VHDL Synthesis

Xilinx VHDL Modeling Hints

Defining Performance

- Depends upon the situation
  1. Minimal Area
  2. Fastest Design
  3. Low Power Consumption
  4. Ease of Maintenance
  5. ... and so on....

Technology Dependent

- In FPGAs, minimizing GATES doesn't necessarily minimize area
  - Want to minimize CLBs
- Minimizing logic delays may not always mean fewest logic gates
  - Want fewest layers of CLBs

Improving Design Performance

- Modifying code
  - Reduce levels of logic to improve timing
  - Redefine hierarchical boundaries to help the compiler optimize design logic
  - Pipeline
  - Eliminate/introduce logic replication
  - Use resource sharing

Placing and Routing Options

- Place and Route Effort Level (right click on "Place and Route", select "Properties")
  - Low (Fastest run time) to Highest (Best results)
- Number of Router Passes
- Multi-pass option
- Re-entrant routing option

Sources of Information

- Xilinx Synthesis and Simulation Design Guide
- Synplicity documents
FPGA Reserved Words

- Don’t use CCLK, DP, GND, VCC, and RST
- CLB names such as AA, AB, and R1C2
- Primitive names such as TD0, BSCAN, M0, M1, M2, or STARTUP
- Do not use pin names such as P1 and A4 for component names
- Do not use pad names such as PAD1 for component names

Naming Identifiers, Types, and Packages

- Maximum of 256 characters per name
- Character Restrictions
  - Alphabetic
  - Numeric
  - Underscores
  - At least one non-numeric character
- Use meaningful names
- Make use of optional names for flow control constructs (such as processes)
- Use NAMED ASSOCIATION lists

Constants

- Give all constants a name and use the name instead of the numerical constant in all expressions.
  - Better documentation
  - Easier to change the constant value if necessary
  - Use of constants instead of variables will result in less complex designs

Order and Group Arithmetic Functions

- ADD <= A1 + A2 + A3 + A4;

Order and Group Arithmetic Functions

- ADD <= (A1 + A2) + (A3 + A4);

Inferring Block RAM

- Xilinx manual is wrong:

```
architecture rtl of ram_example1 is
type mem_array is array (mem_depth-1 downto 0) of std_logic_vector
    (data_width-1 downto 0);
signal mem: mem_array;
attribute syn_ramstyle : string;
attribute syn_ramstyle of mem : signal is "block_ram";
signal raddress : std_logic_vector(address_width-1 downto 0);
begin
l0: process (clk) begin
    if (clk = '1' and clk'event) then
        if (we = '1') then
            mem(CONV_INTEGER(address)) <= data;
        end if;
    end if;
end process;
end rtl;
```

Address must be a register, i.e. synchronous with clock!

144 LUTs (6%), 0 Block RAM
Inferring Block RAM

- Right way—make address a register:
  architecture rtl of ram_example is
  type mem_array is array (mem_depth-1 downto 0) of std_logic_vector
  (data_width-1 downto 0);
  signal mem : mem_array;
  attribute syn_ramstyle : string;
  attribute syn_ramstyle of ram : signal is "block_ram";
  signal raddress : std_logic_vector(address_width-1 downto 0);
  begin
  i: process (clk) begin
      if (clk = '1' and clk'event) then
          raddress <= address;
          if (we = '1') then
              mem(CONV_INTEGER(address)) <= data;
          end if;
      end if;
  end process;
  q <= mem(CONV_INTEGER(raddress));
  end rtl;

Multiplier

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity xcv2_mult18x18 is
Port (a : in std_logic_vector(7 downto 0);
b : in std_logic_vector(7 downto 0);
clk : in std_logic;  prod : out std_logic_vector(15 downto 0))
end xcv2_mult18x18;
architecture arch_xcv2_mult18x18 of xcv2_mult18x18 is
begin
  process(clk) is begin
      if clk'event and clk = '1' then
          prod <= a*b;
      end if;
  end process;
end arch_xcv2_mult18x18;

Resource Sharing

- Advantages
  • Reduces gate count
  • Reduces routing congestion
  • Reduces design area (or number of design components)
- Disadvantages
  • Increases critical path delay

Resource Sharing

architecture BEHAV of RES_SHARING is
begin
P1: process (A1,B1,C1,D1,COND_1)
begin
    if (COND_1='1') then
        Z1 <= A1 + B1;
    else
        Z1 <= C1 + D1;
    end if;
end process; -- end P1
end BEHAV;
Resource Sharing

- Operations in the same process use resource sharing as the default option.
- CASE statement vs. IF statement
- Operations in separate processes do not use resource sharing as the default option.
- Resource sharing (or not resource sharing) can be forced by selecting an option in most synthesis tools.

Comparing Resource Sharing

<table>
<thead>
<tr>
<th>Item</th>
<th>RS Gates</th>
<th>NRS Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>F/G</td>
<td>19</td>
<td>28</td>
</tr>
<tr>
<td>H</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>FCLB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay</td>
<td>27.9 ns</td>
<td>47.0 ns</td>
</tr>
</tbody>
</table>

Encoding State Machines (State Assignment)

- Traditional methods
  - Highly encoded states (Binary encoding)
  - Minimum number of flip-flops
  - Wide combinational functions
  - Acceptable for PALs and Gate Arrays
  - Not optimal for most FPGAs
    - Many flip flops
    - Narrow combinational functions

- For FPGAs
  - Many flip-flops
  - Narrow combinational functions
  - One-hot Encoding is often better for large state machines (more than 5 states)
  - Binary encoding is usually more efficient for state machines with 5 or fewer states.

State Machine Encoding in Synplicity

- FSM encoding default depends on number of states
- Change setting when compiling by
  - Options->Configure VHDL Compiler->VHDL
- Available options
  - One-Hot
  - Gray
  - Sequential
- Can specify in code with syn_enum_encoding and syn_encoding (see Synpify Reference Manual for details)

State Machine Encoding in Synplicity

- Message in log file using default settings:
  *C:\tom\courses\4514f02\assignments\project5\simple_reader_fsm.vhd*" Trying to extract state machine for register state_r
  Extracted state machine for register state_r
  State machine has 5 reachable states with original encodings of:
  00000
  00010
  00100
  01000
  10000
- Message in log file using sequential setting:
  State machine has 5 reachable states with original encodings of:
  000
  001
  010
  011
  100
Summary

- More synthesis examples
- Synthesis of finite state machines