M2 Master E3A Integration Circuits Systems

TD5-6: Realization of a data path and a state machine

1. **IFTAKHER Mohammed Akib – No. 22211204**

2. **KIBIWOTT Albert Kiplagat - No. 22211612**

Instructor: Professor Hervé MATHIAS

In this practical work we realized an operator that allows calculating the average power of a signal using the formula:

$$P = \frac{1}{N}\sum_{k=0}^{N-1} x_k^2$$

Where N is the number of samples used to make the calculation

The x_k signal arrives synchronously to the clock signal and the P signal must remain present until the following calculation. The calculation is started only if the control signal C is at 1. If it is at 0, the operator finishes his last calculation and then goes to standby mode.

1) **Data path diagram allowing the realization of this operator**

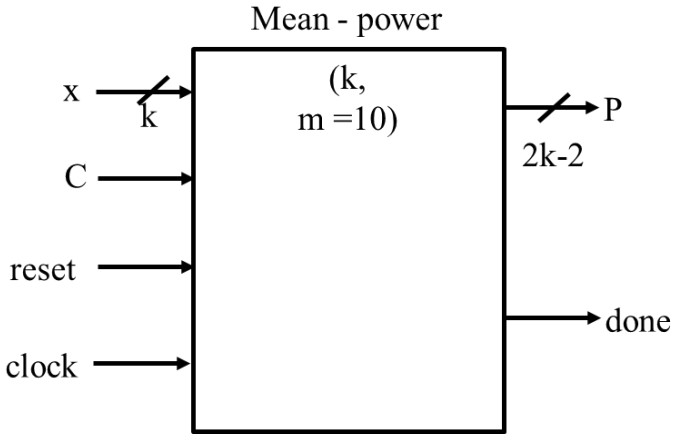

Figure 1: Top-level entity of the circuit.

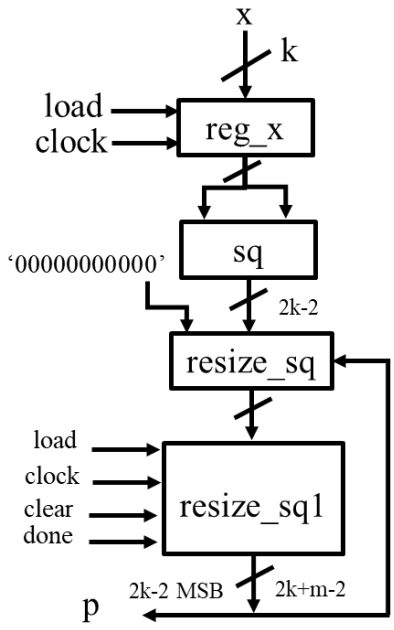

Figure 2: Data path diagram of the circuit.

2) **The signals used to control the data path**

- Load
- Clear
- Done

3) **A state machine diagram to control the previous signals**

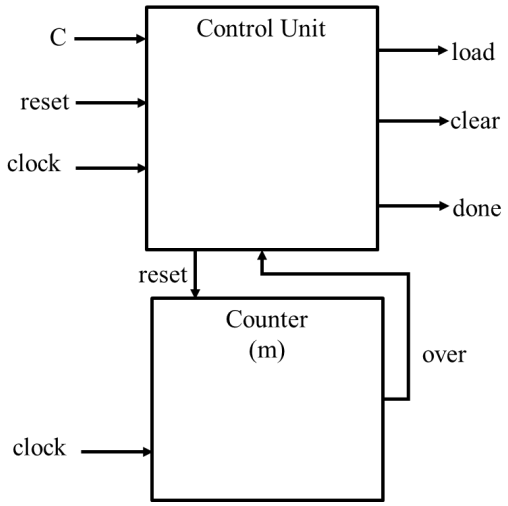

Figure 3: State machine diagram to control the previous signal.

