MAX 10 Memory HDL

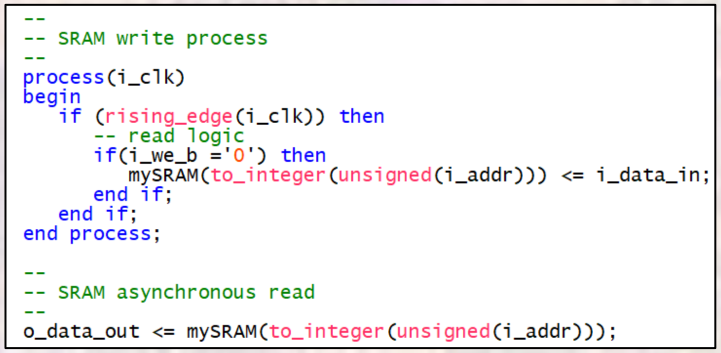
Last updated 7/19/23

MAX10 Memory HDL

- Three types of MAX10 Memory
  - FF based memory using the LEs
    - Most efficient for very small memories
    - Compiler driven
  - M9K Fixed Memory Blocks
    - Embedded SRAM block
    - 8K bits + 1024 parity bits (9216b)
    - MAX 10-M50 has 182 blocks
    - 1,456Kb + parity = 1,677,312b total
  - User Flash
    - User Programmable – 5,888Kb
    - Configuration - 10,752Kb

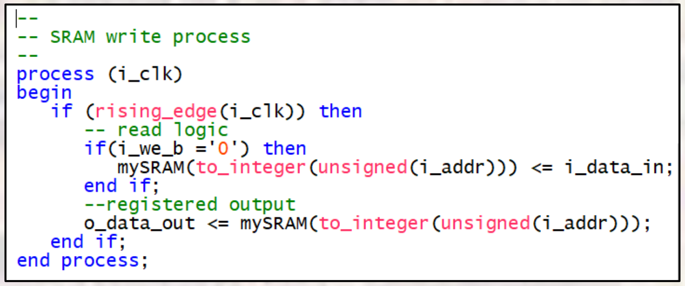
MAX10 Memory HDL

- Three ways to implement memory
  - FF based memory embedded in the LEs



```
--
-- SRAM write process
--
process(i_clk)
begin
   if (rising_edge(i_clk)) then
      -- read logic
      if(i_we_b ='0') then
         mySRAM(to_integer(unsigned(i_addr))) <= i_data_in;
      end if;
   end if;
end process;

--
-- SRAM asynchronous read
--
o_data_out <= mySRAM(to_integer(unsigned(i_addr)));
```

  - Inferred Memory



```
--
-- SRAM write process
--
process (i_clk)
begin
   if (rising_edge(i_clk)) then
      -- read logic
      if(i_we_b ='0') then
         mySRAM(to_integer(unsigned(i_addr))) <= i_data_in;
      end if;
      --registered output
      o_data_out <= mySRAM(to_integer(unsigned(i_addr)));
   end if;
end process;
```

  - IP blocks

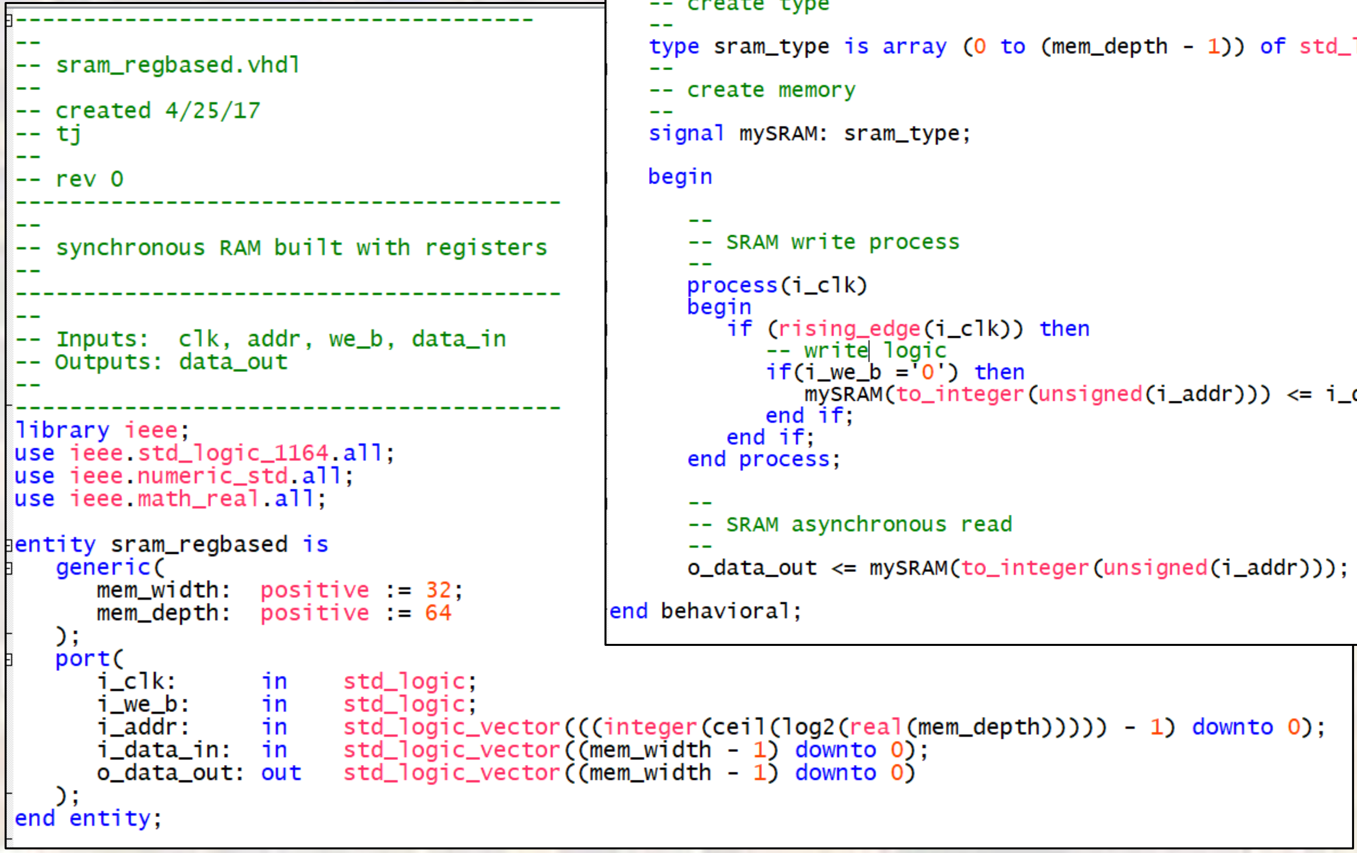
MAX10 Memory HDL

- FF based memory embedded in the LEs
  - Simple memory
    - Static
    - N bits wide
    - M bits deep
    - Synchronous write

