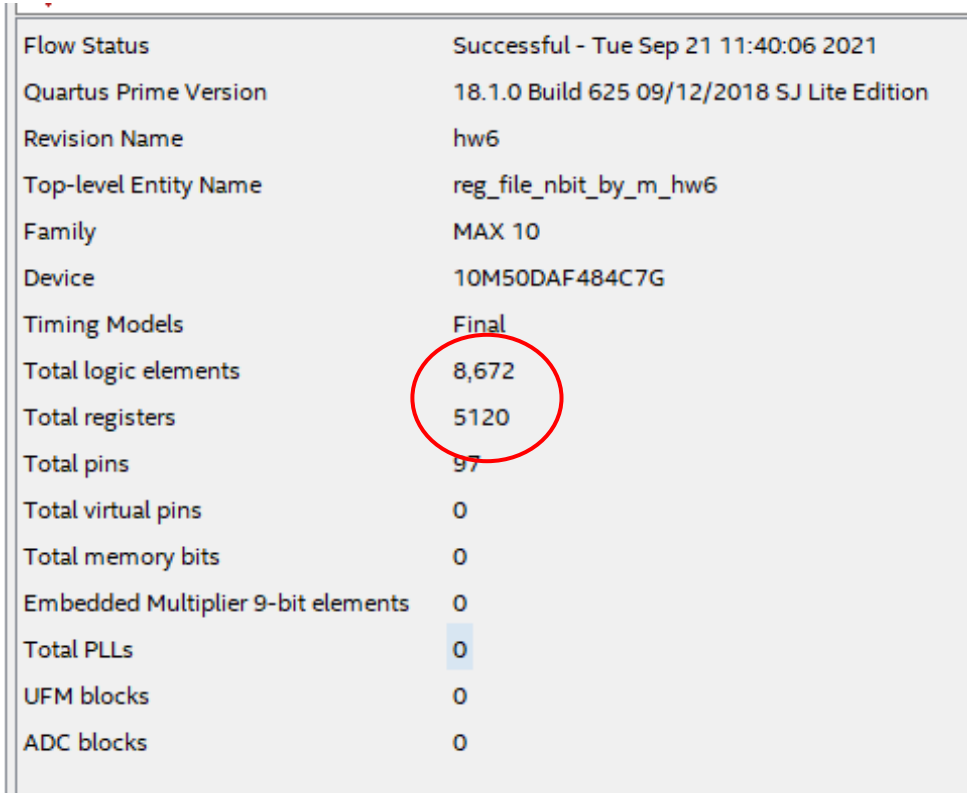
EE 3921

Dr. Johnson

Homework 6

1 – FF Based memory: Modify the design from class (Max10 Memory HDL notes – page 6) to create a 128 word, 40 bit memory. Write down every line of code that must be modified. Compile your design and include the compilation report / flow summary with the number of LE's highlighted. 10pts

```
generic( REG_NUM:          natural  := 128;
         REG_DATA_WIDTH:   natural  := 40
```



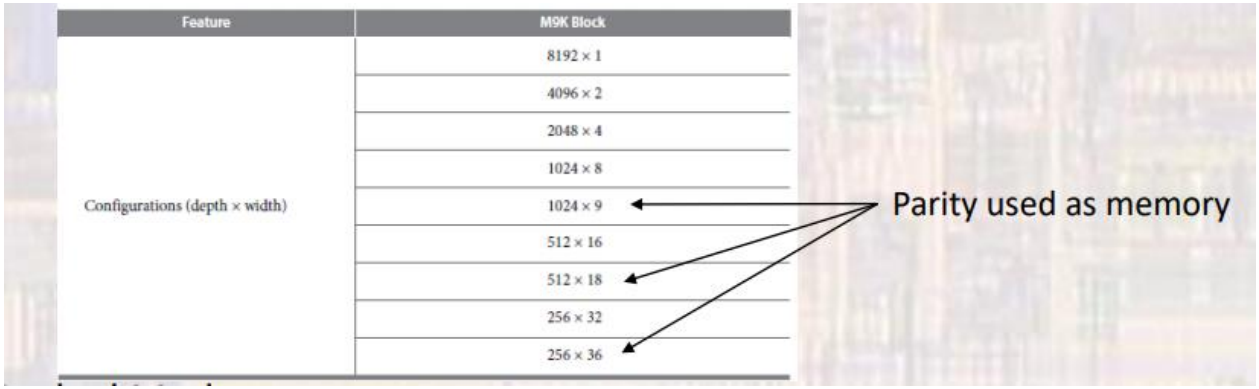
Flow Status	Successful - Tue Sep 21 11:40:06 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	hw6
Top-level Entity Name	reg_file_nbit_by_m_hw6
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	8,672
Total registers	5120
Total pins	97
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

2A – How many onboard memory blocks would be required to support the same size memory as the previous problem? 10pts

128 x 40 = 5120 < one M9K block

2B – What is the largest size 32bit word memory that can fit in a single on chip memory block? 10pts

M9K has extra 1K bits attached to the word width, not the # of words → 8Kb → 1KB → 256W in x32 configuration