Code for the Datapath and Control path

```
 LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.numeric_std.ALL;

ENTITY calcul_puissance IS
    GENERIC(
   m : integer :=10 ;
   k : integer := 16
    );

    PORT(
        reset      : IN std_logic;     -- reset asynchrone
        clock      : IN std_logic;     -- system clock
        C          : IN std_logic;     -- command = 1 when the operation must be done
        X          : IN signed(k-1 downto 0);   -- input signal
        P          : OUT unsigned(2*k-3 downto 0) -- average power of signal.
    );
END calcul_puissance;

architecture a1 of calcul_puissance is

type statetype is (idle, counts , fin); --the three states of state machine (control path)
signal state: statetype;
signal counter: unsigned(m-1 downto 0):=(others => '1'); -- counter to trace the operation

signal reg_x     : signed(k-1 downto 0); -- signal storing the content of X
signal sq        : unsigned(2*k-1 downto 0); -- signal storing the squared value of reg_x
signal resize_sq : unsigned (2*k-1+m downto 0):=(others=>'0'); -- signal used for concatenation of sq signal and '00000000'
signal resize_sq1 : unsigned (2*k-1+m downto 0):=(others=>'0'); -- signal used for accumulation of sum

-- Control signal
signal done : std_logic; -- signal indicates the completion of the counts
signal load : std_logic := '1'; -- signal signifying the operation is going on
signal clear : std_logic; -- signal clearing all the memory
signal over : std_logic; -- signal confirming the finished state


begin
   data_path: process(clock, reset, X, done, clear, load)
   begin
   if ( reset = '1') then
      P <= (others => '0');
      resize_sq <= (others => '0') ;
      load <= '0';

   elsif rising_edge(clock) then
         load <= '1';
         reg_x <= X;
         sq   <= unsigned(reg_x*reg_x);

        if done = '0' and clear = '0' then
         resize_sq <= "0000000000" & sq ;
         resize_sq1 <=  resize_sq + resize_sq1;

         else
         P <= resize_sq1(41 downto 12); -- the final output takes the MSBs
            if clear='1' then
              resize_sq1<= (others => '0');
              end if ;

   end if;

   end if;
   end process;
```

```
67      control_path: process(clock, reset, state, C)
68      begin
69
70         if ( reset = '1') then
71            state <= idle; -- idle and waiting state
72
73
74         elsif rising_edge(clock) then
75
76            case state is
77               when idle =>
78                  if C = '1' then
79                    clear <='1';
80                     state <= counts; ---C <= '1' hence control signal is activated and
   it shifts to counts state
81                  else
82                     over <= '0';
83                     clear <= '1';
84                     load <= '0';    -- acc reg cleared
85                     state <= idle;
86                     done <= '0'; --when idle even with the clock done signal is on
87                     counter <= (others => '1');  -- initializing counter with all '1's
88                  end if;
89               when counts =>
90                  clear <='0';
91                  load <= '1';
92                     if (counter = "0000000000")then
93                     counter <= (others => '1');
94                     done <= '1'; --done signal is activated (all inputs have been
   counted)
95                     state <= fin;
96                     else
97                     counter <= counter - "0000000001";
98                     done <= '0'; --done signal is still low until count is finished
99                     end if;
100
101              when fin =>       --counter has finished counting
102                 over <= '1';
103                 state <= idle;
104
105           end case;
106
107        end if;
108     end process;
109
110
111  end architecture a1 ;
112
```

Test Bench

Here, a sinusoidal input signal with an amplitude equal to the maximum admissible amplitude is used. The amplitude of the input signal is divided by 2 for every N sample. Below is the test-bench code and simulation.

Test Bench Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.numeric_std.ALL;
use IEEE.math_real.ALL;


ENTITY test_calcul_puissance IS

END test_calcul_puissance;


architecture test of test_calcul_puissance is

component calcul_puissance is
    generic(
   k : integer :=16;
   m : integer := 10
    );
port(
        reset     : IN std_logic;     -- reset asynchrone
       clock      : IN std_logic;     -- horloge systeme
       C          : IN std_logic;     -- commande = 1 quand l'opÃ©ration doit se faire
       X          : IN signed(k-1 downto 0);   -- facteur de division
       P          : OUT unsigned(2*k-3 downto 0)
);
end component;

signal       reset       : std_logic := '0';
signal        clock      : std_logic := '0';
signal        C          : std_logic := '1';
signal        X          : signed(15 downto 0) := (others=>'0');
signal        P          : unsigned(29 downto 0) := (others=>'0');
signal         N : integer := 2**10;

constant freq : real := 1.0e5;
signal amp : real := 32767.0 ; -- Maximum admissiable amplitude for 16 bit input
2x32767.

begin
      UTT : calcul_puissance
      port map(
       reset => reset,
       clock => clock,
        C    => C,
        X    => X,
        P    => P
      );


reset <= '0';

clk_process: process --to form the clock
begin
      clock <= '0';
      wait for 50 ns;

      clock <= '1';
      wait for 50 ns;

end process;


main_process: process(clock)
begin

   if rising_edge(clock) then

        if N = 0 then
           amp <= amp/2.0 ;  -- Amplitude is divided by 2 in every N samples.
           N<=2**10;


            else
               N <= N - 1;

            end if;