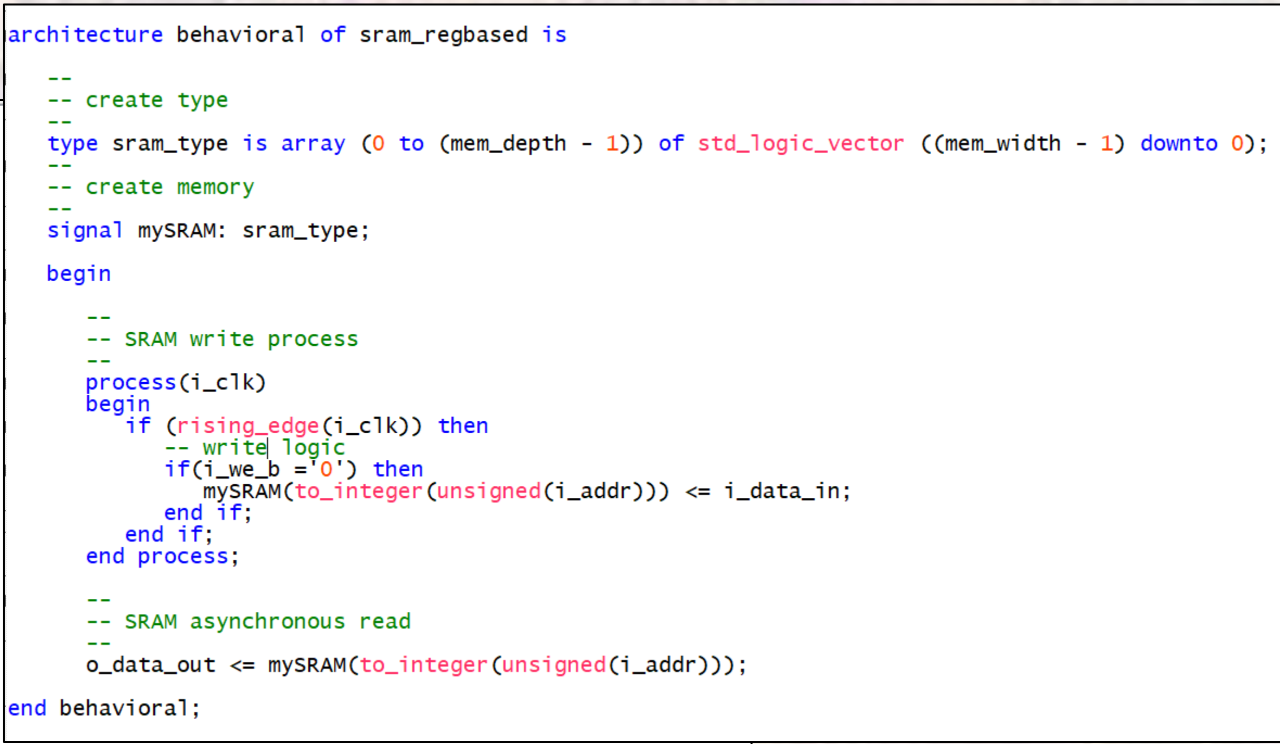
    - Inefficient for all but the smallest memories

MAX10 Memory HDL

- FF based memory embedded in the LEs



```
-----------------------------------------
--
-- sram_regbased.vhdl
--
-- created 4/25/17
-- tj
--
-- rev 0
-----------------------------------------
--
-- synchronous RAM built with registers
--
-----------------------------------------
--
-- Inputs:  clk, addr, we_b, data_in
-- Outputs: data_out
--
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity sram_regbased is
   generic(
      mem_width:  positive := 32;
      mem_depth:  positive := 64
   );
   port(
      i_clk:      in    std_logic;
      i_we_b:     in    std_logic;
      i_addr:     in    std_logic_vector(((integer(ceil(log2(real(mem_depth))))) - 1) downto 0);
      i_data_in:  in    std_logic_vector((mem_width - 1) downto 0);
      o_data_out: out   std_logic_vector((mem_width - 1) downto 0)
   );
end entity;
```

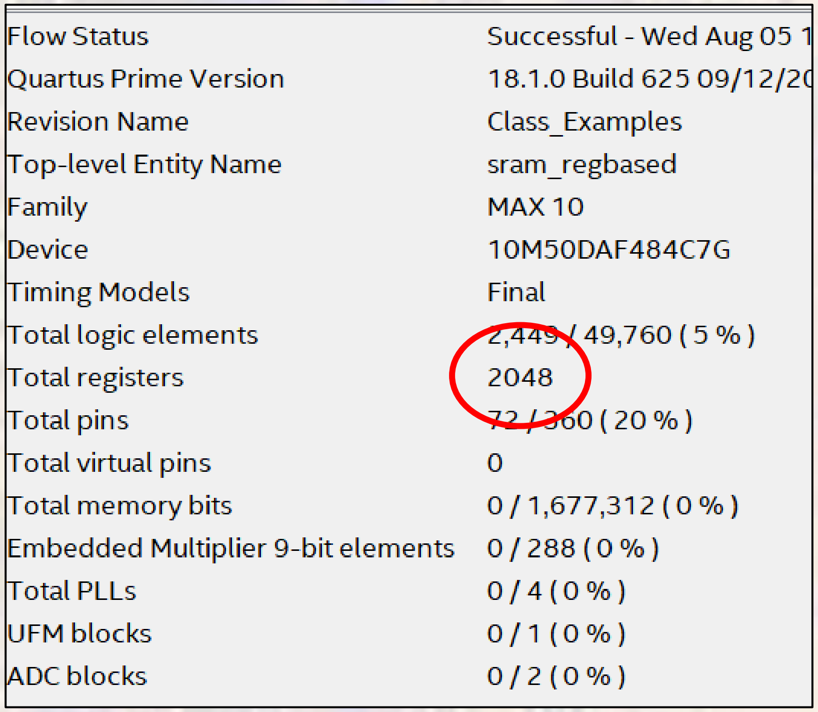


```
architecture behavioral of sram_regbased is

   --
   -- create type
   --
   type sram_type is array (0 to (mem_depth - 1)) of std_logic_vector ((mem_width - 1) downto 0);
   --
   -- create memory
   --
   signal mySRAM: sram_type;

   begin

      --
      -- SRAM write process
      --
      process(i_clk)
      begin
         if (rising_edge(i_clk)) then
            -- write logic
            if(i_we_b ='0') then
               mySRAM(to_integer(unsigned(i_addr))) <= i_data_in;
            end if;
         end if;
      end process;

      --
      -- SRAM asynchronous read
      --
      o_data_out <= mySRAM(to_integer(unsigned(i_addr)));

end behavioral;
```

MAX10 Memory HDL

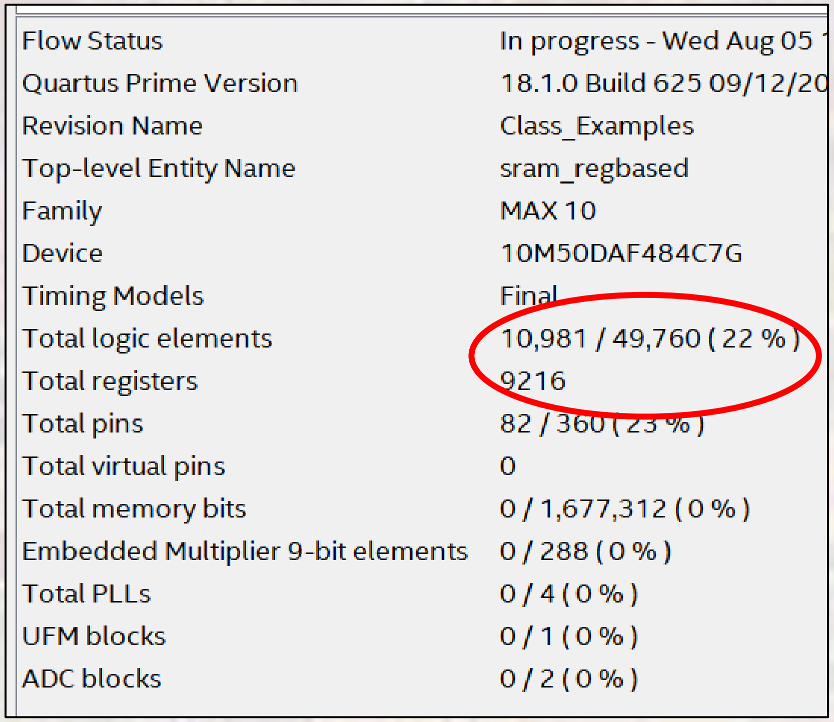
- FF based memory embedded in the LEs

64 x 32b



Flow Status	Successful - Wed Aug 05 1
Quartus Prime Version	18.1.0 Build 625 09/12/20
Revision Name	Class_Examples
Top-level Entity Name	sram_regbased
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	2,449 / 49,760 (5 %)
Total registers	2048
Total pins	72 / 360 (20 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

256 x 36b



Flow Status	In progress - Wed Aug 05 1
Quartus Prime Version	18.1.0 Build 625 09/12/20
Revision Name	Class_Examples
Top-level Entity Name	sram_regbased
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	10,981 / 49,760 (22 %)
Total registers	9216
Total pins	82 / 360 (23 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Equivalent to one M9K block

MAX10 Memory HDL

- M9K Fixed Memory Blocks

  - Functional configurations

    - Single-port
    - Simple dual-port
    - True dual-port (bidirectional dual-port)
    - Shift register
    - ROM
    - FIFO buffers
    - Memory Based Multiplier

