VHDL Memories MUX Based

Last updated 3/26/25

VHDL Memories – Mux Based

- Four major VHDL memory solutions
  - Mux based
    - Only applicable for ROMs
  - FlipFlop based
    - Very large – only acceptable for very small memories
  - Inferred
    - Memory is implemented in a pre-built memory block
      - Memory block must exist in the platform
      - Tightly coupled memory – small but very fast
      - General memory – large and not as fast
  - External
    - The memory interface is implemented
    - The memory itself is a separate chip

VHDL Memories – Mux Based

N words x M bits/word

N array elements x SLV

- VHDL solution for memories
  - An array of std_logic_vectors
  - Coded just like the non-optimized long array of data words
  - Array construct
    - New type, that has array type as its basis

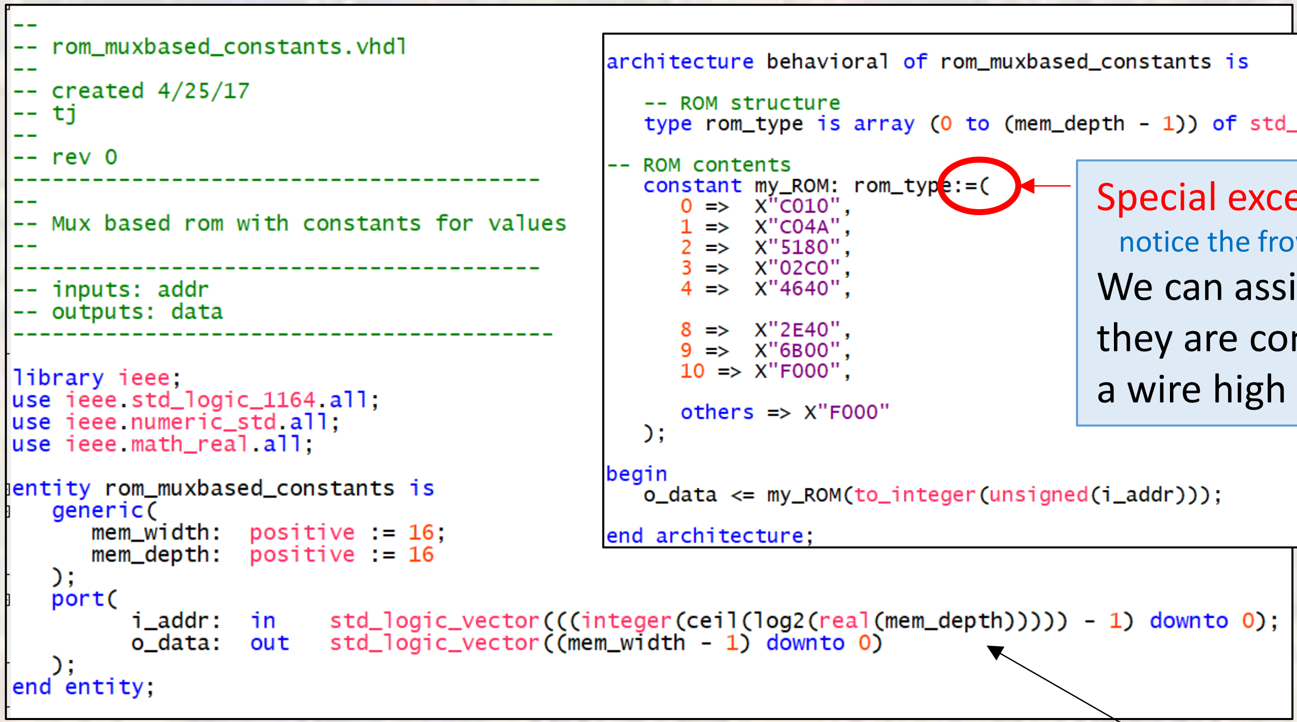
      type my_new_type is array (0 to depth) of some_vhdl_type
  - Memory construct
    - Uses std_logic_vector
      - No understanding of the values (signed/unsigned) is assumed, just bits

```
type my_memory is array (0 to depth) of std_logic_vector((wordwidth - 1) downto 0);
```

