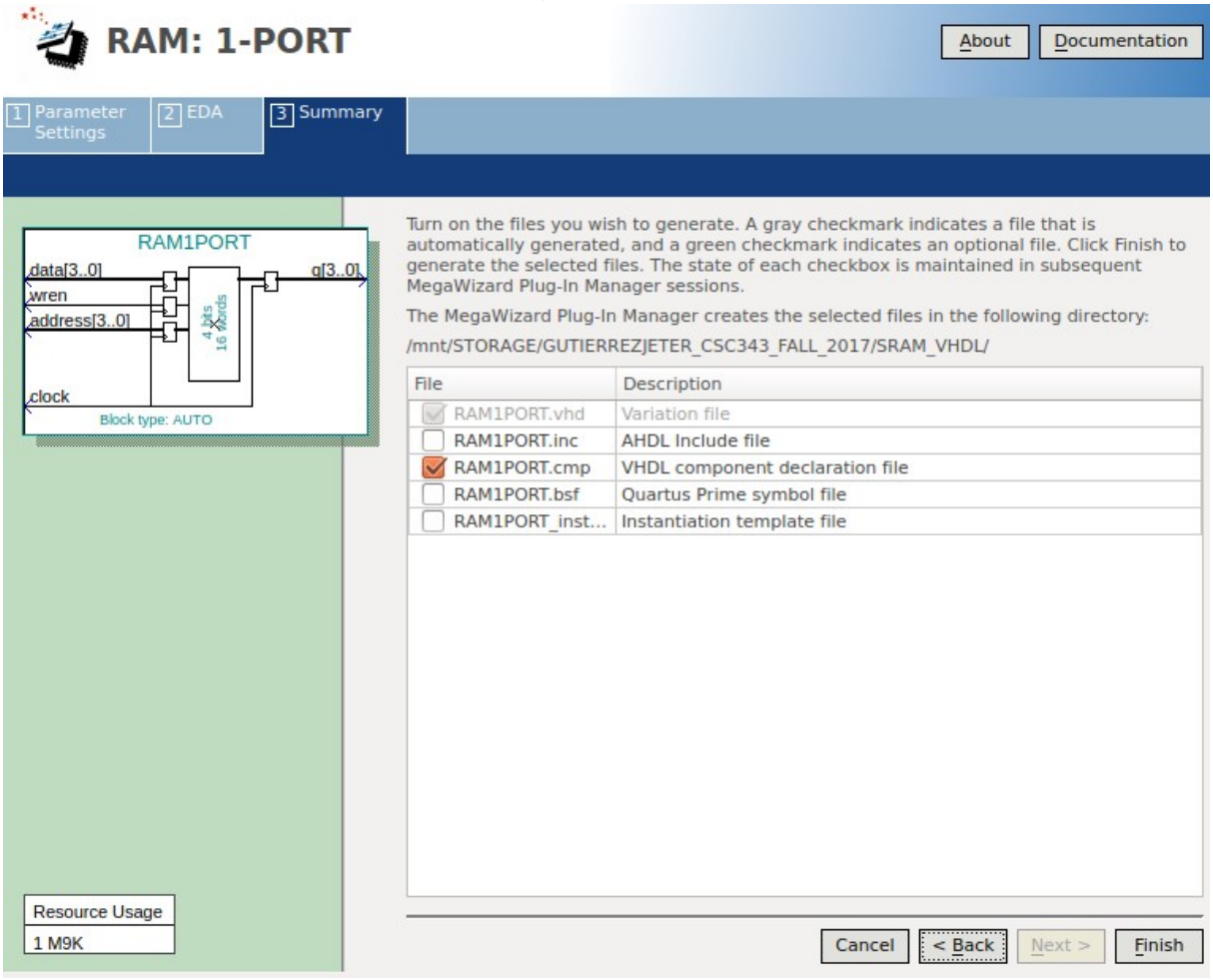


Figure 16: Match these settings and press finish. On the next screen after pressing finish press yes to adding the created file to the current project.

Task

1. Design a 16 x 4 SRAM. The output of this SRAM will be connected to a 4 to 7 decoder that outputs to a seven segment display. Input to the SRAM using 4 switches and a signal to write to memory.
2. Do the same using the LPM RAM 1 PORT.
3. Write a report and film a 1 minute video.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

PART II Objective is to Design a digital circuit to write data to a memory "CHIP" on the board, using switches on the board to input addresses and integer data. Therafter, please display integer data stored in memory using 7 segment display.