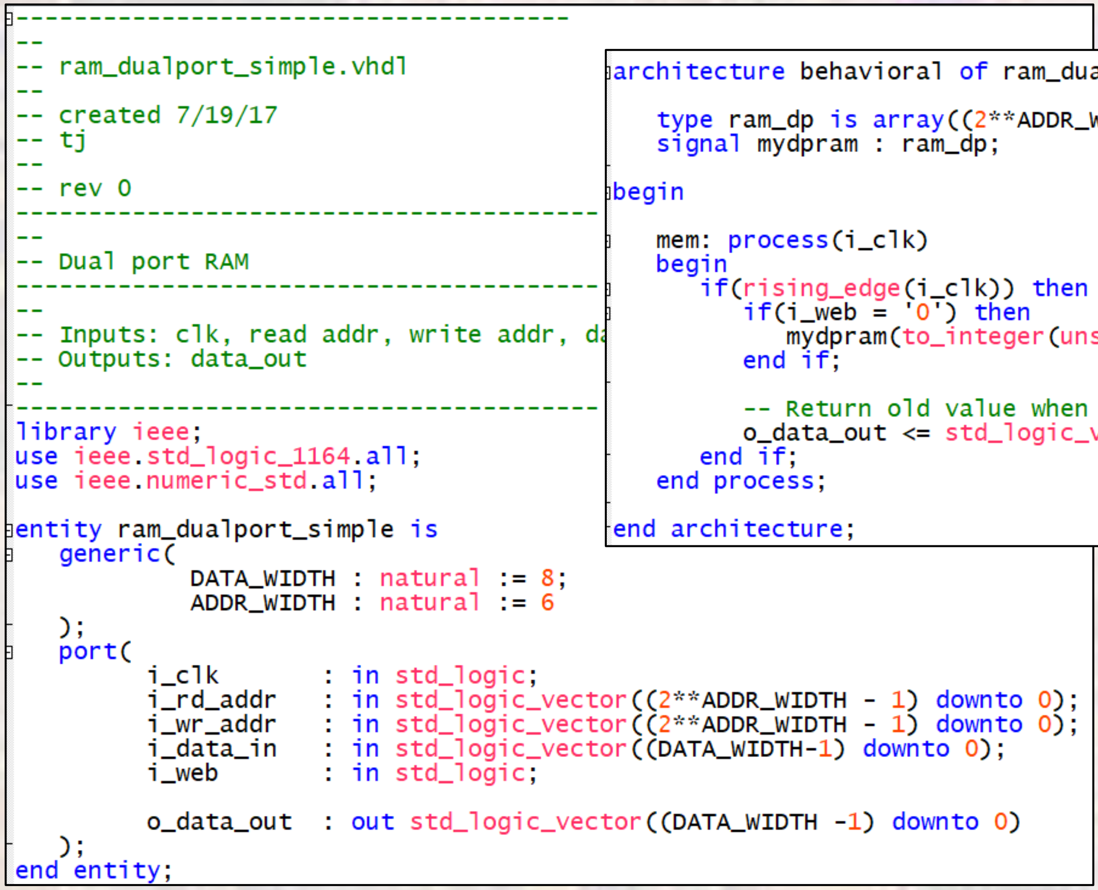
MAX10 Memory HDL

- Inferred - Simple Dual-Port RAM
  - Basic RAM
  - Separate read and write address inputs
  - Provides the old data if reading and writing to the same address (no write-through)
  - Inputs and outputs registered

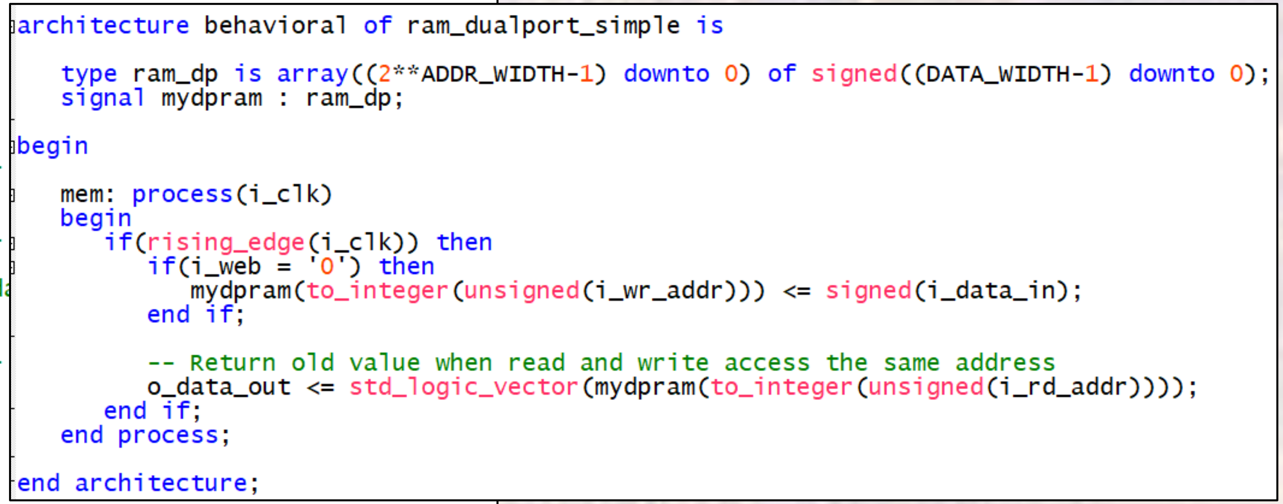
  - To be inferred
    - All input ports must be registered
      - rd addr, wr addr, web, data in
    - Output port must be registered

MAX10 Memory HDL

- Inferred - Simple Dual-Port RAM



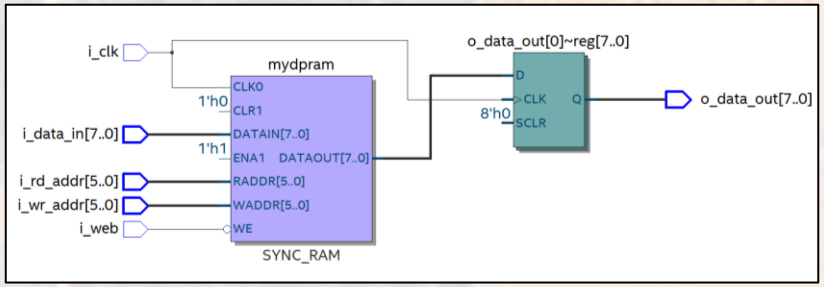
```vhdl
-----------------------------------------
--
-- ram_dualport_simple.vhdl
--
-- created 7/19/17
-- tj
--
-- rev 0
-----------------------------------------
--
-- Dual port RAM
-----------------------------------------
--
-- Inputs: clk, read addr, write addr, da
-- Outputs: data_out
--
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ram_dualport_simple is
   generic(
            DATA_WIDTH : natural := 8;
            ADDR_WIDTH : natural := 6
   );
   port(
         i_clk       : in std_logic;
         i_rd_addr   : in std_logic_vector((2**ADDR_WIDTH - 1) downto 0);
         i_wr_addr   : in std_logic_vector((2**ADDR_WIDTH - 1) downto 0);
         i_data_in   : in std_logic_vector((DATA_WIDTH-1) downto 0);
         i_web       : in std_logic;

         o_data_out  : out std_logic_vector((DATA_WIDTH -1) downto 0)
   );
end entity;
```

