Logic and Computer Design Fundamentals

VHDL

Part 5 – Chapter 8 – Algorithmic State Machine Example: Binary Multiplier

Charles Kime & Thomas Kaminski

Overview

- **Part 1 - VHDL Basics and Types of Descriptions**
- **Part 2 - Behavioral and Hierarchical Description**
- **Part 3 - Finite State Machines**
- **Part 4 - Registers and Counters**
- **Part 5 - Algorithmic State Machine Example: Binary Multiplier**
  - **Conversion of ASM into VHDL Description**
    - **Decomposition**
      - **Sequencing – Synchronous process for FSM state, combinational process for FSM next state**
      - **Register transfers -synchronous process dependent on FSM state and FSM inputs**
    - **Use of `if,then,else` for scalar decision**
    - **Use of `case` for vector decisions**
  - **Illustration using binary multiplier ASM chart in Figure 8-8 of text**

VHDL for Alternative Binary Multiplier - 1

```
-- Alternate Binary Multiplier with n = 4
-- See Figure 8-8 of text
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity alt_bin_multiplier is
 port(CLK, RESET, G, LOADB, LOADQ: in std_logic;
      MULT_IN: in std_logic_vector(3 downto 0);
      MULT_OUT: out std_logic_vector(7 downto 0));
end alt_bin_multiplier;
architecture sequential of alt_bin_multiplier is
type state_type is (IDLE, MUL);
signal state, next_state: state_type;
signal P: std_logic_vector(1 downto 0);
signal A,B,Q: std_logic_vector(3 downto 0);
signal Z: std_logic;
```

VHDL for Alternative Binary Multiplier - 2

```
begin
-- Test P for value 0:
Z <= not(P(1) or P(0));


-- state register
state_register: process (CLK, RESET)
begin
 if (RESET = '1') then
  state <= IDLE;
 elsif (CLK'event and (CLK = '1')) then
  state <= next_state;
 end if;
end process;
-- next state function
ns_function: process (state, G, Z)
begin
```

VHDL for Alternative Binary Multiplier - 3

```
 if (state = IDLE) then
  if (G = '1') then
   next_state <= MUL;
  else
   next_state <= IDLE;
  end if;
 else -- state = MUL
  if (Z = '1') then
   next_state <= IDLE;
  else
   next_state <= MUL;
  end if;
 end if;
end process;
-- register transfers
reg_trans: process (CLK)
variable AQ: std_logic_vector(7 downto 0);
```

VHDL for Alternative Binary Multiplier - 4

```
begin
if (CLK'event and (CLK = '1')) then
 if (LOADB = '1') then
  B <= MULT_IN;
 elsif (LOADQ = '1') then
  Q <= MULT_IN;
 elsif ((state = IDLE) and (G = '1')) then
    P <= "11";
    A <= "0000";
 elsif (state = MUL) then
     P <= P - "01";
     if (Q(0) = '1') then
      AQ := ( ('0' & A) + ('0' & B)) & Q(3 downto 1);
-- The 0s preceding A and B permit the carry out of the
  addition to be captured instead of having an overflow.
```

VHDL for Alternative Binary Multiplier - 5

```
      else
       AQ := '0' & A & Q(3 downto 1); -- Zero fill is used
                 -- since this is an unsigned right shift.
      end if;
      A <= AQ(7 downto 4);
      Q <= AQ(3 downto 0);
     end if;
end if;
end process;
MULT_OUT <= A & Q;
end sequential;
```

Terms of Use

- Return to Title Page