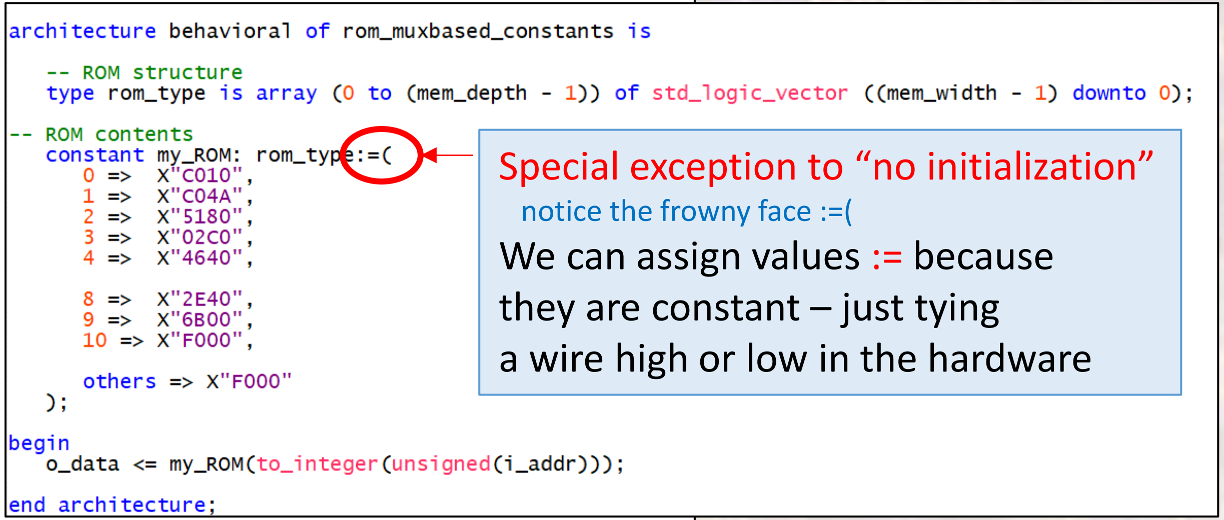
VHDL Memories – Mux Based

- ROM – mux based
  - Read only
  - Memory values stored as constants

16 word, 16b/w (2B/w) ROM



```
--
-- rom_muxbased_constants.vhdl
--
-- created 4/25/17
-- tj
--
-- rev 0
----------------------------------------
--
-- Mux based rom with constants for values
--
----------------------------------------
-- inputs: addr
-- outputs: data
----------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity rom_muxbased_constants is
    generic(
        mem_width:  positive := 16;
        mem_depth:  positive := 16
    );
    port(
            i_addr:  in     std_logic_vector(((integer(ceil(log2(real(mem_depth))))) - 1) downto 0);
            o_data:  out    std_logic_vector((mem_width - 1) downto 0)
    );
end entity;
```



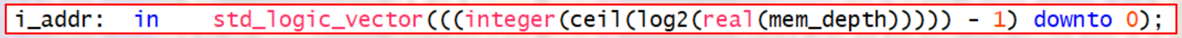
```
architecture behavioral of rom_muxbased_constants is

   -- ROM structure
   type rom_type is array (0 to (mem_depth - 1)) of std_logic_vector ((mem_width - 1) downto 0);

-- ROM contents
   constant my_ROM: rom_type:=(
       0 =>   X"C010",
       1 =>   X"C04A",
       2 =>   X"5180",
       3 =>   X"02C0",
       4 =>   X"4640",

       8 =>   X"2E40",
       9 =>   X"6B00",
       10 =>  X"F000",

       others => X"F000"
   );

begin
   o_data <= my_ROM(to_integer(unsigned(i_addr)));

end architecture;
```

