VHDL for FPGA Design/4-Bit ALU

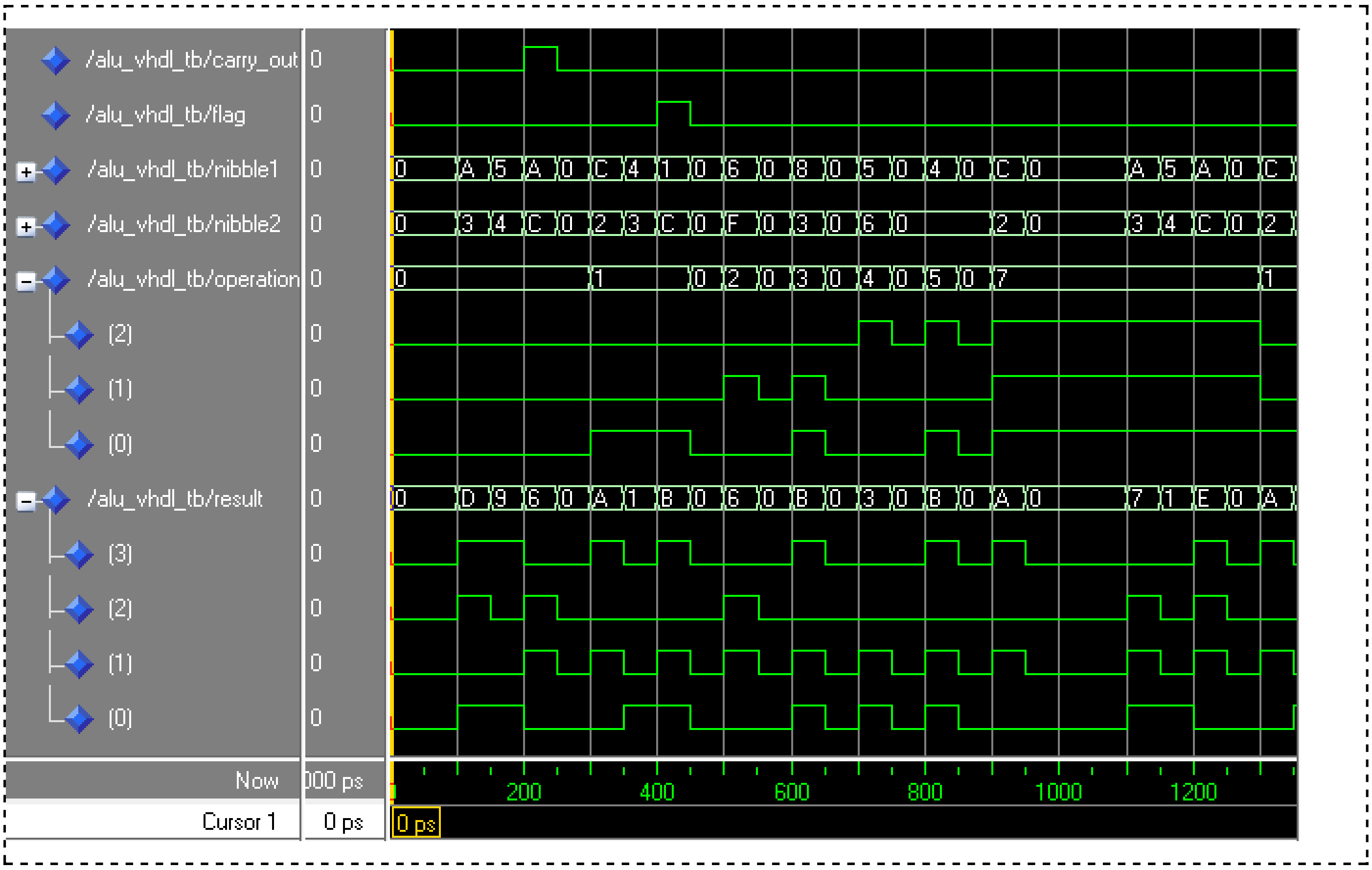
4-Bit ALU VHDL Code

A combinatorial ALU with the following operations:

