ДОДАТОК А

Графічна частина кваліфікаційної роботи

Харківський національний університет радіоелектроніки

Кафедра АПОТ

Кваліфікаційна робота магістра

Подієві HDL-моделі керуючих автоматів в системах логічного управління

Магістранта групи СКСм-19-2
Малишенко Дмитра
Олександровича

Керівник:
доц. каф. АПОТ
Шкіль О.С

Харків 2021

Постановка задачі

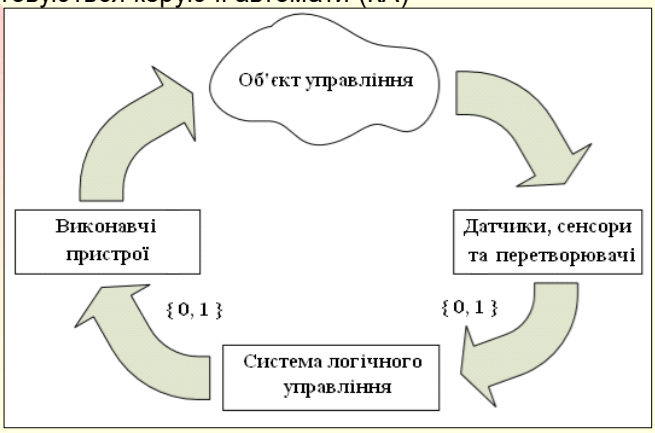
- При апаратній реалізації часових керуючих автоматів виникає проблема обробки зовнішніх вхідних сигналів і подій, які з'являються в довільні моменти часу.

Для вирішення завдання необхідно:

- ввести класифікацію моделей часових автоматів у відповідності зі способом обробки вхідних сигналів (подій) і способом формування вихідних сигналів;
- ввести класифікацію вхідних сигналів і подій та способів їх обробки;
- для кожного типу моделей розробити шаблон (pattern) синтезованої підмножини VHDL-моделі даного автомата в стилі автоматного програмування і підтвердити коректність запропонованих моделей шляхом їх моделювання, синтезу та імплементації в FPGA інструментальними засобами CAD XILINX ISE.

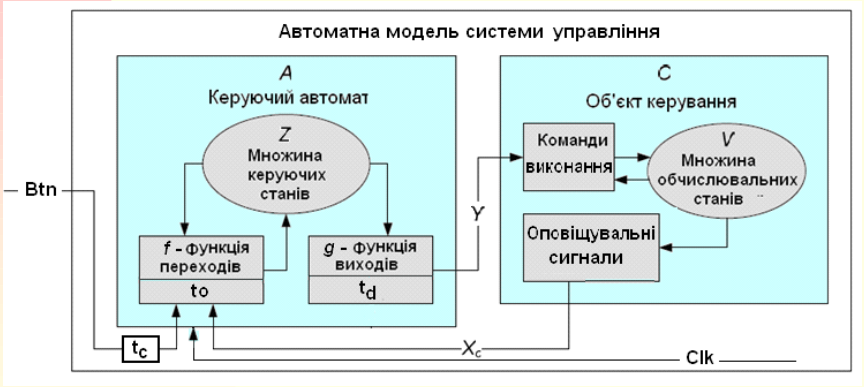
Системи логічного управління

- Одним з класів систем автоматичного управління є системи логічного управління (СЛУ). Специфіка СЛУ полягає в тому, що у них вхідні сигнали X і вихідні сигнали Y можуть приймати тільки два значення - 0 та 1. Для реалізації системи управління використовуються керуючі автомати (КА)



Автоматна модель системи управління

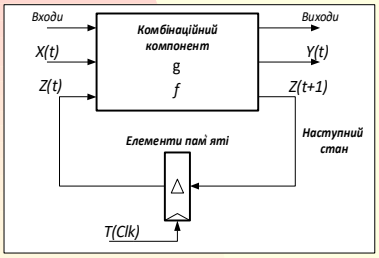
- Z – множина керуючих станів; Btn – множина зовнішніх подій;
- X_C – множина оповіщувальних сигналів; Clk – синхропослідовність



- t_c – часові (вхідні) обмеження; to - таймаути; t_d - вихідні затримки.