Special exception to "no initialization"
notice the frowny face :=(
We can assign values := because they are constant – just tying a wire high or low in the hardware

Calculating the # of address bits based on the mem-depth
- see next slide

VHDL Memories – Mux Based

- ROM – mux based
  - Address bit calculation



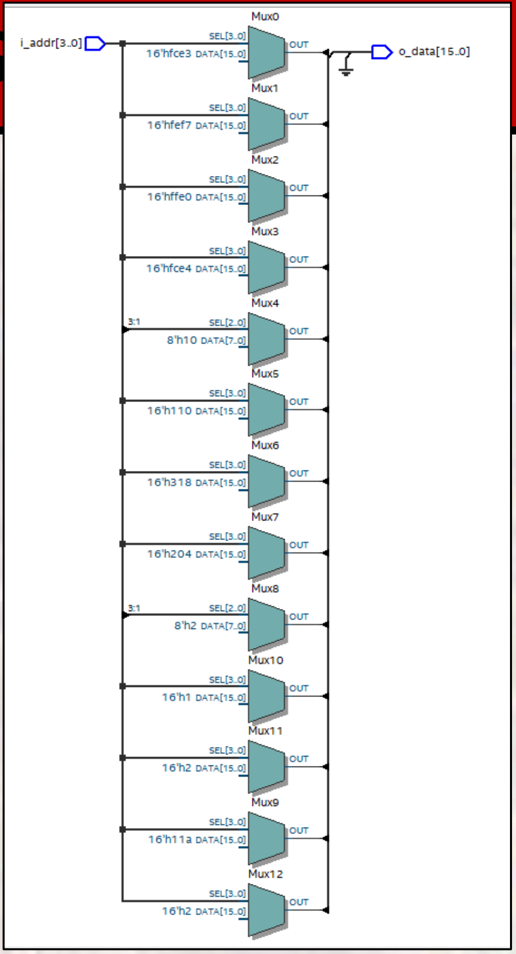
```
i_addr:  in     std_logic_vector(((integer(ceil(log2(real(mem_depth)))) - 1) downto 0);
```

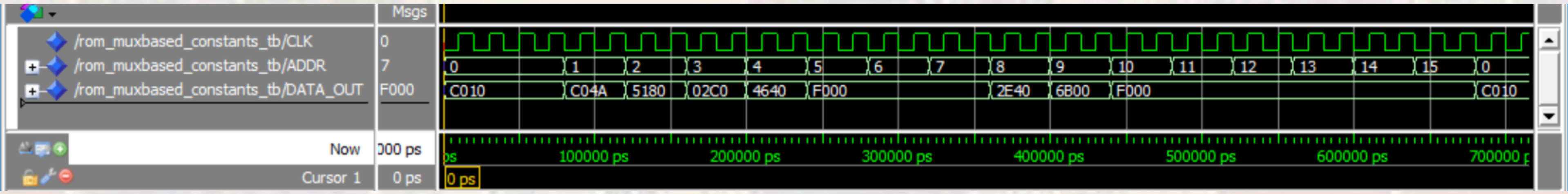
mem_depth	only makes sense to be an integer
real(mem_depth)	turns it into a real number (not an int)
log2(real(mem_depth))	calculates the log base 2 requires a real input provides a real output
ceil(log2(real(mem_depth)))	rounds up (next largest whole real number) provides support for non-2^N sizes 24 → 4.585 → 5.0
integer(ceil(log2(real(mem_depth))))	converts the real to an integer must be integer to use as index

VHDL Memories – Mux Bas

- ROM – mux based
  - Memory values stored as constants

Modelled as 1 mux per bit in the word





VHDL Memories – Mux Based

- Memory Test Benches
  - A proper memory testbench would test:
    - All addresses