EE3810 – Lecture 20

Using Higher-Level Xilinx Features And Primitives

Using More Advanced Features

- The Xilinx FPGA Devices are extremely flexible.
- Some contain internal RAM, Clock Controllers, Multipliers and other features.
- Some, such as the Virtex Series of parts, can contain internal microprocessors (PowerPC).

Spartan-3 Capabilities

- The Spartan-3 family contains many devices:

Spartan-3	XC 3S50	XC 3S200	XC 3S400	XC 3S1000	XC 3S1500	XC 3S2000	XC 3S4000	XC 3S5000
EasyPath cost reduction	–	–	–	–	XCE 3S1500	XCE 3S2000	XCE 3S4000	XC 3S5000
System Gates	50K	200K	400K	1000K	1500K	2000K	4000K	5000K
Logic Cells	1,728	4,320	8,064	17,280	29,952	46,080	62,208	74,880
18x18 Multipliers	4	12	16	24	29,952	40	96	104
Block RAM Bits	72K	216K	288K	432K	576K	720K	1,728K	1,872K
Distributed RAM Bits	12K	30K	56K	120K	208K	320K	432K	520K
DCMs	2	4	4	4	4	4	4	4
I/O Standards	24	24	24	24	24	24	24	24
Max Differential I/O Pairs†	56	76	116	175	221	270	312	344
Max Single Ended I/O†	124	173	264	391	487	565	712	784

Making Bigger RAMs

- There are 216K bits of block RAM on the device we're using.
- This RAM is divided into twelve sections of 18K-bits each.
- The configuration is arbitrary in each block:
  - 2304 x 8, 1152 x 16, 576 x 32
  - 3104 x 41

Model to Infer Block RAM

```
entity blockram1 is
    Port ( clk : in  STD_LOGIC;
           we : in  STD_LOGIC;
           en : in  STD_LOGIC;
           addr : in  STD_LOGIC_VECTOR (4 downto 0);
           di : in  STD_LOGIC_VECTOR (3 downto 0);
           do : out  STD_LOGIC_VECTOR (3 downto 0));
end blockram1;

architecture Behavioral of blockram1 is
   type ram_type is array (31 downto 0) of std_logic_vector (3 downto 0);
   signal ram : ram_type;
begin
   process( clk )
   begin
      if clk'event and clk = '1' then
         if en = '1' then
            if we = '1' then
               ram(conv_integer(addr)) <= di;
            end if;
            do <= ram(conv_integer(addr));
         end if;
      end if;
   end process;
end Behavioral;
```

Synthesis Results

- We see that ISE was able to infer block ram:

```
Synthesizing Unit <blockram1>.
    Related source file is "C:/blockram1/blockram1.vhd".
    Found 32x4-bit single-port block RAM for signal <ram>.
    -----------------------------------------------------------------------
    | mode                | read-first                        |           |
    | aspect ratio        | 32-word x 4-bit                   |           |
    | clock               | connected to signal <clk>         | rise      |
    | enable              | connected to signal <en>          | high      |
    | write enable        | connected to signal <we>          | high      |
    | address             | connected to signal <addr>        |           |
    | data in             | connected to signal <di>          |           |
    | data out            | connected to signal <do>          |           |
    | ram_style           | Auto                              |           |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 RAM(s).
Unit <blockram1> synthesized.
```