Часовий структурний автомат

- Для реалізації системи управління реального часу використовується модель часового автомата (timed FSM), яка дозволяє враховувати вплив метричного часу на переходи між технічними станами системи управління за рахунок введення узагальненої часової змінної T.
- Часовий автомат $W = (X, Y, Z, f, g, z_0, T)$



Функція виходів $Y(t) = g(X(t), Z(t))$

Функція переходів $Z(t+1) = f(X(t), Z(t))$

Новий стан $Z(t+1) \equiv Z(t)$

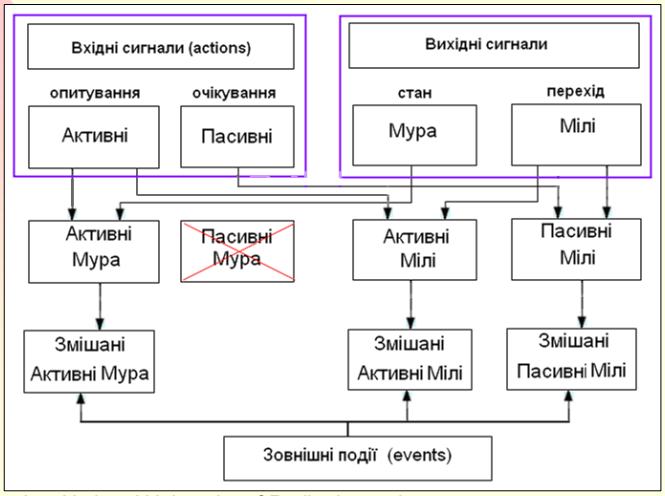
Часовий структурний автомат $Z(t+1) = f(X(t), Z(t), T)$

$Y(t) = g(X(t), Z(t), T)$

- Часова змінна T складається з трьох параметрів: часові обмеження t_c (timing constraints), вхідні таймаути t_{to} (timeouts) і вихідні затримки t_d (output delays), які іноді називаються вихідними таймаутами.

Класифікація моделей керуючих автоматів

- Вхідні сигнали – вхідні дії (input action) та зовнішні події (events).
- Вихідні сигнали – вхідні дії (output action) та діяльності (activities).



Класифікація та способи обробки подій

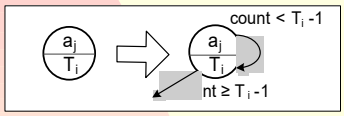
- Вхідні сигнали – вхідні дії (input action) та зовнішні події (events).
- Вихідні сигнали – вхідні дії (output action) та діяльності (activities).
- **Подія** - це абстрактне поняття, що припускає таку зміну умов зовнішнього середовища, яке породжує певну реакцію системи. Події можуть генеруватися як зовнішнім середовищем, так і всередині самої системи управління її компонентами.

Обробка подій, як правило, пов'язана з системами реального часу, які визначаються, виходячи з динамічних характеристик контрольованих процесів і пов'язаних з ним подій.

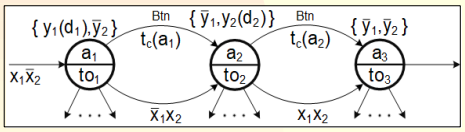
Реалізація часових параметрів та зовнішніх подій в часових автоматах Мура

- Графові моделі та часові діаграми часових автоматів **Мура**

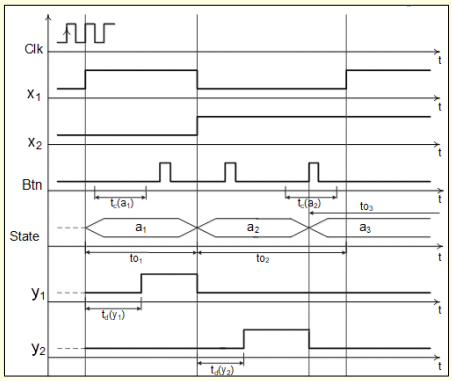
Реалізація затримок в графовій моделі автомата Мура



Реалізація часових параметрів в графовій моделі автомата Мура

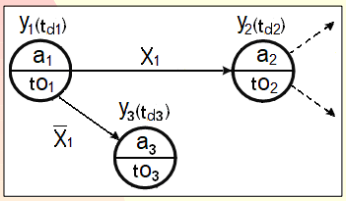


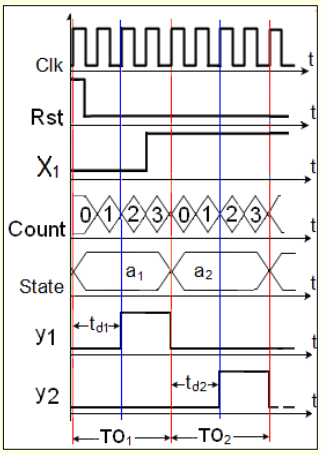
Waveform часового автомата Мура з урахуванням її зовнішніх подій Btn



Активний автомат Мура

- Активний часовий автомат функціонує в залежності від значення вхідного сигналу в певний момент часу.). Зміна вхідного безпосередньо не ініціює зміни стану автомата. Автомат опитує вхідні сигнали в моменти часу, які визначаються алгоритмом його роботи і, таким чином, реалізує функцію переходів.