In this PART II you will be writing data, at a specified address, to a separate memory "chip" Synchronous Static Ram (SSRAM) on the DE2 board, thereafter you will have to display data at the specified address using seven segment display.

To get more information about SSRAM, please refer DE2 board users guide and reference manual.

We summarize here the description of SSRAM on the DE2 board.

■ **SRAM**

The DE2-115 board has 2MB SRAM memory with 16-bit data width. Being featured with a maximum performance frequency of about 125MHz under the condition of standard 3.3V single power supply makes it suitable of dealing with high-speed media processing applications that need ultra data throughput. The related schematic is shown in Figure 4-33.

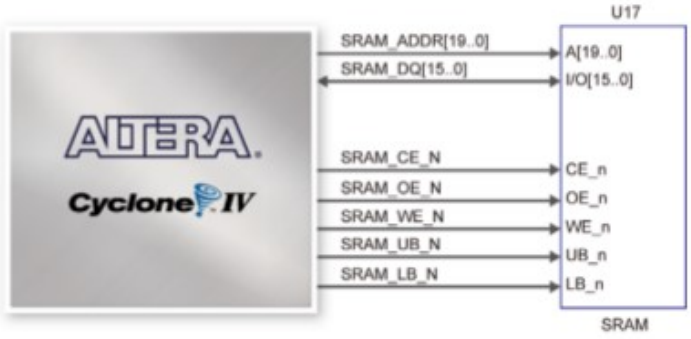


Figure 4-33 Connections between FPGA and SRAM

The FPGA boards come with memory. In the DE2-115, there are 2^{20} address locations represented by an array of 20 bits: **SRAM_ADDR[19..0]**. Each address can hold up to 16 bits of

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

data that can be accessed via its own Pin. The user manual contains a chart listing the pins of the addresses in the form of an array, with the name **SRAM_ADDR[19..0]** while the data that can be stored at each address is of the form **SRAM_DQ[15..0]**.

NOTE: In VHDL, the type for the data, SRAM_DQ, is **inout** instead of just **in** or **out** because it can be read from and written to on the FPGA board.

Among other variables that will be required to write to the board, the following will have to be initialized to **0** in order to read and write to the SRAM on the FPGA board:

SRAM_CE_N --Chip Select
SRAM_OE_N --Output enable
SRAM_UB_N --Sram higher byte strobe
SRAM_LB_N --Sram lower byte strobe

NOTE:

- Initialize the variables above, to '0' in your VHDL code, BUT they are still required to be assigned to their respective pins on the FPGA board. You will need to set them to '0' because that is how they will be interpreted as TRUE and grants access to the FPGA boards SRAM.
- Make sure that SRAM_ADDR, SRAM_CE_N, SRAM_OE_N, SRA,_UB_N, SRAM_LB_N, SRAM_LB_N are **out** signals.

For example on the DE2-115 Board the pins are:

SRAM_OE_N	PIN_AD5
SRAM_WE_N	PIN_AE8
SRAM_CE_N	PIN_AF8
SRAM_LB_N	PIN_AD4
SRAM_UB_N	PIN_AC4

We will be using the **SRAM_WE_N (Write enable)** to write to a specified address data segment.

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

YOUR TASK:

1. Create a circuit using VHDL (QUARTUS< MODELSIM) that will allow you to read and write to 8 addresses, each 8 bits wide, within the board's memory. USE switches on the board to input addresses and integer data.
    a. You will be accessing **SRAM_ADDR[7..0]**. This means that there are 2^8 different addresses but you will only be writing to 8. Set **SRAM_ADDR[19..8]**, to 0 in your VHDL code. However, you MUST still assign a pin to each SRAM_ADDR.
    b. Use switches 0-7 to input data into **SRAM_DQ[7..0]**. Set the values in **SRAM_DQ[15...8]** to 0 in your VHDL code since you will not be using them.
    c. You have to verify whether the information will be stored to **SRAM_DQ** when **SRAM_WE_N** is 1, 0, rising edge or falling edge.
2. In your simulation and on the DE2 Board, store a different value in 8 different addresses and read from them. Display the values, in hexadecimal, within the SRAM_DQ using 7 segment displays.
3. Write a report and create a 1 min demonstration video.

