RAMs Models in VHDL

A data type definition that is useful is constructing an array of std_logic_vectors. This can be used as a RAM as illustrated in the example model given below:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity RAM is
        generic (K: integer:=8;         -- number of bits per word
                W: integer:=8);         -- number of address bits
        port (WR: in std_logic;         -- active high write enable
                ADDR: in std_logic_vector (W-1 downto 0);        -- RAM address
                DIN: in std_logic_vector (K-1 downto 0);         -- write data
                DOUT: out std_logic_vector (K-1 downto 0));      -- read data
end entity RAM;
architecture RAMBEHAVIOR of RAM is
subtype WORD  is std_logic_vector ( K-1 downto 0);       -- define size of WORD
type MEMORY is array (0 to 2**W-1) of WORD;              -- define size of MEMORY
signal RAM256: MEMORY;          -- define RAM256 as signal of type MEMORY
begin
process (WR, DIN, ADDR)
variable RAM_ADDR_IN: integer range 0 to 2**W-1;        -- to translate address to integer
variable STARTUP: boolean :=true;                        -- temporary variable for initialization
        begin
        if (STARTUP = true) then      -- for initialization of RAM during start of simulation
                RAM256 <=  (0 => "00000101",      -- initializes first 5 locations in RAM
                            1 => "00110100",      -- to specific values
                            2 => "00000110",      -- all other locations in RAM are
                            3 => "00011000",      -- initialized to all 0s
                            4 => "00000011",
                            others => "00000000");
                DOUT <= "XXXXXXXX";   -- force undefined logic values on RAM output
                STARTUP :=false;      -- now this portion of process will only execute once
        else
                RAM_ADDR_IN := conv_integer (ADDR);           -- converts address to integer
                if (WR='1') then                              -- write operation to RAM
                        RAM256 (RAM_ADDR_IN) <= DIN ;
                end if;
                DOUT <= RAM256 (RAM_ADDR_IN);     -- always does read operation
        end if;
end process;
end architecture RAMBEHAVIOR;
```

The model above is technology independent and good for simulation. It is parameterized such that it can easily be used to emulate many RAMs of various sizes with similar behavior. It can now be used as the basis for simulation and verification of other hierarchical functions such as a First-In-First-Out (FIFO) memory. However, this model is not usually good for synthesis. For example, the model will synthesize in ISE but it requires 1736 slices in a Spartan 3 200 FPGA (see Figure 1) but can easily fit in only one block RAM in the device illustrated in the upper left hand corner in Figure 1 and only requires 2K of the 18K total memory space in that block RAM. Therefore, one would want to instantiate the Xilinx specific block RAM model prior to final synthesis in order to obtain the more efficient (and higher performance) implementation of a block RAM. While the Xilinx specific block RAM model can be used for simulation as well as synthesis, it is not technology dependent and requires

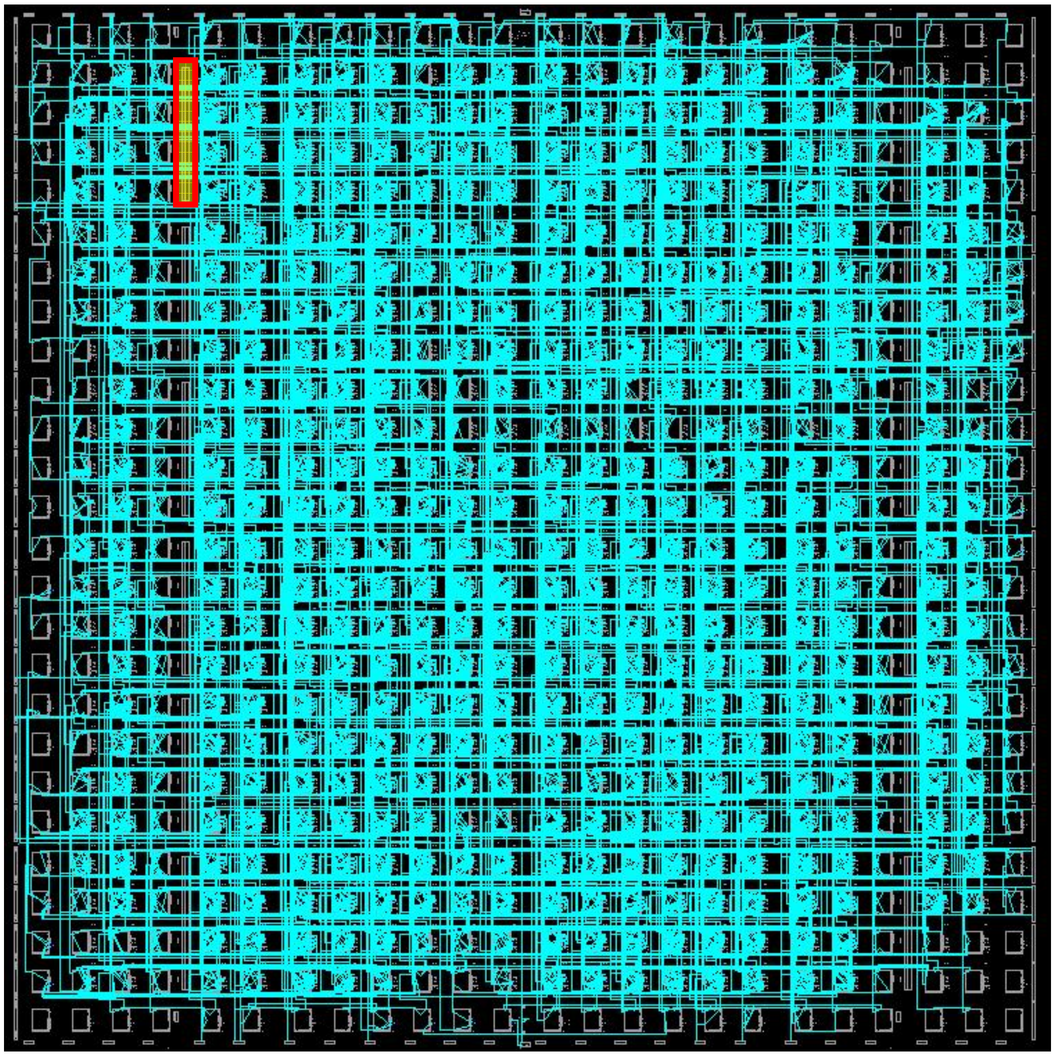

Figure 1. RAM model synthesized in flip-flops of FPGA.