Comparator Appendix

Half Adder Example

A simple 1-bit half adder circuit has two inputs and two outputs (Figure 1).

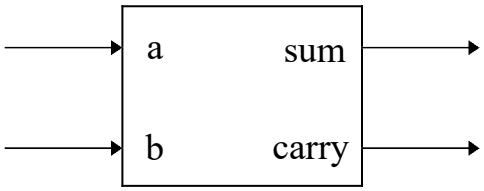

Figure 1: Half Adder Block Diagram

Table 1: Half Adder Truth Table

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The functionality of the half adder can be specified with the truth table (Table 1). It provides enough information to write out the Boolean equations (1) and (2) for the outputs.

$$sum = (\neg a) \cdot b + a \cdot (\neg b) \quad (1) \qquad carry = a \cdot b \quad (2)$$

VHDL code, which is based on the above specification, is provided in Listing 1. Code consists of two main parts: *entity* and *architecture*. *Entity* represents the interface of the circuit, while *architecture* describes either internal structure or behavior.

Listing 1: VHDL Description of a Half Adder

```
entity half_adder is
        port (a, b: in std_logic;
              sum, carry: out std_logic);
end half_adder;

architecture half_adder_arch of half_adder is
        begin
        sum <= (not a and b) or (a and not b);
        carry <= a and b;
end  half_adder_arch;
```

Entity represents the external view of the half adder, just like in the block diagram (Figure 1). Note, that inputs and outputs are of *std_logic* type, as it closely resembles the values a signal may have in the actual circuit. There is no need to declare this type, as the skeleton code generated by Vivado does so by default.

Architecture body consists of two concurrent signal assignment statements. These assignments correspond to Boolean equations (1) and (2), and are used to compute the output. However, **and**, **or** and **not** keywords in these statements are VHDL logic operators, and do not represent actual gates. Physical implementation may be synthesized using different logic gates or logic elements after undergoing optimization.

Alternatively, combinational logic can be described in a more abstract way using general conditional VHDL statements. For example, consider a simple 2-bit two-to-one multiplexer (Figure 2). I has two 2-bit data inputs *a* and *b*, control input *sel* and data output *o*. When *sel* is equal to logic 1, data input *a* is propagated to the data output *o*. Similarly, when *sel* is equal to logic 0, data input *b* is propagated to the data output *o*. Such behavior can be described without devising neither the truth table, nor Boolean equations.

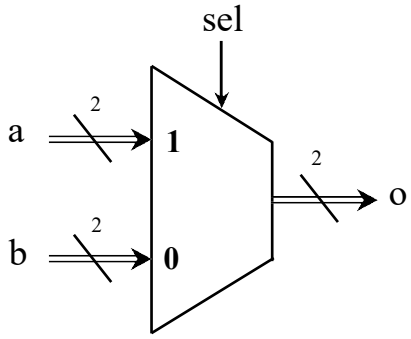

Figure 2: 2-bit 2-to-1 Multiplexer Block Diagram

VHDL code, which is based on the above specification, is provided in Listing 2. In order to describe a 2-bit two-to-one multiplexer in VHDL, "**when** .. **else**" conditional statement can be used. Data inputs/output *a*, *b* and *o* are now of *std_logic_vector* type (from 1 downto 0), since their size has been defined as being 2 bits wide. Note, that this description would generally invoke an actual multiplexer during synthesis.

Listing 2: VHDL Description of a 2-bit 2-to-1 Multiplexer

```
entity mux is
        port (a, b: in std_logic_vector(1 downto 0);
              sel: in std_logic;
              o: out std_logic_vector(1 downto 0));
end mux;

architecture mux_arch of mux is
begin
  o <= a when (sel = '1') else b;
end  mux_arch;
```