Block RAM is Synchronous

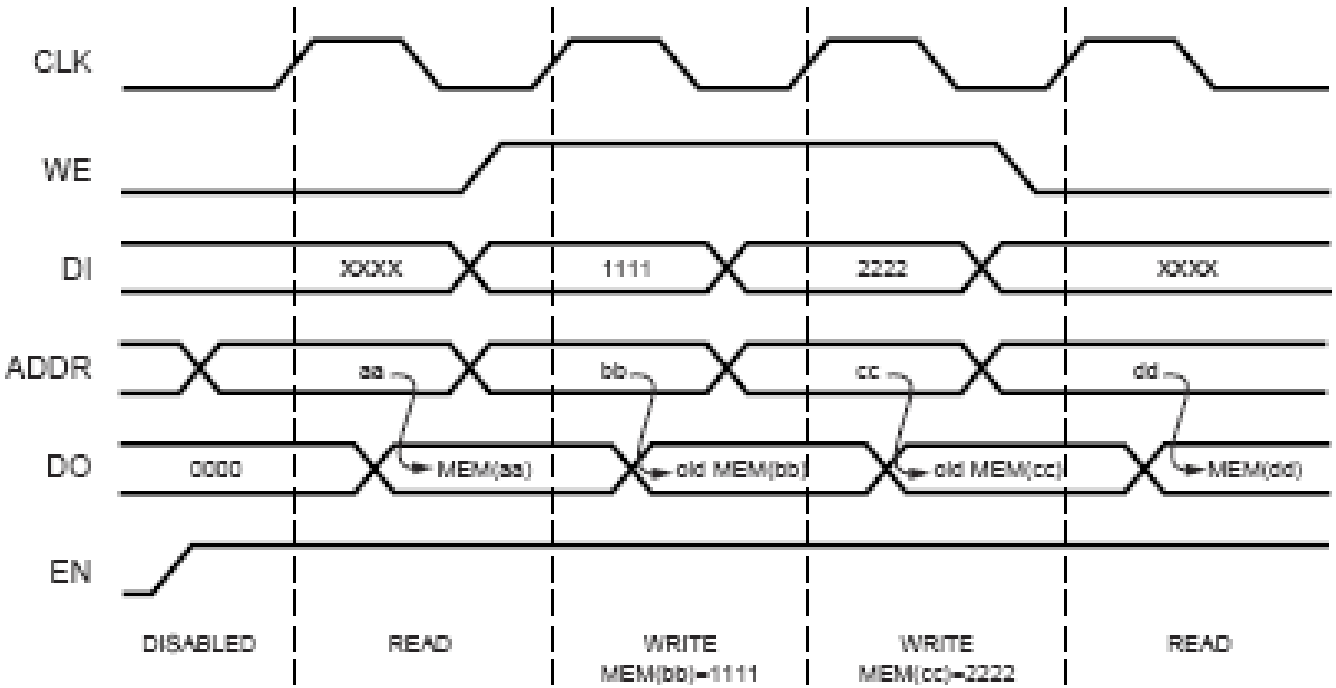
- Block RAM behaves a little differently than “normal” RAM.
  - Reading and writing Block RAM is synchronous
  - Several modes are defined:
    - Write-first
    - Read-first
    - No-change

The Read-First Mode

- In the read-first mode reads occur as expected whenever the device is enabled and clocked.
- In the write-mode, new data is written to the block RAM, but during the write the previous content of the address is put on the data outputs.

Read-First Mode Timing

- Our first example was of the read-first mode:



The Write-First Mode

- In the write-first mode reads occur as expected whenever the device is enabled and clocked.
- In the write-mode, new data is output to the data outputs at the same time that the data is written to the block RAM.

