Electrical and Computer Engineering

Array Calculating Absolute Value of a Signed Vector

By Prawat Nagvajara

Synopsis

This note covers an array structure for calculating the absolute value of a signed n-bit vector.

Code

The code first constructs the processing element which adds the not x and c_in. Next , the two's complement array comprises the processing elements where the rightmost cell has a one as c_in , effectively, adding a one to the one's complement of x. Figure 1 shows a schematic of the array.

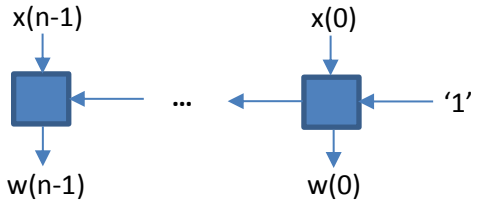

Fig. 1 Two's Complement Array

A multiplexer selects x or the two's complement of x to the output z based on the leftmost bit x(n-1) of the input x. This follows from the fact that a signed number is negative if and only if the leftmost bit is a one.

```
----------------------------------------------------
-- Company: Drexel ECE
-- Engineer: Prawat
-- Description: Absolute value circuit comprises
-- prcessing elements calculating thw two's
-- complement. The output is the two's complement
-- of the input if the input is negative.
----------------------------------------------------

-----------------------------
-- processing element
-- Adding two bits x and c_in
-----------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity pe is
port (x, c_in    : in    std_logic;
          z, c_out : out std_logic);
end pe;
architecture dataflow of pe is
begin
c_out <= (not x) and c_in;
z <= (not x) xor c_in;
end dataflow;
-----------------------------
-- Absolute Array
-- Array calculting the two's complement
-- A mux assigning the output
-- if x<0 then z <= -x; else z <= x; end if;
-----------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity abs_array is
generic (n: natural := 4);
port (x :  in  std_logic_vector(n-1 downto 0);
      z :  out std_logic_vector(n-1 downto 0));
end abs_array;

architecture Behavioral of abs_array is
component  pe is
port (x, c_in     : in  std_logic;
         z, c_out : out std_logic);
end component;
signal c,w: std_logic_vector(n-1 downto 0);
```

```
begin
-------------------
-- array calculating -x
-------------------
c(0) <= '1';
G1: for i in 0 to n-1 generate
G2: if i < n-1 generate
U: pe port map(x(i), c(i), w(i), c(i+1));
end generate G2;
G3: if i = n-1 generate
U: pe port map(x(i), c(i), w(i), open);
end generate G3;
end generate G1;
--------------------
-- mux
--------------------
process(x(n-1), w)
begin
 if x(n-1) = '1' then z <= w;
 else z <= x;
 end if;
end process;

end Behavioral;
```

Verification

Figure 2 below shows a simulation wave. The number of bits n is default to 4. When the input x is negative the output z is the two's complement of x. When x is equal to -2^{n-1} the output is incorrect because the correct result positive 2^{n-1} requires at least n+1 bits as for its representation. For instance, the two's complement of -4 ("1100") is 4 ("0100"), however, with the 3-bit representation -4 is "100" however positive 4 cannot be presented with a 3-bit vector.

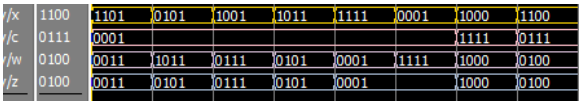

Fig. 2 Simulation Wave