Модель активного автомата Мура при to = 1 і t_d = 0 збігається з моделлю традиційного мікропрограмного автомата.

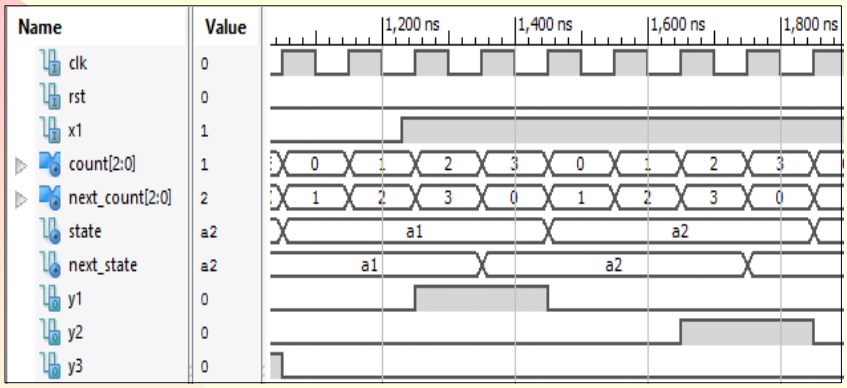
VHDL-модель активного автомату Мура

```
process (Clk, Rst)
  begin
    if Rst = '1' then   state <= a1;
                count <= (others => '0' );
    elsif rising_edge(Clk) then
                state <= next_state;   count <= next_count;
    end if;
end process;
process (state, count)
  begin
  next_count <= (others => '0');
  case state is
    when a1 =>
      if count < TO1 - 1 then
                next_state <=a1;
                next_count <= count + 1;
      elsif x1 = '1' then next_state <= a2;
      else  next_state <= a3;
      end if;
      . . .
  end case;
end process;
y1 <= '1'  when ( state = a1 and count >= td1 ) else '0';
y2 <= '1'  when ( state = a2 and count >= td2 ) else '0';
y3 <= '1'  when ( state = a3 and count >= td3 ) else '0';
```

Модель складається з двох процесів: комбінаційної частини, яка обчислює наступний стан та контролює лічильник, який реалізує часові параметри та послідовностної частини, яка на основі синхропослідовності реалізує перехід до наступного стану

Вихідні сигнали автомату Мура реалізуються за рахунок використання умовних операторів паралельного призначення сигналів

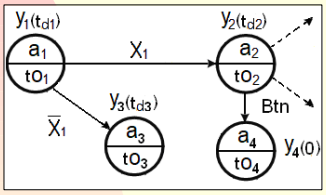
Waveform активного автомату Мура



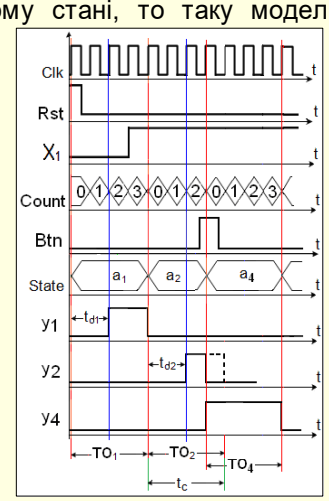
- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої моделі

Активний змішаний автомат Мура

- Якщо у активній моделі керуючого автомата Мура крім вхідних сигналів, що опитуються, використовуються зовнішні події Btn, які переривають знаходження у поточному стані, то таку модель прийнято називати змішаною.



Зовнішня подія Btn перериває таймаут знаходження автомату у поточному стані та ініціює перехід до нового стану.