Your circuit will resemble the image below:

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

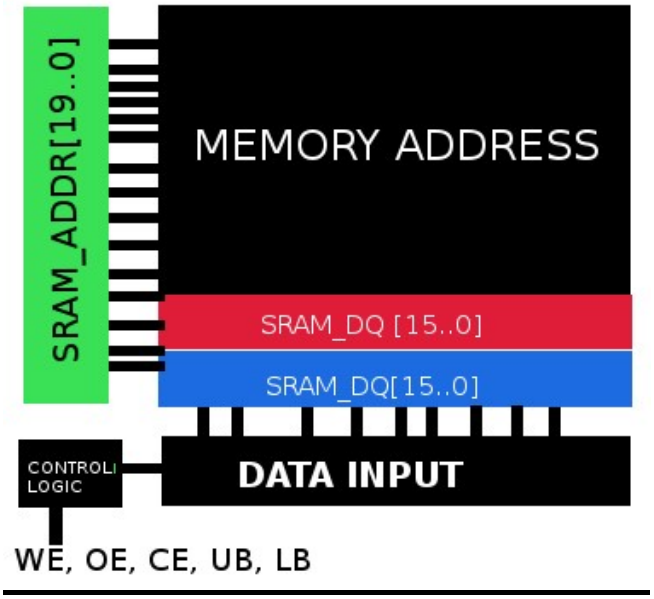


Figure : Block diagram for Synchronous Static Ram

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Template of SRAM design.

Your job is to fill in the blanks and make sure to correctly output the SRAM_DQ[15..0] using 7segment displays in Hexadecimal format and on the LED Lights.

```
library ieee;
use ieee.std_logic_1164.all;
entity Sram is
    port(
        INPUT_ADDR: in std_logic_vector(7 downto 0);
        INPUT_DATA: in std_logiC_vector(7 downto 0);
        INPUT_WE: in std_logic;
        SRAM_ADDR: out std_logic_vector(19 downto 0);
        SRAM_DQ: inout std_logic_vector(15 downto 0);
        SRAM_CE_N: out std_logic;
        SRAM_OE_N: out std_logic;
        SRAM_WE_N: out std_logic;
        SRAM_UB_N: out std_logic;
        SRAM_LB_N: out std_logic

        -- Add the led outputs and HEX display outputs for 4 7-segment displays.

        );
end Sram;

architecture arch of Sram is

    signal address: std_logic_vector(7 downto 0);
    signal data: std_logic_vector(7 downto 0);
    signal Wenable: std_logic;

    begin
        Wenable<=INPUT_WE;
        address <=INPUT_ADDR;
        data <= INPUT_DATA when Wenable = ? else (others => ?);--Fill in ?.
        SRAM_WE_N <= ?--Fill in ?
        SRAM_CE_N <= '0';
        SRAM_OE_N <= '0';
        SRAM_UB_N <= '0';
        SRAM_LB_N <= '0';
        SRAM_ADDR(19 downto 8) <= ? ---set unused ports to 0 Fill in ?
        SRAM_ADDR(7 downto 0) <= address;--assign address to first 8 ports
        SRAM_DQ(15 downto 8) <= ? --set unused port to 0 fill in ?
        SRAM_DQ(7 downto 0) <= data; --assign data to register

        -- You will still need to display results for the SRAM_DQ[15..0]
        -- on the Hexadecimal display and on LED lights.

end arch;
```

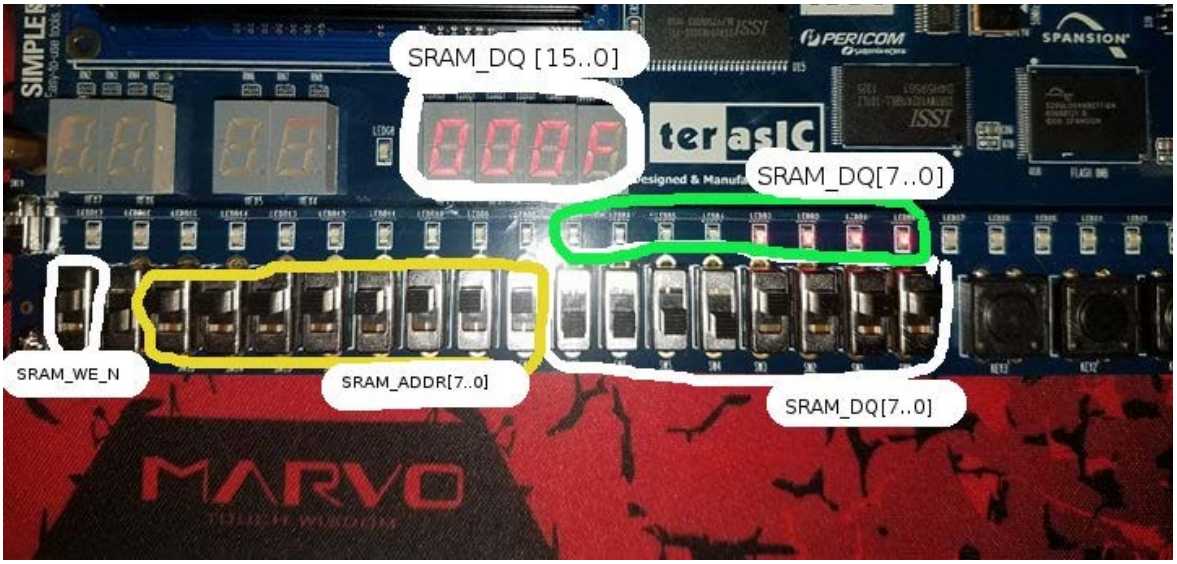
Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