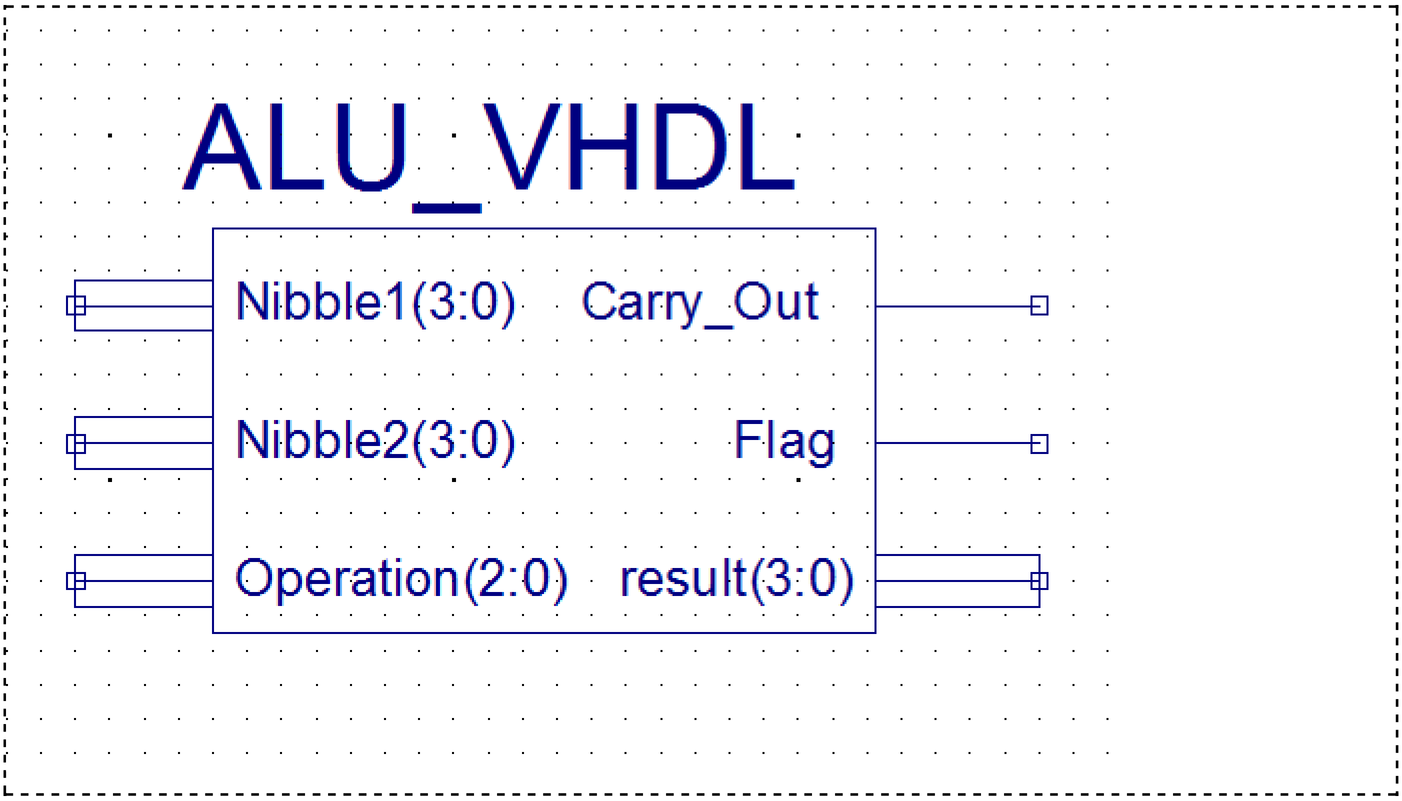
Operation	Result	Flag	Description
000	Nibble1 + Nibble2	Carry = Overflow	Addition
001	| Nibble1 - Nibble2 |	1 if Nibble2 > Nibble1, 0 otherwise	Test / diff
010	Nibble1 AND Nibble2	0	Bitwise AND
011	Nibble1 OR Nibble2	0	Bitwise OR
100	Nibble1 XOR Nibble2	0	Bitwise XOR
101	15 - Nibble1	0	Bitwise inverse of Nibble1
110	15 - Nibble2	0	Bitwise inverse of Nibble2
111	Nibble1 + Nibble2 + 1	Carry = Overflow	Addition

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ALU_VHDL is
   port
   (
      Nibble1, Nibble2 : in std_logic_vector(3 downto 0);
      Operation : in std_logic_vector(2 downto 0);

      Carry_Out : out std_logic;
      Flag : out std_logic;
      Result : out std_logic_vector(3 downto 0)
   );
end entity ALU_VHDL;

architecture Behavioral of ALU_VHDL is

   signal Temp: std_logic_vector(4 downto 0);

begin

   process(Nibble1, Nibble2, Operation, temp) is
   begin
      Flag <= '0';
      case Operation is
         when "000" => -- res = nib1 + nib2, flag = carry = overflow
            Temp   <= std_logic_vector((unsigned("0" & Nibble1) + unsig
            Result <= temp(3 downto 0);
            Flag   <= temp(4);
         when "001" => -- res = |nib1 - nib2|, flag = 1 iff nib2 > nib1
            if (Nibble1 >= Nibble2) then
               Result <= std_logic_vector(unsigned(Nibble1) - unsigned(
               Flag   <= '0';
            else
               Result <= std_logic_vector(unsigned(Nibble2) - unsigned(
               Flag   <= '1';
            end if;
         when "010" =>
            Result <= Nibble1 and Nibble2;
         when "011" =>
            Result <= Nibble1 or Nibble2;
         when "100" =>
            Result <= Nibble1 xor Nibble2;
         when "101" =>
            Result <= not Nibble1;
         when "110" =>
            Result <= not Nibble2;
         when others => -- res = nib1 + nib2 + 1, flag = 0
            Temp   <= std_logic_vector((unsigned("0" & Nibble1) + unsig
            Result <= temp(3 downto 0);
            Flag   <= temp(4);
      end case;
   end process;

end architecture Behavioral;
```

Simulation Waveform



Generated Symbol


ALU_VHDL
Nibble1(3:0)
Nibble2(3:0)
Operation(2:0)
Carry_Out
Flag
result(3:0)