Synthesizing Write-First Mode

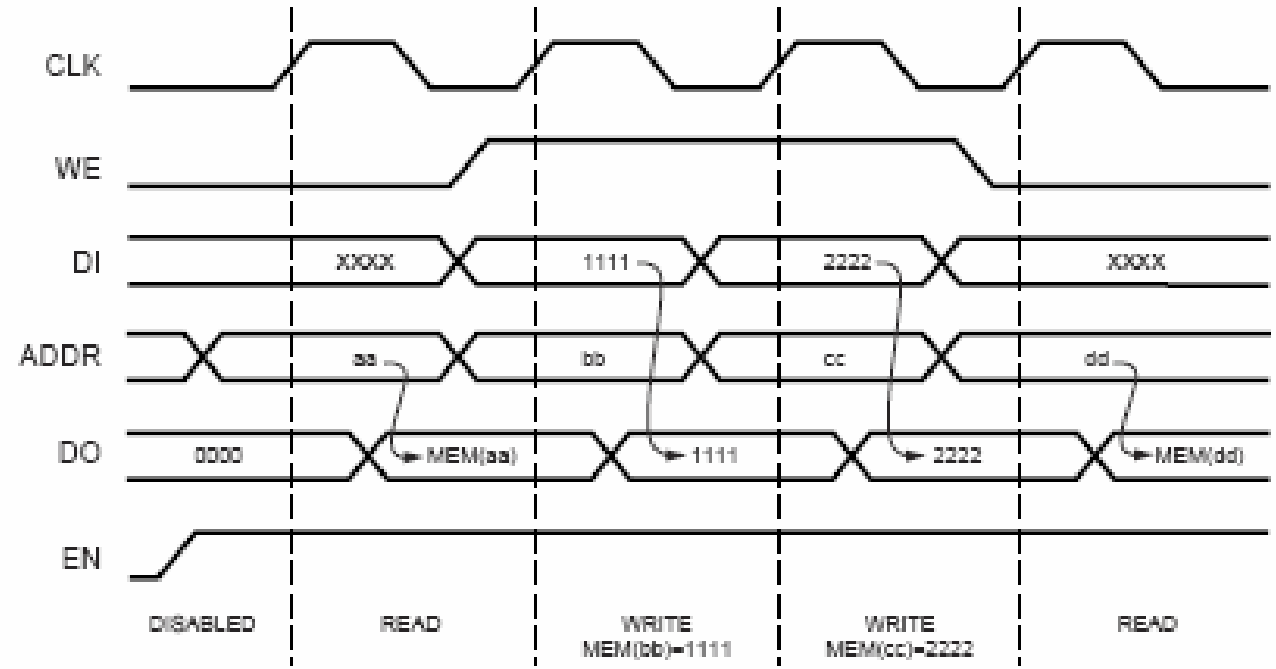
```
entity blockram2 is
    Port ( clk, we, en : in  STD_LOGIC;
           addr : in  STD_LOGIC_VECTOR (4 downto 0);
           di : in  STD_LOGIC_VECTOR (3 downto 0);
           do : out  STD_LOGIC_VECTOR (3 downto 0));
end blockram2;

architecture Behavioral of blockram2 is
   type ram_type is array (31 downto 0) of std_logic_vector (3 downto 0);
   signal ram : ram_type;
begin
   process( clk )
   begin
      if clk'event and clk = '1' then
         if en = '0' then
            if we = '0' then
               ram(conv_integer(addr)) <= di;
               do <= di;
            else
               do <= ram(conv_integer(addr));
            end if;
         end if;
      end if;
   end process;
end Behavioral;
```

Inferring Write-First Mode

```
Synthesizing Unit <blockram2>.
    Related source file is "C:/blockram2/blockram1.vhd".
    Found 32x4-bit single-port block RAM for signal <ram>.
    -----------------------------------------------------------------------
    | mode                | write-first                       |           |
    | aspect ratio        | 32-word x 4-bit                   |           |
    | clock               | connected to signal <clk>         | rise      |
    | enable              | connected to signal <en>          | low       |
    | write enable        | connected to signal <we>          | low       |
    | address             | connected to signal <addr>        |           |
    | data in             | connected to signal <di>          |           |
    | data out            | connected to signal <do>          |           |
    | ram_style           | Auto                              |           |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 RAM(s).
Unit <blockram2> synthesized.
```

Write-First Mode Timing

- Note the subtle difference between this mode and the read-first mode:



The No-Change Mode

- In the no-change mode reads occur as expected whenever the device is enabled and clocked.
- In the write-mode, the data outputs latch whatever data was on the data input bus just before write enable is active.