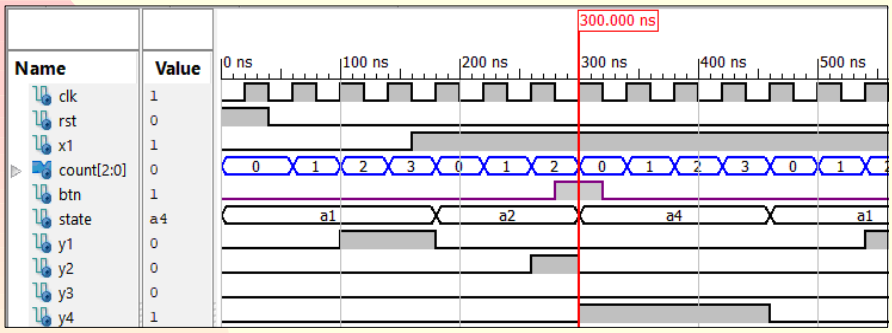
VHDL-модель змішаного автомату Мура

```
process (Clk, Rst)
  begin
   if Rst = '1' then   state <= a1;
            count <= ( others => '0' );
   elsif rising_edge(Clk) then
            state <= next_state;
            count <= next_count;
   end if;
end process;
process (state, count, Btn, x1)
  begin
  next_count <= (others => '0');
  case state is
   when a1 =>
    if count < TO1 - 1 then
            next_state <=a1;
            next_count <= count + 1;
    elsif x1 = '1' then next_state <= a2;
    else  next_state <= a3;
    end if;
```

```
_when a2 =>
    if Btn = '1' then next_state <= a4;
    elsif count < TO2 - 1 then
            next_state <= a2.;
            next_count <= count + 1;
    else  next_state <= . . .;
    end if;
  . . .
when others => next_state <= a1;
            next_count <= ( others => '0' );
  end case;
end process;

y1 <= '1'  when ( state = a1 and count >=
td1 ) else '0';
y2 <= '1'  when ( state = a2 and count >=
td2 ) else '0';
y3 <= '1'  when ( state = a3 and count >=
td3 ) else '0';
Y4 <= '1'  when ( state = a4 ) else '0';
```

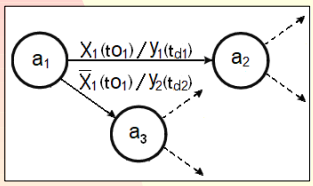
Waveform активного змішаного автомату Мура

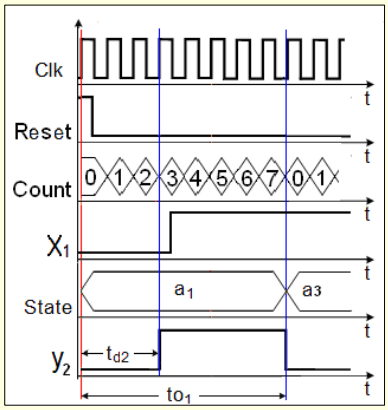


- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої модел

Активний автомат Мілі

- У часового автомата Мілі час появи вихідних сигналів визначаються часом появи вхідних сигналів (з урахуванням затримок). Час існування вихідних сигналів автомата Мілі визначається тривалістю переходу в новий стан і в момент входу в новий стан вихідні сигнали даного стану обнуляються.





Модель активного автомата Мілі досить близька до традиційного (мікропрограмному) автомату, а при to = 1 і t_d = 0 збігається з нею.

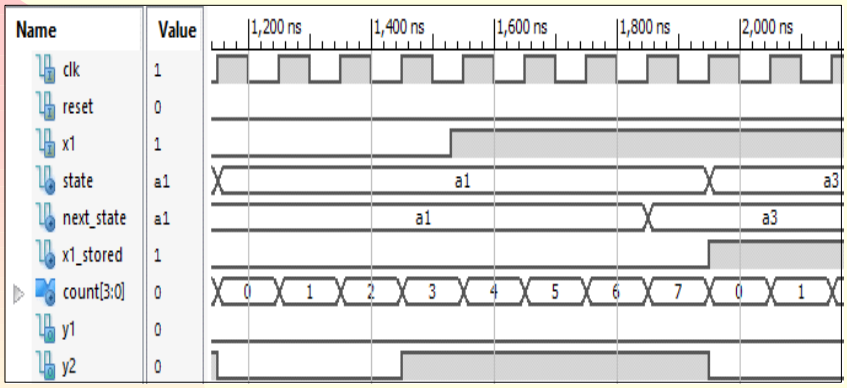
VHDL-модель активного автомату Мілі

```
-- register input signals and events
process (Clk, Reset)
   begin
     if Reset = '1' then   x1_stored <= '0';
     elsif rising_edge(Clk) then
              if state /= next_state then
                x1_stored <= x1;                          -- x1='0'
              end if;
     end if;
end process;
combinational logic for next_state and Y
process (state, x1_stored, count)
   begin
     y1 <= '0';    y2 <= '0';
     case State is
         when a1 =>
              if x1_stored = '1' then          --transition a1 –> a2
                if count < TO1 - 1 then next_state <= a1;
                else next_state <= a2;
                end if;
                if count >= Td1 then y1 <= '1';
                end if;
              else                              -- transition a1 –> a3
                if count < TO1 - 1 then next_state <= a1;
                else next_state <= a3;
                end if;
                if count >= Td2 then y2 <= '1';
                end if;
              end if;
. . .
     end case;
end process;
```

Модель складається з двох процесів: комбінаційної частини, яка обчислює наступний стан та контролює лічильник, який реалізує часові параметри та послідовностної частини, яка на основі синхропослідовності реалізує перехід до наступного стану

Вихідні сигнали автомату Мілі формуються в комбінаційному процесі

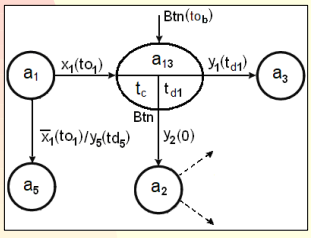
Waveform активного автомату Мілі



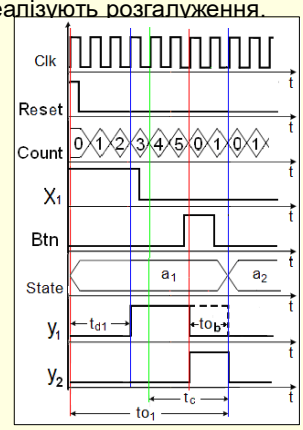
- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої моделі

Змішаний активний автомат Мілі

- Графова модель змішаного автомата Мілі має одну особливість Обробка переривають подій передбачає, що для переходу e_j має бути заданий перехід e_m, який ініціюється як реакція на переривючу подію. Але традиційна графовая модель не передбачає наявності дуги, відповідної переходу e_m, так як дуги графовой моделі не реалізують розгалуження.



Модифікована графова модель змішаного часового автомата Мілі

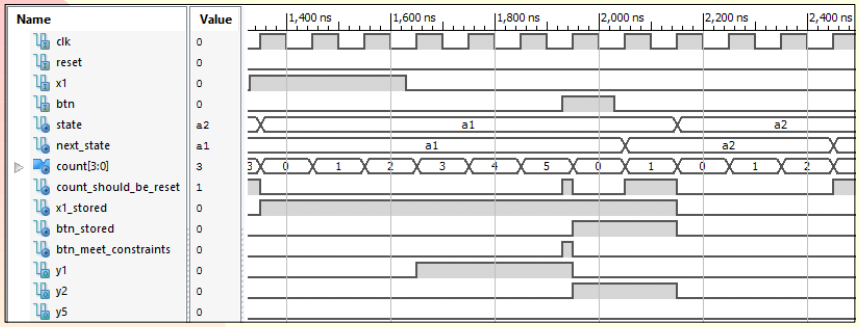


VHDL-модель змішаного активного автомату Мілі

```
¶
--·register·input·signals·and·events¶
process·(Clk,·Reset)¶
···begin·¶
·····if·Reset·=·'1'·then¶
    →    ····x1_stored·<=·'0';¶
    →    ····Btn_stored·<=·'0';¶
·····elsif·rising_edge(Clk)·then¶
    →    if·state·/=·next_state·then¶
    →    ····x1_stored·<=·x1;¶
    →    ····Btn_stored·<=·'0';¶
    →    elsif·Btn_meet_constraints·=·'1'·then¶
    →    ····Btn_stored·<=·'1';¶
    →    end·if;¶
·····end·if;¶
end·process;¶
¶
--·combination·logic·for·Btn_meet_constraints¶
process·(·state,·x1_stored,·Btn,·count·)¶
···begin¶
    →    Btn_meet_constraints·<=·'0';¶
    →    if·Btn·=·'1'·then¶
    →    ···if·state·=·a1·and·x1_stored·=·'1'·and·¶
    →    ·····count·>=·TC0·-·1·and·count·<·TC1·¶
    →    ····then·Btn_meet_constraints·<=·'1';¶
    →    ····end·if;¶
    →    end·if;¶
end·process;¶
¶
```

```
combinational·logic·for·next_state·and·Y¶
process·(·state,·x1_stored,·Btn_stored,·count·)¶
···begin¶
···y1·<=·'0';···y2·<=·'0';····y5·<=·'0';¶
·····case·State·is¶
·····when·a1·=>¶
    →    if·x1_stored·=·'1'·then¶
    →    ····if·Btn_stored·=·'1'·then¶
·····if·count·<·TOBtn·-·1·then···next_state·<=·a1;¶
·····else·next_state·<=·a2;¶
·····end·if;¶
    →    ·····y2·<=·'1';¶
    →    ····else¶
    →    ·····if·count·<·TO1·-·1·then·next_state·<=·a1;¶
    →    ·····else····next_state·<=·a3;¶
    →    ·····end·if;¶
    →    ·····if·count·>=·Td1·then·y1·<=·'1';¶
    →    ····end·if;¶
    →    end·if;¶
    →    else¶
    →    ····if·count·<·TO1·-·1·then·next_state·<=·a1;¶
    →    ····else····next_state·<=·a5;¶
    →    ····end·if;¶
    →    ····if·count·>=·Td5·then·y5·<=·'1';¶
    →    ····end·if;¶
    →    end·if;¶
···¶
·····end·case;¶
end·process;¶
```