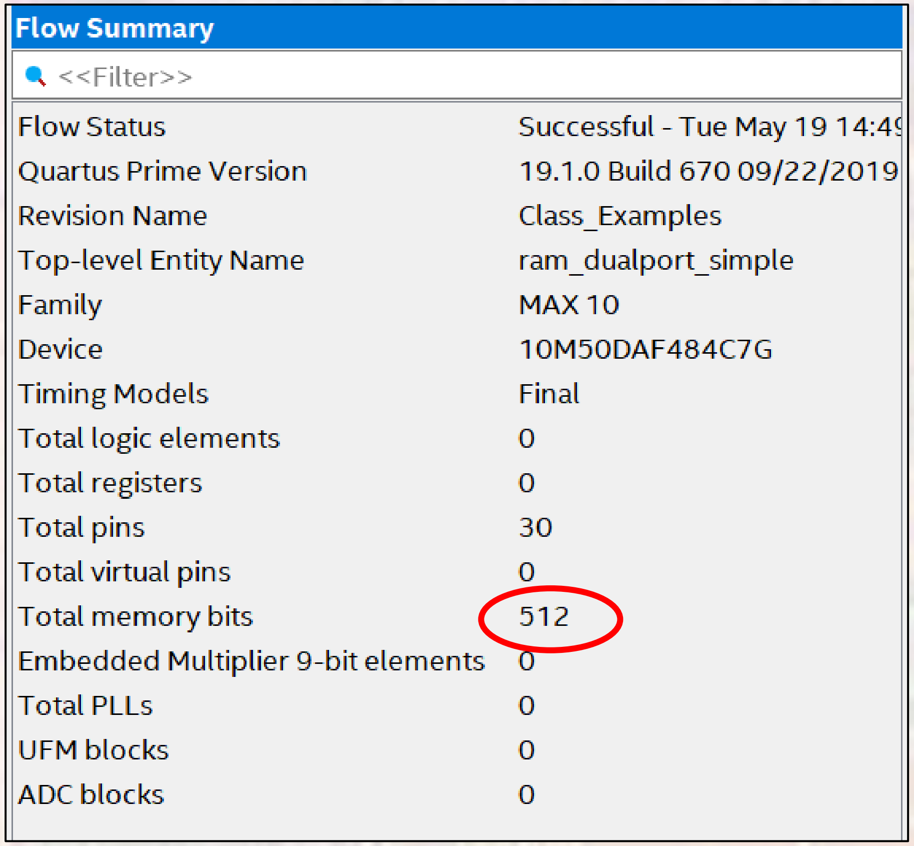


```vhdl
architecture behavioral of ram_dualport_simple is

   type ram_dp is array((2**ADDR_WIDTH-1) downto 0) of signed((DATA_WIDTH-1) downto 0);
   signal mydpram : ram_dp;

begin

   mem: process(i_clk)
   begin
      if(rising_edge(i_clk)) then
         if(i_web = '0') then
            mydpram(to_integer(unsigned(i_wr_addr))) <= signed(i_data_in);
         end if;

         -- Return old value when read and write access the same address
         o_data_out <= std_logic_vector(mydpram(to_integer(unsigned(i_rd_addr))));
      end if;
   end process;

end architecture;
```

MAX10 Memory HDL

- Inferred - Simple Dual-Port RAM





Flow Summary

<<Filter>>

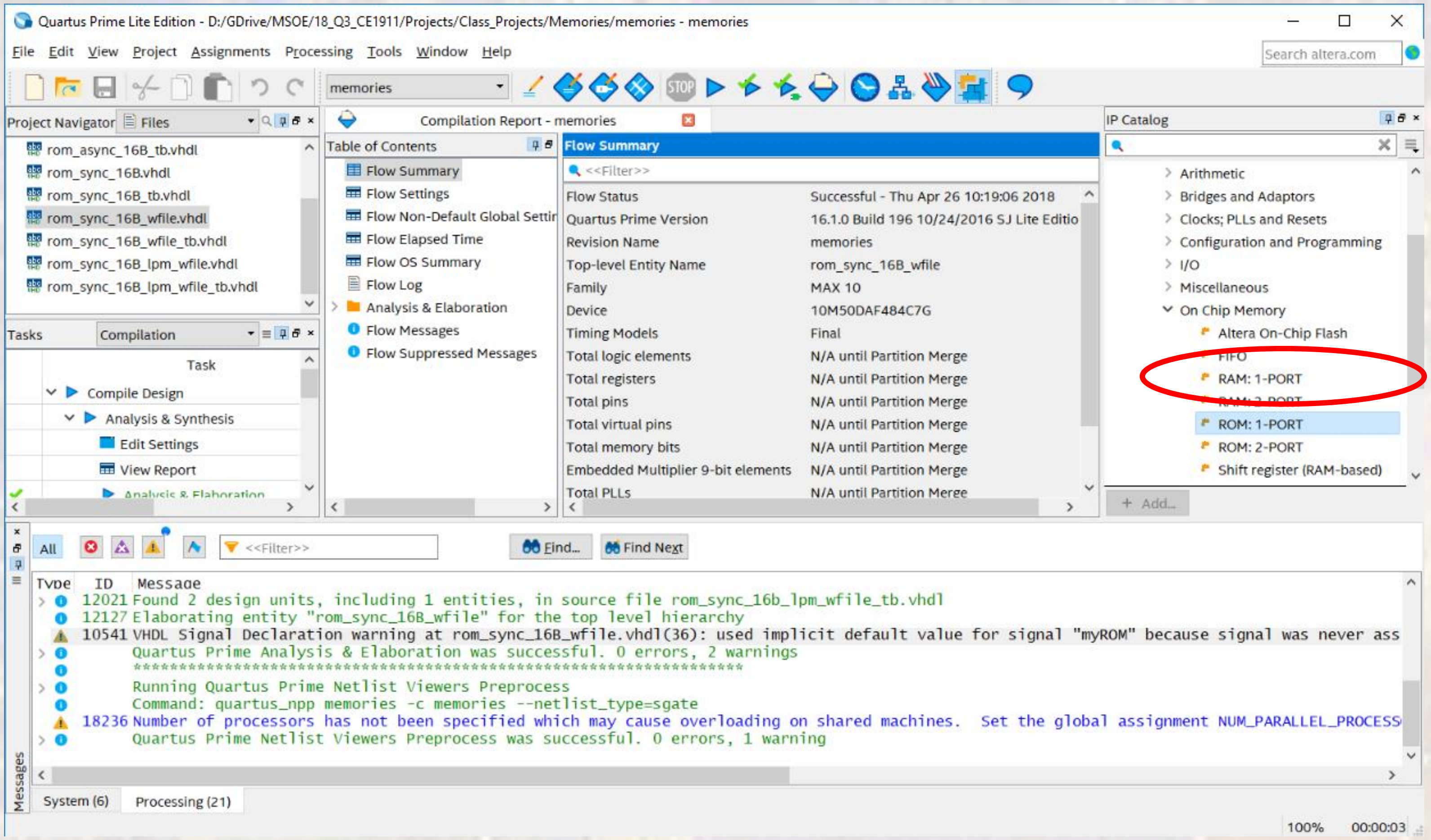
Flow Status	Successful - Tue May 19 14:4
Quartus Prime Version	19.1.0 Build 670 09/22/2019
Revision Name	Class_Examples
Top-level Entity Name	ram_dualport_simple
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	0
Total registers	0
Total pins	30
Total virtual pins	0
Total memory bits	512
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

MAX10 Memory HDL

- IP Based - RAM
  - Quartus has a series of pre-defined blocks
  - They can be created in Quartus - MegaWizard
  - They need to be instantiated in our design

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard process



Quartus Prime Lite Edition - D:/GDrive/MSOE/18_Q3_CE1911/Projects/Class_Projects/Memories/memories - memories

rom_async_16B_tb.vhdl
rom_sync_16B.vhdl
rom_sync_16B_tb.vhdl
rom_sync_16B_wfile.vhdl
rom_sync_16B_wfile_tb.vhdl
rom_sync_16B_lpm_wfile.vhdl
rom_sync_16B_lpm_wfile_tb.vhdl

Flow Summary
Flow Settings
Flow Non-Default Global Settir
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Elaboration
Flow Messages
Flow Suppressed Messages