Synthesizing No-Change Mode

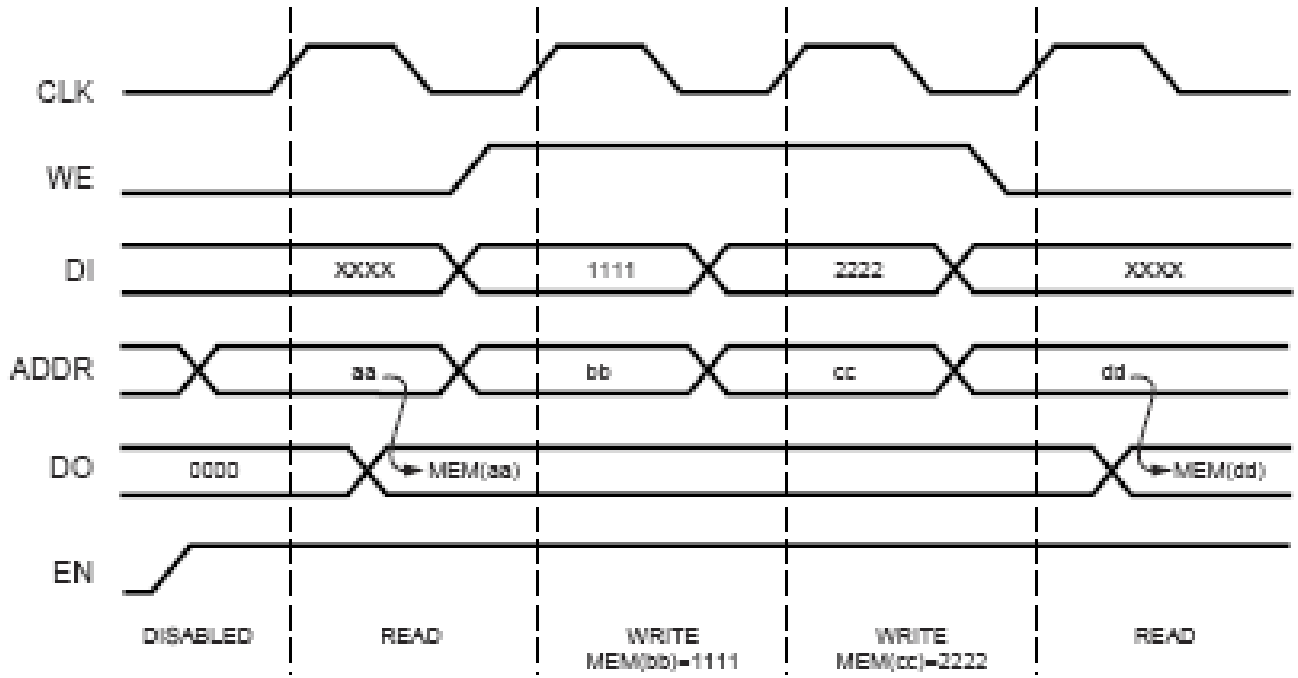
```
entity blockram2 is
    Port ( clk, we, en : in  STD_LOGIC;
           addr : in  STD_LOGIC_VECTOR (4 downto 0);
           di : in  STD_LOGIC_VECTOR (3 downto 0);
           do : out  STD_LOGIC_VECTOR (3 downto 0));
end blockram2;

architecture Behavioral of blockram2 is
   type ram_type is array (31 downto 0) of std_logic_vector (3 downto 0);
   signal ram : ram_type;
begin
   process( clk )
   begin
      if clk'event and clk = '1' then
         if en = '0' then
            if we = '0' then
               ram(conv_integer(addr)) <= di;
            else
               do <= ram(conv_integer(addr));
            end if;
         end if;
      end if;
   end process;
end Behavioral;
```

Inferring No-Change Mode

```
Synthesizing Unit <blockram3>.
    Related source file is "C:/blockram3/blockram1.vhd".
    Found 32x4-bit single-port block RAM for signal <ram>.
    -----------------------------------------------------------------------
    | mode                | no-change                             |        |
    | aspect ratio        | 32-word x 4-bit                       |        |
    | clock               | connected to signal <clk>             | rise   |
    | enable              | connected to signal <en>              | high   |
    | write enable        | connected to signal <we>              | high   |
    | address             | connected to signal <addr>            |        |
    | data in             | connected to signal <di>              |        |
    | data out            | connected to signal <do>              |        |
    | ram_style           | Auto                                  |        |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 RAM(s).
Unit <blockram3> synthesized.
```

No-Change Mode Timing

- Note the subtle difference between this mode and the other modes:



Other Xilinx Primitives

- The Spartan-3 also contains twelve 18x18 multipliers.
- These share some I/O with the block RAM, so certain configurations are limited when using both.
- The hardware multipliers allow increasing speed and reduce gate utilization.

The MULT18X18 Primitive

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity comb_multiplier is
   Port ( Ain, Bin : in  STD_LOGIC_VECTOR (17 downto 0);
          Qout : out  STD_LOGIC_VECTOR (35 downto 0));
end comb_multiplier;

architecture Behavioral of comb_multiplier is
component MULT18X18
        port ( P : out STD_LOGIC_VECTOR (35 downto 0);
        A : in STD_LOGIC_VECTOR (17 downto 0);
        B : in STD_LOGIC_VECTOR (17 downto 0));
end component;
begin
MULT: MULT18X18 port map (P => Qout, A => Ain, B => Bin);
end Behavioral;
```

Simulating the Multiplier

- Clearly ISE was able to find and utilize the internal multiplier: