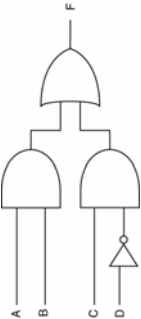
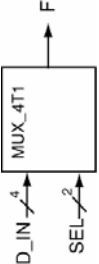
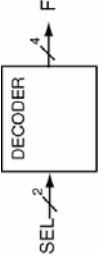
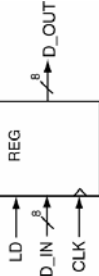
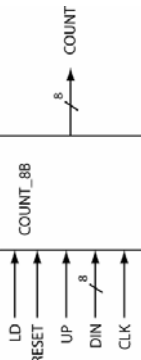


VHDL Cheat-Sheet

©Copyright: 2005 Bryan Mealy

Concurrent Statements		Sequential Statements
Concurrent Signal Assignment	↔	Signal Assignment
<code>target <= expression;</code>		<code>target <= expression;</code>
<code>A <= B AND C;</code> <code>DAT <= (D AND E) OR (F AND G);</code>		<code>A <= B AND C;</code> <code>DAT <= (D AND E) OR (F AND G);</code>
Conditional Signal Assignment	↔	<i>if</i> statements
<code>target <= expressn when condition else</code> <code> expressn when condition else</code> <code> expressn;</code>		<code>if (condition) then</code> { sequence of statements } <code>elsif (condition) then</code> { sequence of statements } <code>else --(the else is optional)</code> { sequence of statements } <code>end if;</code>
<code>F3 <= '1' when (L='0' AND M='0') else</code> <code> '1' when (L='1' AND M='1') else</code> <code> '0';</code>		<code>if (SEL = "111") then F_CTRL <= D(7);</code> <code>elsif (SEL = "110") then F_CTRL <= D(6);</code> <code>elsif (SEL = "101") then F_CTRL <= D(1);</code> <code>elsif (SEL = "000") then F_CTRL <= D(0);</code> <code>else F_CTRL <= '0';</code> <code>end if;</code>
Selective Signal Assignment	↔	<i>case</i> statements
<code>with chooser_expression select</code> <code>target <= expression when choices,</code> <code> expression when choices;</code>		<code>case expression is</code> <code>when choices =></code> {sequential statements} <code>when choices =></code> {sequential statements} <code>when others =></code> {sequential statements} <code>end case;</code>
<code>with SEL select</code> <code>MX_OUT <= D3 when "11",</code> <code>D2 when "10",</code> <code>D1 when "01",</code> <code>D0 when "00",</code> <code>'0' when others;</code>		<code>case ABC is</code> <code>when "100" => F_OUT <= '1';</code> <code>when "011" => F_OUT <= '1';</code> <code>when "111" => F_OUT <= '1';</code> <code>when others => F_OUT <= '0';</code> <code>end case;</code>
Process		
<code>label: process(sensitivity_list)</code> <code>begin</code> {sequential_statements} <code>end process label;</code>		
<code>procl: process(A,B,C)</code> <code>begin</code> <code>if (A = '1' and B = '0') then</code> <code>F_OUT <= '1';</code> <code>elsif (B = '1' and C = '1') then</code> <code>F_OUT <= '1';</code> <code>else</code> <code>F_OUT <= '0';</code> <code>end if;</code> <code>end process procl;</code>		

Description	CKT Diagram	VHDL Model
Typical logic circuit		<pre> entity my_ckt is Port (A,B,C,D : in std_logic; F : out std_logic); end my_ckt; architecture ckt1 of my_ckt is begin F <= (A AND B) OR (C AND (NOT D)); end ckt1; architecture ckt2 of my_ckt is begin F <= '1' when (A = '1' AND B = '1') else '1' when (C = '1' AND D = '0') else '0'; end ckt2; </pre>
4:1 Multiplexor		<pre> entity MUX_4T1 is Port (SEL : in std_logic_vector(1 downto 0); D_IN : in std_logic_vector(3 downto 0); F : out std_logic); end MUX_4T1; architecture my_mux of MUX_4T1 is begin F <= D_IN(0) when (SEL = "00") else D_IN(1) when (SEL = "01") else D_IN(2) when (SEL = "10") else D_IN(3) when (SEL = "11") else '0'; end my_mux; </pre>
2:4 Decoder		<pre> entity DECODER is Port (SEL : in std_logic_vector(1 downto 0); F : out std_logic_vector(3 downto 0)); end DECODER; architecture my_dec of DECODER is begin with SEL select F <= "0001" when "00", "0010" when "01", "0100" when "10", "1000" when "11", "0000" when others; end my_dec; </pre>
8-bit register with chip load enable		<pre> entity REG is port (LD,CLK : in std_logic; D_IN : in std_logic_vector (7 downto 0); D_OUT : out std_logic_vector (7 downto 0)); end REG; architecture my_reg of REG is begin process (CLK,LD) begin if (LD = '1' and rising_edge(CLK)) then D_OUT <= D_IN; end if; end process; end my_reg; </pre>
8-bit up/down counter with asynchronous reset		<pre> entity COUNT_8B is port (RESET,CLK,LD,UP : in std_logic; DIN : in std_logic_vector (7 downto 0); COUNT : out std_logic_vector (7 downto 0)); end COUNT_8B; architecture my_count of COUNT_8B is signal t_cnt : std_logic_vector(7 downto 0); begin process (CLK, RESET) begin if (RESET = '1') then t_cnt <= "00000000"; elsif (rising_edge(CLK)) then if (LD = '1') then t_cnt <= DIN; else if (UP = '1') then t_cnt <= t_cnt + 1; else t_cnt <= t_cnt - 1; end if; end if; end if; end process; COUNT <= t_cnt; end my_count; </pre>