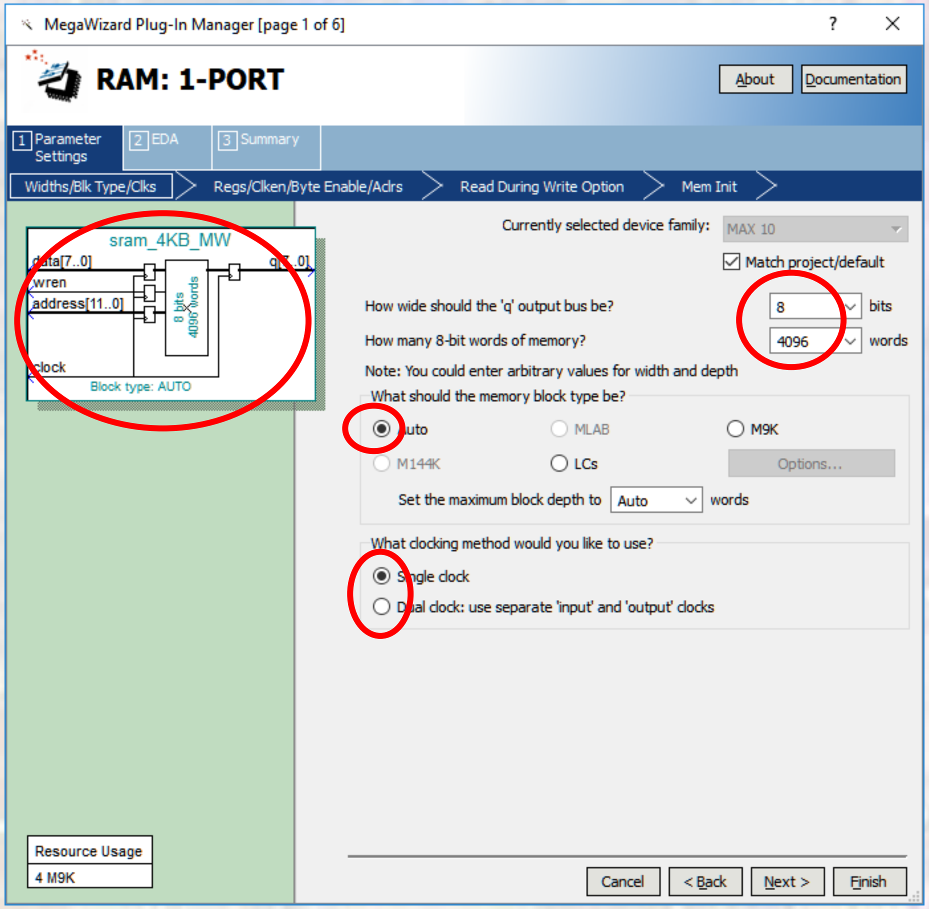
Flow Summary

Flow Status	Successful - Thu Apr 26 10:19:06 2018
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Editio
Revision Name	memories
Top-level Entity Name	rom_sync_16B_wfile
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	N/A until Partition Merge
Total registers	N/A until Partition Merge
Total pins	N/A until Partition Merge
Total virtual pins	N/A until Partition Merge
Total memory bits	N/A until Partition Merge
Embedded Multiplier 9-bit elements	N/A until Partition Merge
Total PLLs	N/A until Partition Merge

```
Type  ID    Message
      12021 Found 2 design units, including 1 entities, in source file rom_sync_16b_lpm_wfile_tb.vhdl
      12127 Elaborating entity "rom_sync_16B_wfile" for the top level hierarchy
      10541 VHDL Signal Declaration warning at rom_sync_16B_wfile.vhdl(36): used implicit default value for signal "myROM" because signal was never ass
            Quartus Prime Analysis & Elaboration was successful. 0 errors, 2 warnings
            ***********************************************************************
            Running Quartus Prime Netlist Viewers Preprocess
            Command: quartus_npp memories -c memories --netlist_type=sgate
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESS
            Quartus Prime Netlist Viewers Preprocess was successful. 0 errors, 1 warning
```

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard process



MegaWizard Plug-In Manager [page 1 of 6]

RAM: 1-PORT

Currently selected device family: MAX 10

[✓] Match project/default

How wide should the 'q' output bus be? 8 bits

How many 8-bit words of memory? 4096 words

Note: You could enter arbitrary values for width and depth

What should the memory block type be?

[✓ Auto] MLAB M9K M144K LCs

Options...

Set the maximum block depth to Auto words

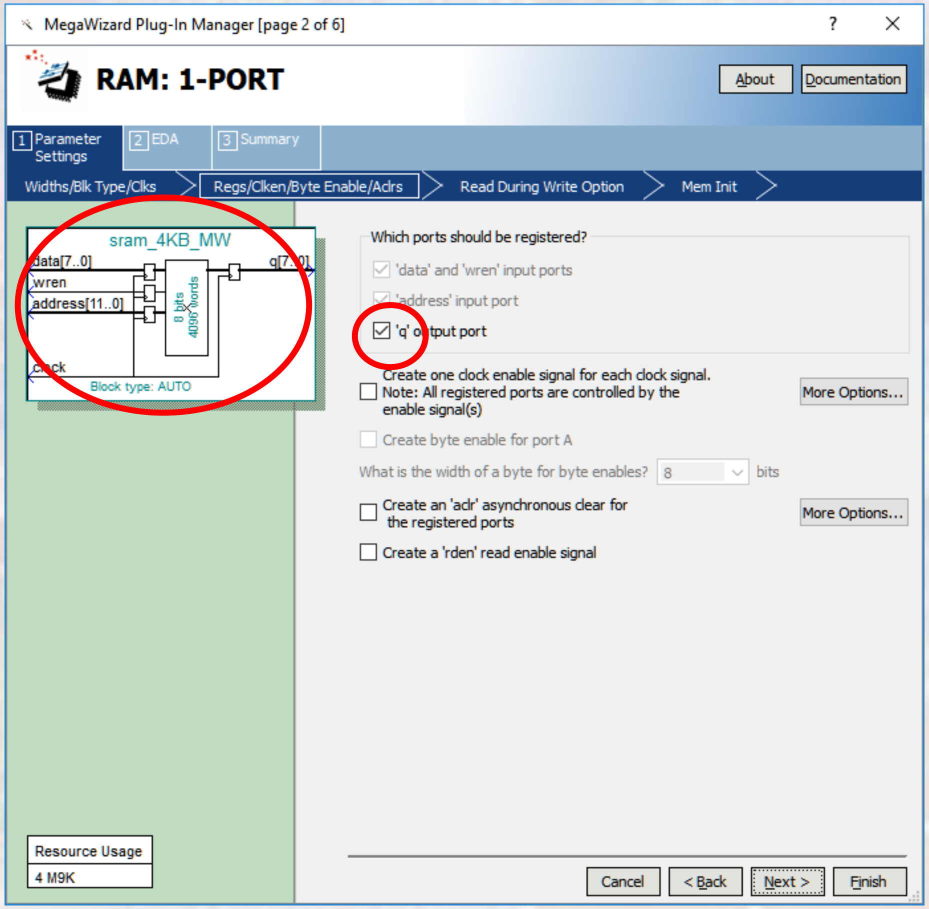
What clocking method would you like to use?

[✓ Single clock]

Dual clock: use separate 'input' and 'output' clocks

Resource Usage

4 M9K



MegaWizard Plug-In Manager [page 2 of 6]

RAM: 1-PORT

Which ports should be registered?

[✓] 'data' and 'wren' input ports

[✓] 'address' input port

[✓] 'q' output port

[ ] Create one clock enable signal for each clock signal. Note: All registered ports are controlled by the enable signal(s)

More Options...

[ ] Create byte enable for port A

What is the width of a byte for byte enables? 8 bits

[ ] Create an 'aclr' asynchronous clear for the registered ports

More Options...

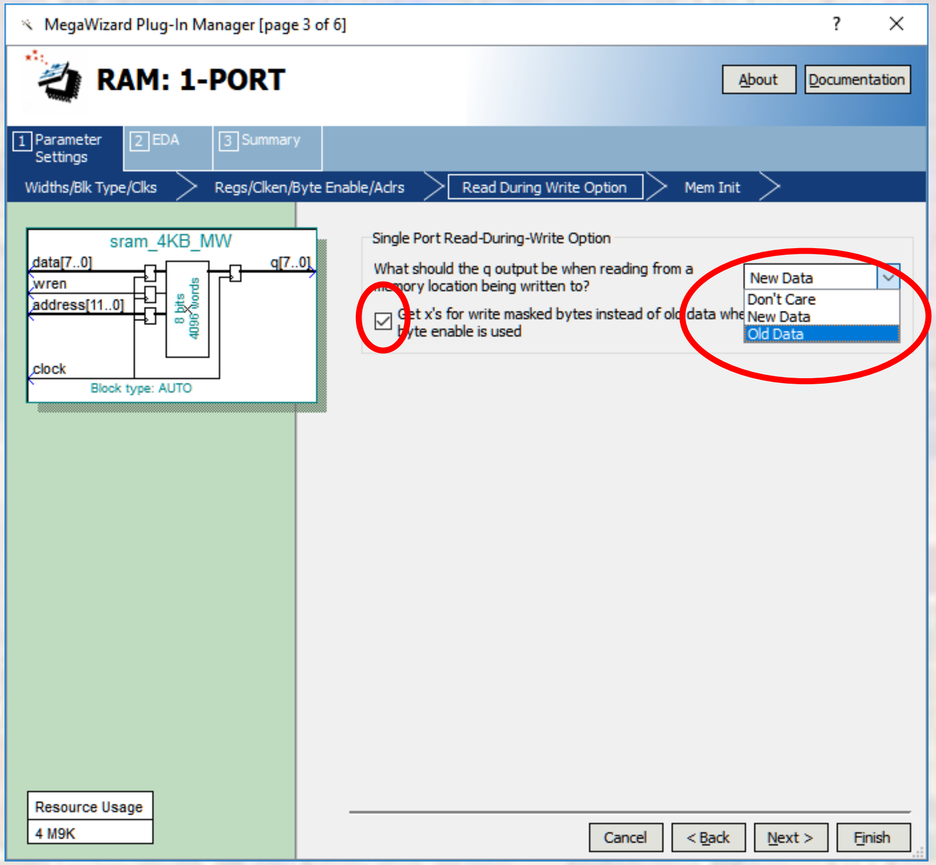
[ ] Create a 'rden' read enable signal

Resource Usage

4 M9K

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard process


MegaWizard Plug-In Manager [page 3 of 6]

RAM: 1-PORT

About | Documentation

1 Parameter Settings | 2 EDA | 3 Summary

Widths/Blk Type/Clks > Regs/Clken/Byte Enable/Adrs > Read During Write Option > Mem Init >

Single Port Read-During-Write Option

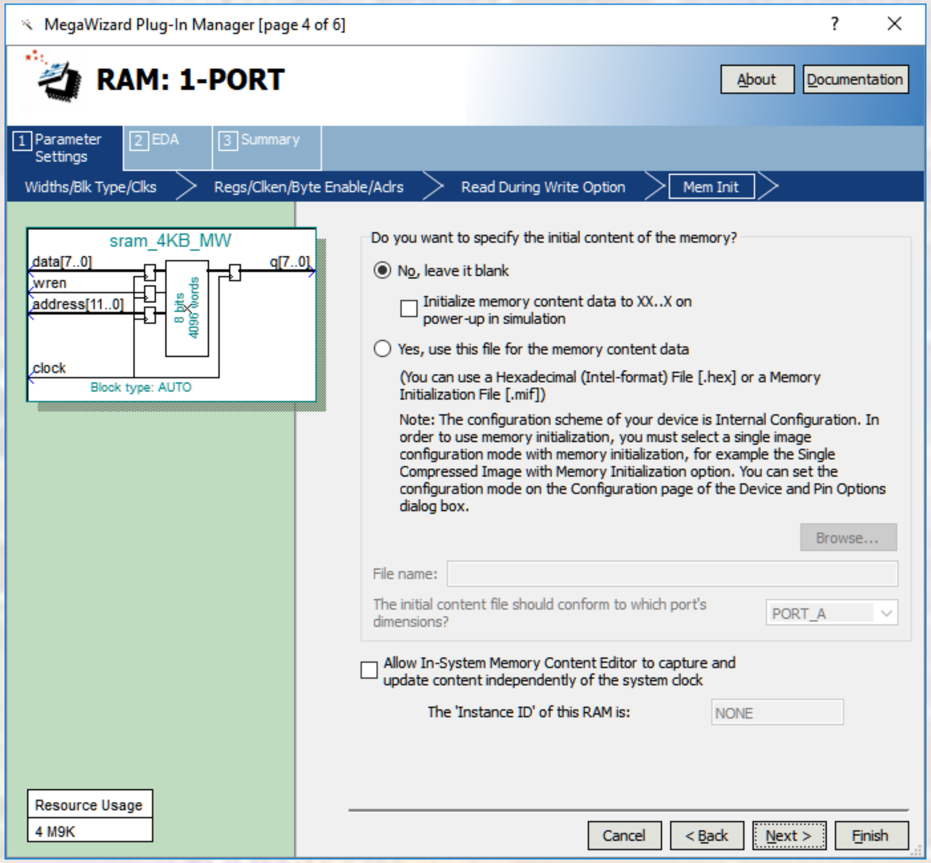
What should the q output be when reading from a memory location being written to?

New Data
- Don't Care
- New Data
- Old Data

[✓] Get x's for write masked bytes instead of old data whe… byte enable is used

Resource Usage: 4 M9K

Cancel | < Back | Next > | Finish


MegaWizard Plug-In Manager [page 4 of 6]

RAM: 1-PORT

About | Documentation

1 Parameter Settings | 2 EDA | 3 Summary

Widths/Blk Type/Clks > Regs/Clken/Byte Enable/Adrs > Read During Write Option > Mem Init >

Do you want to specify the initial content of the memory?

● No, leave it blank

☐ Initialize memory content data to XX..X on power-up in simulation

○ Yes, use this file for the memory content data

(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])

Note: The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the Device and Pin Options dialog box.

Browse...

File name:

The initial content file should conform to which port's dimensions? PORT_A

☐ Allow In-System Memory Content Editor to capture and update content independently of the system clock

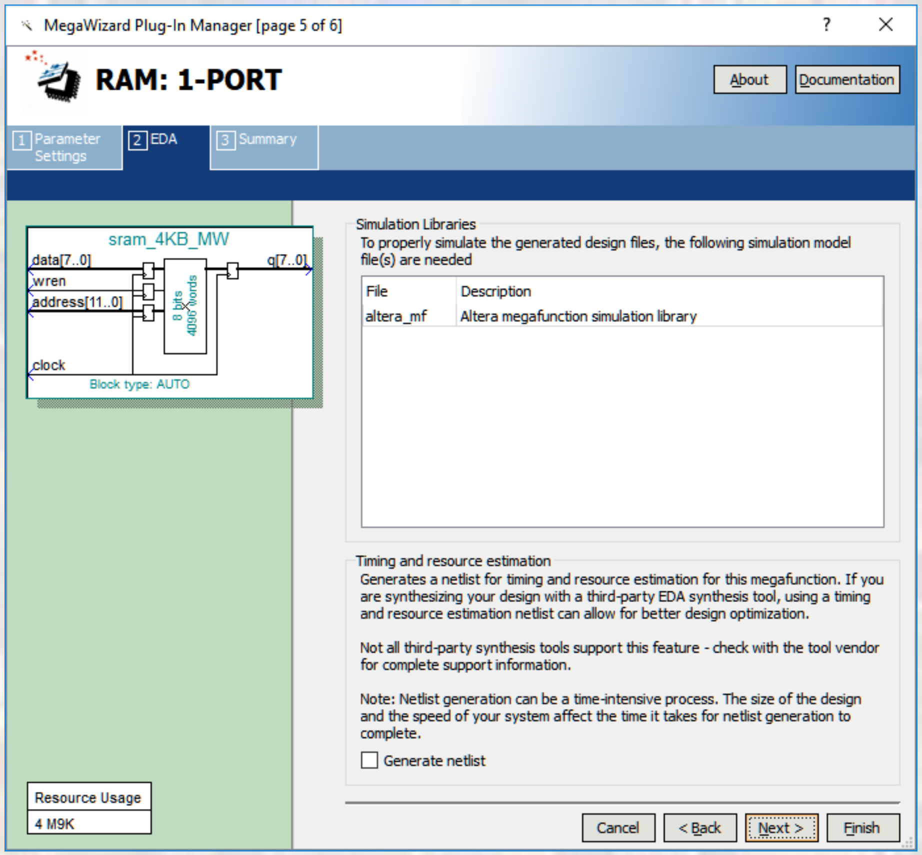
The 'Instance ID' of this RAM is: NONE

Resource Usage: 4 M9K

Cancel | < Back | Next > | Finish

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard process



Simulation Libraries

To properly simulate the generated design files, the following simulation model file(s) are needed