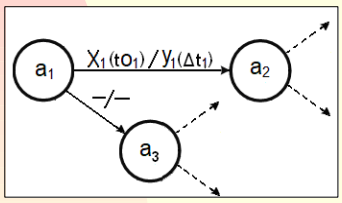
Waveform змішаного активного автомату Мілі

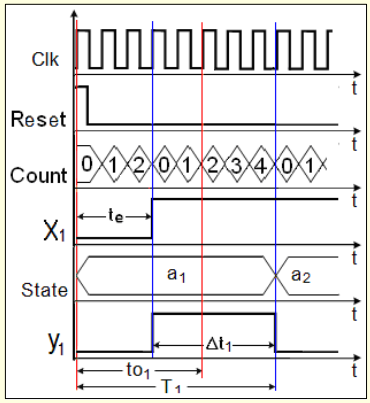


- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої моделі

Пасивний автомат Мілі

- Пасивний автомат Мілі в поточному стані очікує появи вхідного сигналу протягом проміжку часу, який визначається таймаутом to_i для даного i-го стану, і після приходу вхідного сигналу ініціює перехід новий стан з видачею вихідного сигналу тривалістю Δt. При цьому приймається, що $t_d = 0$. Якщо за період часу очікування вхідний сигнал не виникає, то автомат безумовно переходить у новий стан без видачі вихідного сигналу





Пасивний автомат Мілі досить близький до традиційного (мікропрограмного) автомату, а при to = 1 і $t_d = 1$ збігається з ним.

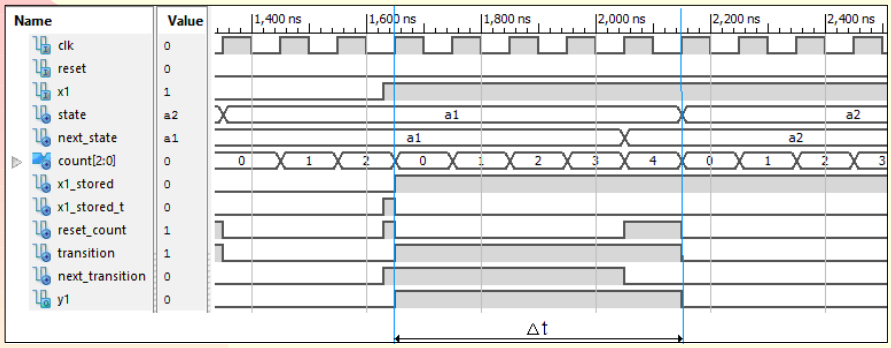
VHDL-модель пасивного автомату Мілі

- .

```
-- comb. logic for next_state, Y, next_transition

process ( state, x1_stored, x1, transition, count )
  begin
   y1 <= '0';
   x1_stored_t <= '0';
   next_transition <= '0';
     case State is
         when a1  =>
           if transition = '1' then
             if x1_stored = '1' then
                 if count < DLT1 - 1  then
                   next_state <= a1;
                   next_transition <= '1';
                 else     next_state <= a2;
                 end if;
                 if count < DLT1 then y1 <= '1';
                 end if;
             else next_state <= a3;
             end if;
           else                    -- timeout
             next_state <= a1;
             if x1_stored = '0' and x1 = '1' then
                 next_transition <= '1';
                 x1_stored_t <= '1';
           elsif count >= TO1 - 1 then
                 next_transition <= '1';
                 x1_stored_t <= '0';
           end if;
         end if;
       end case;
end process;
```

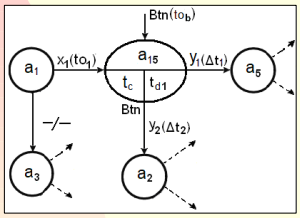
Waveform пасивного автомату Мілі

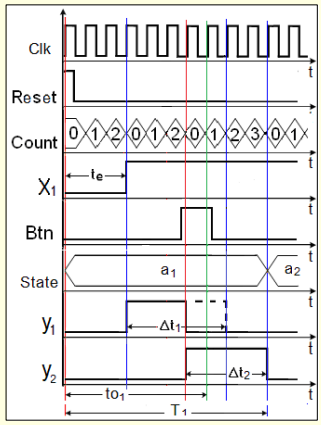


- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої моделі

Змішаний пасивний автомат Мілі

- Якщо в моделі пасивного керуючого автомата Мілі, окрім вхідних сигналів, що що очікуються протягом таймауту, використовуються перериваючі зовнішні події, то таку модель прийнято називати змішаною.



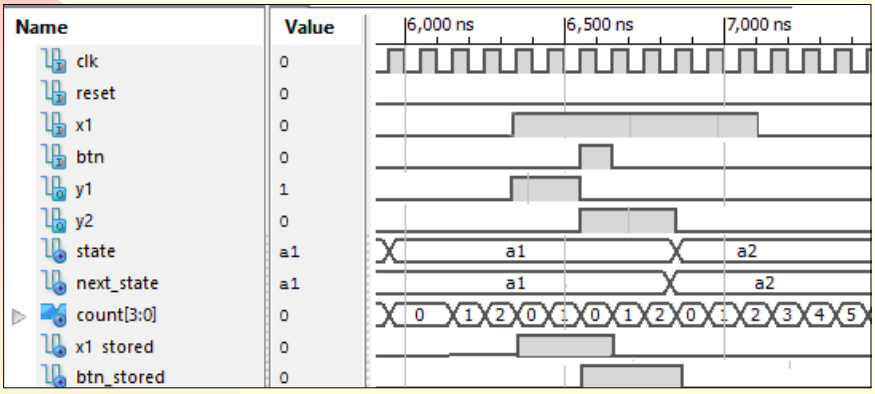


Для пасивного часового автомата Мілі з перериваючою подією також пропонується використовувати модифікацію графової моделі у формі дводольного графу з вершинами двох типів.

VHDL-модель змішаного пасивного автомату Мілі

```
process ( state, x1_stored, Btn_stored, x1, transition, count )
 begin
  y1 <= '0';
  x1_stored_t <= '0';
  next_transition <= '0';
    case State is
    when a1  =>
        if transition = '1' then
          if x1_stored = '1' then
           if Btn_stored = '1' then
             if count < output_delay_Y2 - 1 then
               next_state <= a1; next_transition <= '1';
             else next_state <= a2;
             end if;
             if count < output_delay_Y2 then  Y2 <= '1';
             end if;
          else  next_state <= a3;    --відповідає переходу без події
          end if;
        else
          next_state <= a1;
          if x1_stored = '0' and x1 = '1' then
                next_transition <= '1';    x1_stored_t <= '1';
          elsif count >= T1 - 1 then
                next_transition <= '1';    x1_stored_t <= '0';
          end if;
        end if;
```


Waveform змішаного пасивного автомату Мілі



- Порівняння отриманих результатів моделювання після синтезу та імплементації показує повне співпадіння зі специфікацією, що підтверджує працездатність запропонованої моделі

Результати

- Синтез і імплементація отриманих фрагментів VHDL-моделей на FPGA XC3S500E-5fg320 підтвердили їх коректність і працездатність. Результати часового моделювання після імплементації в цілому співпали з результатами поведінкового моделювання

Тип автомата	Частота моделю-вания, MHz	Розра-хункове число тригерів	Flip-Flops	Latches	BELS	Slices/ LUTs
Мура активний	320,513	5	5	0	11	6/11
Мура активний з перериваючою подією	256.430	5	5	0	21	9/18
Мілі активний	287,650	6	6	0	15	7/13
Мілі активний з перериваючою подією	165,307	9	9	0	30	6/30
Мілі пасивний	232,626	7	7	0	15	8/15
Мілі пасивний з перериваючою подією	141,898	11	11	0	42	9/45

ВИСНОВКИ

- У роботі запропонована класифікація керуючих автоматів за способом отримання вихідних сигналів на моделі Мура і Мілі, за способом обробки вхідних сигналів на активні і пасивні моделі і класифікація подій за способом їх обробки на ті, що ініціюють, та ті, що переривають. Дана класифікація дозволила побудувати VHDL-шаблони моделей часових керуючих автоматів для вирішення різноманітних завдань в системах логічного управління.
- Моделювання, синтез і імплементація в FPGA підтвердили приналежність розроблених шаблонів до підмножини VHDL, що синтезується, та дотримання часових параметрів, заданих специфікаціями.
- Реалізація запропонованих VHDL-моделей виконана на FPGA XC3S500E-5fg320. Результати діагностичного експерименту підтвердили коректність і працездатність запропонованих моделей.