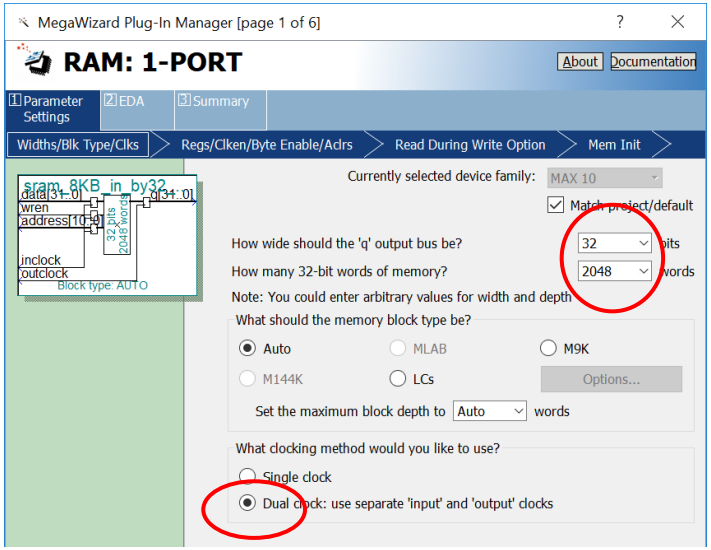


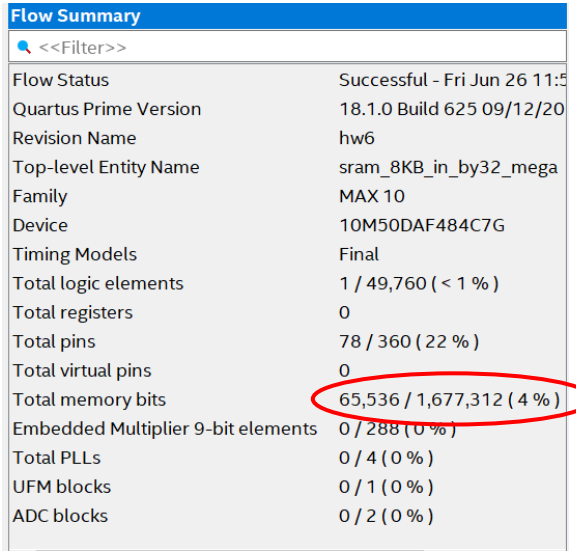
3 – Implement a dual clock 8KB SRAM in by-32 configuration using the Mega Wizard 50pts

Compile your HDL and provide the code showing the address/data widths and clocks

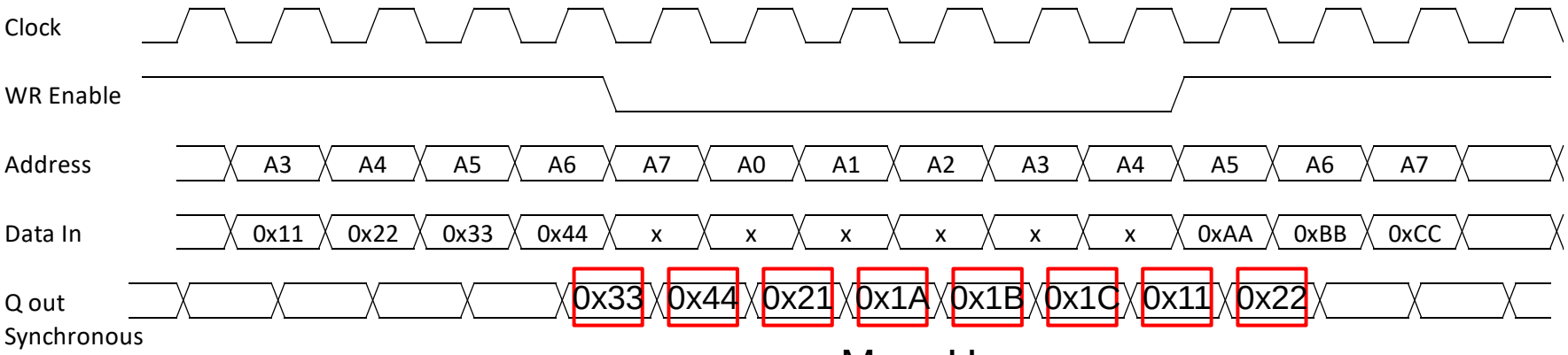
Provide the compilation summary showing the number of memory bits used

```
-----------------------------------------
--
-- sram_8KB_in_by32_mega.vhdl
--
-- created 4/25/17
-- tj
--
-- rev 0
-----------------------------------------
--
-- 8KB in x32 dq RAM from mega library
--
-----------------------------------------
--
-- Inputs: clk, addr, data_in, we_b
-- Outputs: data_out
--
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sram_8KB_in_by32_mega is

    port(
          i_clk_in:   in std_logic;
          i_clk_out:  in std_logic;
          i_we_b:     in std_logic;
          i_addr:     in std_logic_vector(10 downto 0);
          i_data_in:  in std_logic_vector(31 downto 0);
          o_data_out: out std_logic_vector(31 downto 0)
    );
end;

architecture behavioral of sram_8KB_in_by32_mega is
--
-- invert we
--
    signal we_sig: std_logic;

    component SRAM_8KB_in_by32
    PORT(
       address     : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
       data        : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
       inclock     : IN STD_LOGIC  := '1';
       outclock    : IN STD_LOGIC ;
       wren        : IN STD_LOGIC ;
       q           : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
    );
    end component;

    begin
       we_sig <= not i_we_b;

       SRAM_8KB_in_by32_inst : SRAM_8KB_in_by32
          PORT MAP (
             address     => i_addr,
             data        => i_data_in,
             inclock     => i_clk_in,
             outclock    => i_clk_out,
             wren        => we_sig,
             q           => o_data_out
          );

end behavioral;
```





4 – Simple single port memory: Given the timing diagram below for the simple single port memory, fill in the values for the empty boxes on the diagram and the final memory values. Assume a new data output model 20pts



	Mem Hex BEFORE	Mem Hex AFTER
A0	0x1A	0x1A
A1	0x1B	0x1B
A2	0x1C	0x1C
A3	0x1D	0x11
A4	0x1E	0x22
A5	0x1F	0xAA
A6	0x10	0xBB
A7	0x21	0xCC