File	Description
altera_mf	Altera megafunction simulation library

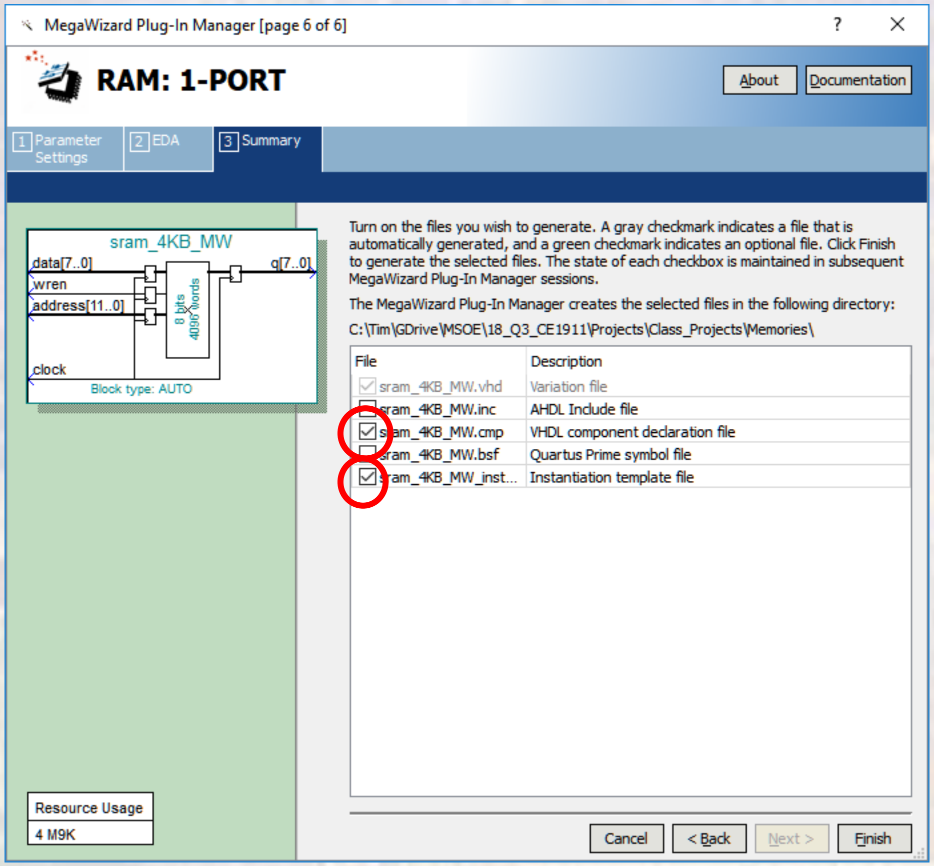
Timing and resource estimation

Generates a netlist for timing and resource estimation for this megafunction. If you are synthesizing your design with a third-party EDA synthesis tool, using a timing and resource estimation netlist can allow for better design optimization.

Not all third-party synthesis tools support this feature - check with the tool vendor for complete support information.

Note: Netlist generation can be a time-intensive process. The size of the design and the speed of your system affect the time it takes for netlist generation to complete.

☐ Generate netlist



Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

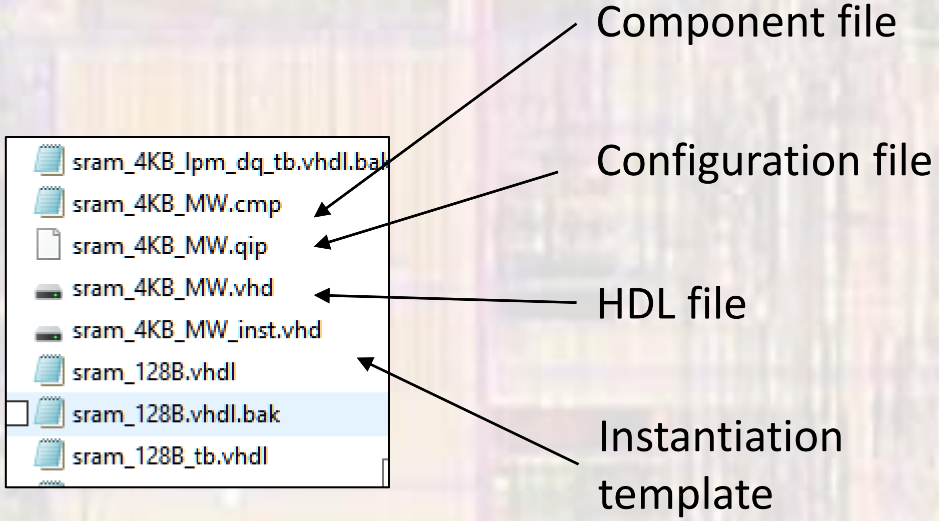
The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\Tim\GDrive\MSOE\18_Q3_CE1911\Projects\Class_Projects\Memories\

File	Description
✓ sram_4KB_MW.vhd	Variation file
☐ sram_4KB_MW.inc	AHDL Include file
✓ sram_4KB_MW.cmp	VHDL component declaration file
☐ sram_4KB_MW.bsf	Quartus Prime symbol file
✓ sram_4KB_MW_inst...	Instantiation template file

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard files


sram_4KB_lpm_dq_tb.vhdl.bak
sram_4KB_MW.cmp
sram_4KB_MW.qip
sram_4KB_MW.vhd
sram_4KB_MW_inst.vhd
sram_128B.vhdl
sram_128B.vhdl.bak
sram_128B_tb.vhdl

MAX10 Memory HDL

- IP Based - RAM
  - MegaWizard files

HDL file

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;

ENTITY sram_4KB_MW IS
    PORT
    (
        address        : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
        clock      : IN STD_LOGIC  := '1';
        data       : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        wren       : IN STD_LOGIC ;
        q        : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END sram_4KB_MW;


ARCHITECTURE SYN OF sram_4kb_mw IS

    SIGNAL sub_wire0  : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

```
BEGIN
    q    <= sub_wire0(7 DOWNTO 0);

    altsyncram_component : altsyncram
    GENERIC MAP (
        clock_enable_input_a => "BYPASS",
        clock_enable_output_a => "BYPASS",
        intended_device_family => "MAX 10",
        lpm_hint => "ENABLE_RUNTIME_MOD=NO",
        lpm_type => "altsyncram",
        numwords_a => 4096,
        operation_mode => "SINGLE_PORT",
        outdata_aclr_a => "NONE",
        outdata_reg_a => "CLOCK0",
        power_up_uninitialized => "FALSE",
        read_during_write_mode_port_a =>
"NEW_DATA_NO_NBE_READ",
        widthad_a => 12,
        width_a => 8,
        width_byteena_a => 1
    )
    PORT MAP (
        address_a => address,
        clock0 => clock,
        data_a => data,
        wren_a => wren,
        q_a => sub_wire0
    );

END SYN;
```

MAX10 Memory HDL

- IP Based – RAM
  - MegaWizard files

Component file

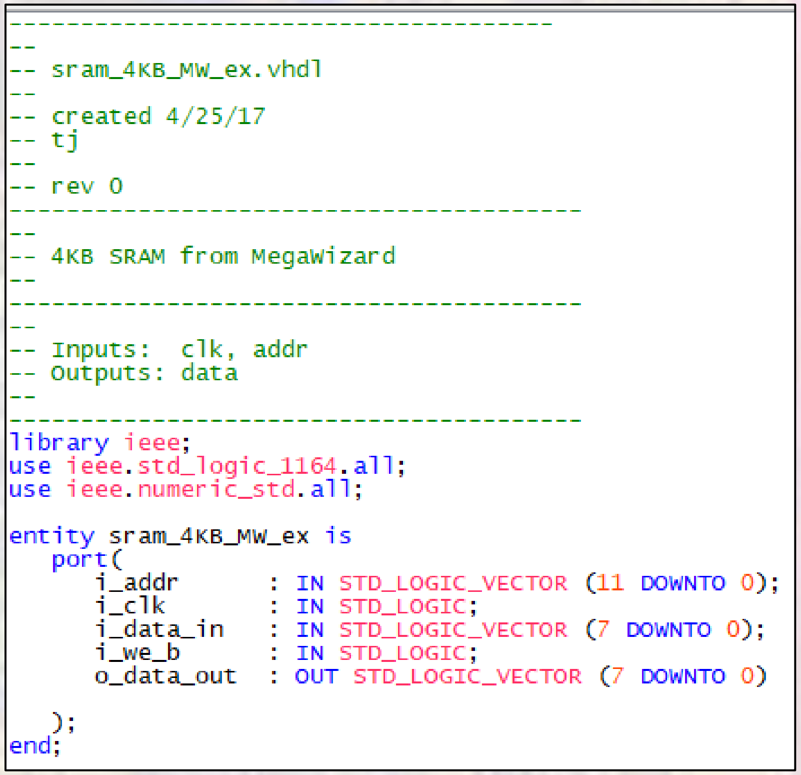
```
component sram_4KB_MW
    PORT
    (
        address   : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
        clock     : IN STD_LOGIC  := '1';
        data      : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        wren      : IN STD_LOGIC ;
        q         : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
end component;
```