DISPLAY ON DE2 BOARD:



In this image, although SRAM_ADDR and SRAM_DQ are assigned to the correct pins the input data for SRAM_DQ[7..0] is input using the first 8 switches, is shown using first 8 led lights and in the first 2 seven segment displays. In this case we are storing 0000000000001111 into the address 00000000000011111111111.

For the DE2-115 Board the pin assignments that will be used are as follows:

Laboratory Project: Review MEMORY LAB

DUE Date: March 14, 2022

Instructor: Professor Izidor Gertner

Prepared with the help of Gutierrez, Jeter and Lu, Kelly

Signal Name	*FPGA Pin No.*	*Description*
SRAM_ADDR[0]	PIN_AB7	SRAM Address[0]
SRAM_ADDR[1]	PIN_AD7	SRAM Address[1]
SRAM_ADDR[2]	PIN_AE7	SRAM Address[2]
SRAM_ADDR[3]	PIN_AC7	SRAM Address[3]
SRAM_ADDR[4]	PIN_AB6	SRAM Address[4]
SRAM_ADDR[5]	PIN_AE6	SRAM Address[5]
SRAM_ADDR[6]	PIN_AB5	SRAM Address[6]
SRAM_ADDR[7]	PIN_AC5	SRAM Address[7]
SRAM_ADDR[8]	PIN_AF5	SRAM Address[8]
SRAM_ADDR[9]	PIN_T7	SRAM Address[9]
SRAM_ADDR[10]	PIN_AF2	SRAM Address[10]
SRAM_ADDR[11]	PIN_AD3	SRAM Address[11]
SRAM_ADDR[12]	PIN_AB4	SRAM Address[12]
SRAM_ADDR[13]	PIN_AC3	SRAM Address[13]
SRAM_ADDR[14]	PIN_AA4	SRAM Address[14]
SRAM_ADDR[15]	PIN_AB11	SRAM Address[15]
SRAM_ADDR[16]	PIN_AC11	SRAM Address[16]
SRAM_ADDR[17]	PIN_AB9	SRAM Address[17]
SRAM_ADDR[18]	PIN_AB8	SRAM Address[18]
SRAM_ADDR[19]	PIN_T8	SRAM Address[19]
SRAM_DQ[0]	PIN_AH3	SRAM Data[0]
SRAM_DQ[1]	PIN_AF4	SRAM Data[1]
SRAM_DQ[2]	PIN_AG4	SRAM Data[2]
SRAM_DQ[3]	PIN_AH4	SRAM Data[3]
SRAM_DQ[4]	PIN_AF6	SRAM Data[4]
SRAM_DQ[5]	PIN_AG6	SRAM Data[5]
SRAM_DQ[6]	PIN_AH6	SRAM Data[6]
SRAM_DQ[7]	PIN_AF7	SRAM Data[7]
SRAM_DQ[8]	PIN_AD1	SRAM Data[8]
SRAM_DQ[9]	PIN_AD2	SRAM Data[9]
SRAM_DQ[10]	PIN_AE2	SRAM Data[10]
SRAM_DQ[11]	PIN_AE1	SRAM Data[11]
SRAM_DQ[12]	PIN_AE3	SRAM Data[12]
SRAM_DQ[13]	PIN_AE4	SRAM Data[13]
SRAM_DQ[14]	PIN_AF3	SRAM Data[14]
SRAM_DQ[15]	PIN_AG3	SRAM Data[15]
SRAM_OE_N	PIN_AD5	SRAM Output Enable
SRAM_WE_N	PIN_AE8	SRAM Write Enable
SRAM_CE_N	PIN_AF8	SRAM Chip Select
SRAM_LB_N	PIN_AD4	SRAM Lower Byte Strobe
SRAM_UB_N	PIN_AC4	SRAM Higher Byte Strobe