            X <= to_signed(integer(amp*sin(math_2_pi*freq*real((now/(1 ns))*1.0e-9))),x'
length); -- Input signal

      end if;
end process;

end test;
```

Test Bench Simulation

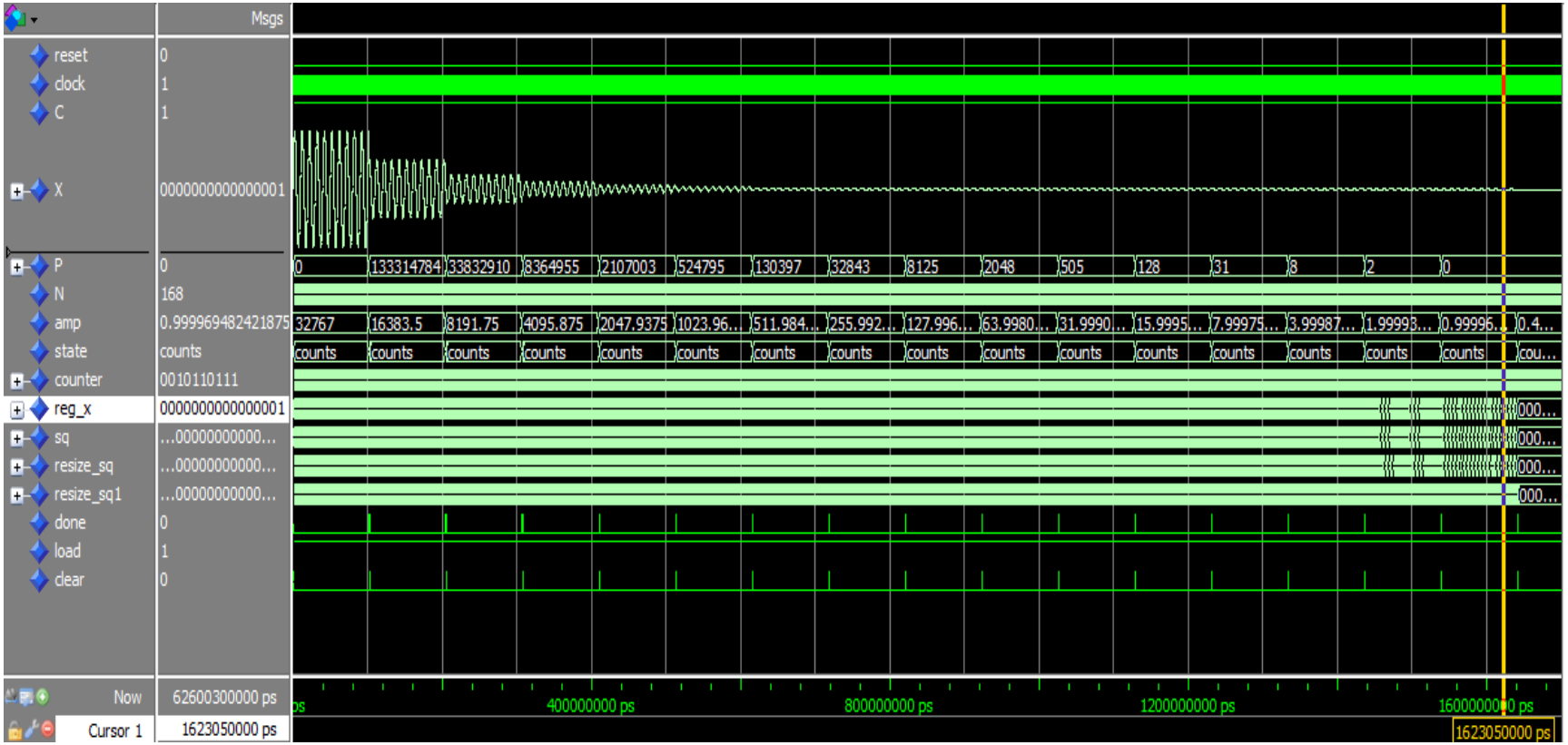

Figure 4: Simulation of the state machine.

The figure above shows the simulation of the whole circuit. As shown, the amplitude of the input signal X is divided by 2 for every N sample. As a result, P is reduced by 4 at each new calculation. For example, the last calculation $\frac{8}{4} = 2$.

These results verify that our operator is working successfully and the desired objectives of this practical work have been met.