Instantiation template

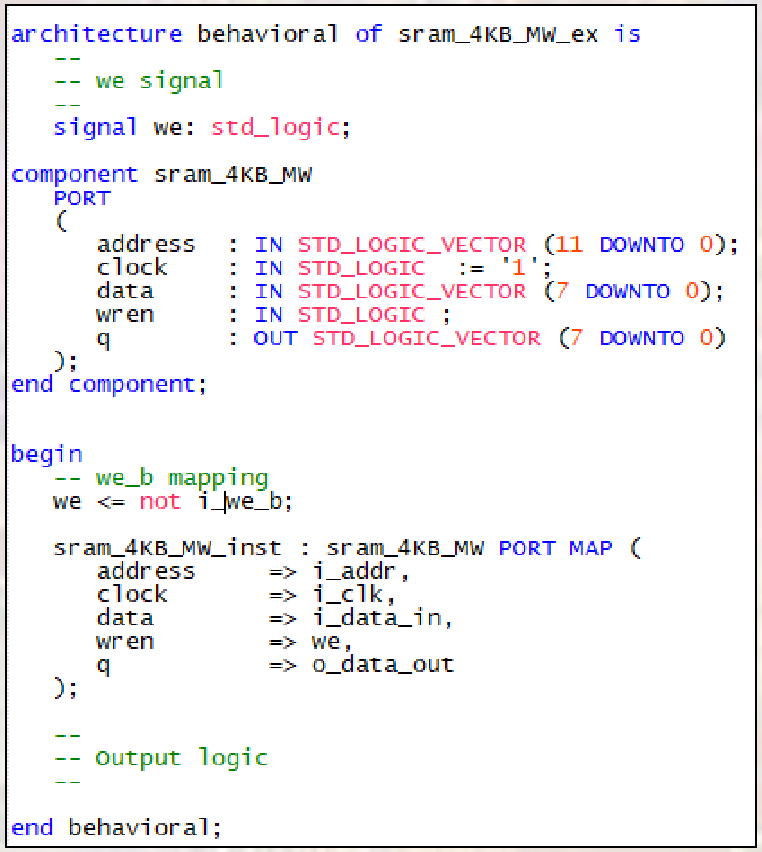
```
sram_4KB_MW_inst : sram_4KB_MW PORT MAP (
        address    => address_sig,
        clock   => clock_sig,
        data    => data_sig,
        wren    => wren_sig,
        q    => q_sig
    );
```

MAX10 Memory HDL

- IP Based - RAM
  - Implementation into our design



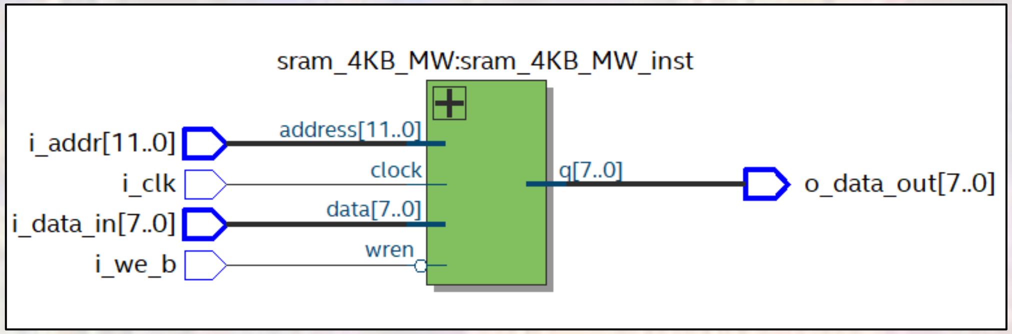
```vhdl
--------------------------------------
--
-- sram_4KB_MW_ex.vhdl
--
-- created 4/25/17
-- tj
--
-- rev 0
----------------------------------------
--
-- 4KB SRAM from MegaWizard
--
----------------------------------------
--
-- Inputs:  clk, addr
-- Outputs: data
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sram_4KB_MW_ex is
   port(
      i_addr       : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
      i_clk        : IN STD_LOGIC;
      i_data_in    : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      i_we_b       : IN STD_LOGIC;
      o_data_out   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)

   );
end;
```

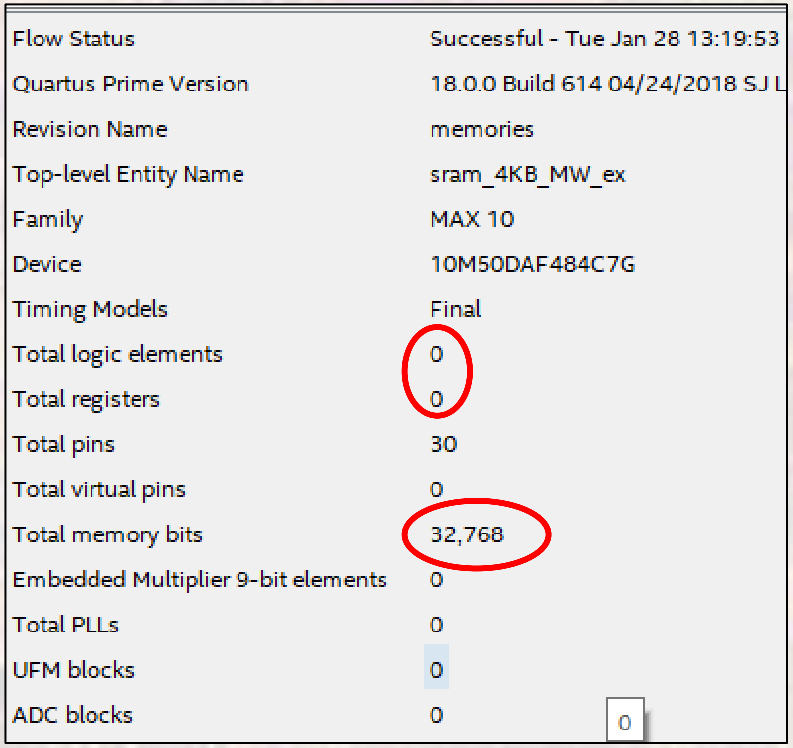


```vhdl
architecture behavioral of sram_4KB_MW_ex is
   --
   -- we signal
   --
   signal we: std_logic;

component sram_4KB_MW
   PORT
   (
      address   : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
      clock     : IN STD_LOGIC   := '1';
      data      : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      wren      : IN STD_LOGIC ;
      q         : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
   );
end component;


begin
   -- we_b mapping
   we <= not i_we_b;

   sram_4KB_MW_inst : sram_4KB_MW PORT MAP (
      address      => i_addr,
      clock        => i_clk,
      data         => i_data_in,
      wren         => we,
      q            => o_data_out
   );

   --
   -- output logic
   --

end behavioral;
```

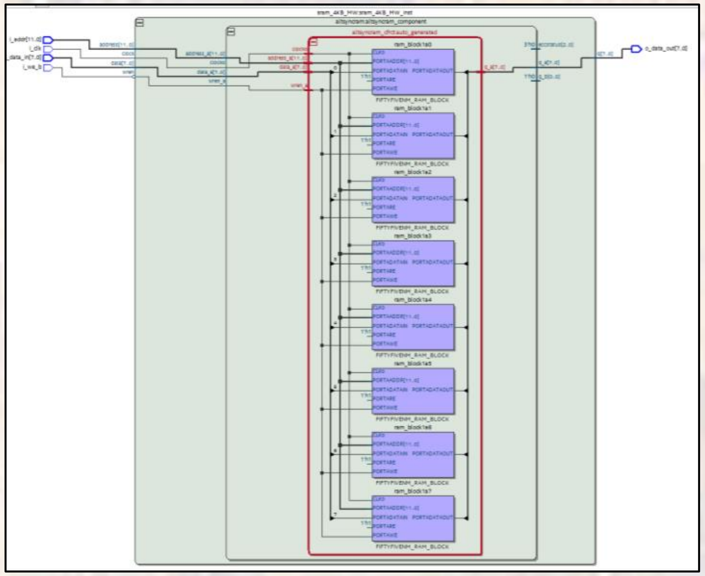
MAX10 Memory HDL

- IP Based - RAM
  - Implementation into our design





Flow Status	Successful - Tue Jan 28 13:19:53
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ L
Revision Name	memories
Top-level Entity Name	sram_4KB_MW_ex
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	0
Total registers	0
Total pins	30
Total virtual pins	0
Total memory bits	32,768
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0



8 – ½ M9K blocks (1 word bit / block)