ДОДАТОК Б

Результати синтезу та імплементації VHDL-моделі змішаного часового автомата Мура

Повний протокол моделювання, синтезу та імплементації змішаного автомата Мура

Лістинг Б.1 – VHDL-модель змішаного автомата Мура

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;


entity Moor_Btn is
port (Clk, Rst, x1, Btn: in std_logic;
            y1, y2, y3, y4 : out std_logic);
end Moor_Btn;

architecture Behavioral of Moor_Btn is

   constant count_length : integer := 3;

constant TO1 : std_logic_vector ( count_length - 1 downto 0 ) := "100";
      -- 4
constant TO2 : std_logic_vector ( count_length - 1 downto 0 ) := "100";
      -- 4
constant TO3 : std_logic_vector ( count_length - 1 downto 0 ) := "110";
      -- 6
constant TO4 : std_logic_vector ( count_length - 1 downto 0 ) := "100";
      -- 4


constant td1: std_logic_vector ( count_length - 1 downto 0 ) := "010";
-- 2
constant td2: std_logic_vector ( count_length - 1 downto 0 ) := "010";
-- 2
constant td3: std_logic_vector ( count_length - 1 downto 0 ) := "011";
-- 3
constant td4: std_logic_vector ( count_length - 1 downto 0 ) := "000";
-- 0

type state_type is ( a1, a2, a3, a4);
signal state, next_state : state_type;
signal count, next_count : std_logic_vector (count_length-1 downto 0 );


begin
-- state and timer synchronization
process (Clk, Rst)
   begin
     if Rst = '1' then   state <= a1;
      count <= ( others => '0' );
     elsif rising_edge(Clk) then
      state <= next_state;
```

```
            count <= next_count;
           end if;
      end process;
             -- combinational logic for transition
      process (state, count, Btn, x1) -- process (state, count, Btn) was here
- error;
         begin
         next_count <= (others => '0');
         case state is
           when a1 =>
             if count < TO1 - 1 then
            next_state <=a1;
            next_count <= count + 1;
             elsif x1 = '1' then next_state <= a2;
             else  next_state <= a3;
             end if;
          when a2 =>
            if Btn = '1' then next_state <= a4;
            elsif count < TO2 - 1 then
            next_state <= a2;
            next_count <= count + 1;
            else  next_state  <=  a3;     -- the transition a2-a3 was added with condition
not(Btn)
             end if;
           when a3 => if count < TO3 - 1 then
            next_state <=a3;
            next_count <= count + 1;
            else next_state <= a1;
             end if;
              when a4 => if count < TO4 - 1 then
            next_state <=a4;
            next_count <= count + 1;
            else next_state <= a1;
             end if;

            when others =>
                             next_state <= a1;
                             next_count <= ( others => '0' );

       end case;
      end process;
      y1 <= '1'  when ( state = a1 and count >= td1 ) else '0';
      y2 <= '1'  when ( state = a2 and count >= td2 ) else '0';
      y3 <= '1'  when ( state = a3 and count >= td3 ) else '0';
      Y4 <= '1'  when ( state = a4 ) else '0';


      end Behavioral;
```

Лістинг Б.2 – Верифікація моделі

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY test_Moor_Btn IS
END test_Moor_Btn;

ARCHITECTURE behavior OF test_Moor_Btn IS
```

```
    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Moor_Btn
    PORT(
         Clk : IN  std_logic;
         Rst : IN  std_logic;
         x1 : IN  std_logic;
         Btn : IN  std_logic;
         y1 : OUT  std_logic;
         y2 : OUT  std_logic;
         y3 : OUT  std_logic;
         y4 : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal Clk : std_logic := '0';
   signal Rst : std_logic := '1';
   signal x1 : std_logic := '0';
   signal Btn : std_logic := '0';

      --Outputs
   signal y1 : std_logic;
   signal y2 : std_logic;
   signal y3 : std_logic;
   signal y4 : std_logic;

   -- Clock period definitions
   constant Clk_period : time := 40 ns;

BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: Moor_Btn PORT MAP (
          Clk => Clk,
          Rst => Rst,
          x1 => x1,
          Btn => Btn,
          y1 => y1,
          y2 => y2,
          y3 => y3,
          y4 => y4
        );

   -- Clock process definitions
   Clk_process :process
   begin
            Clk <= '0';
            wait for Clk_period/2;
            Clk <= '1';
            wait for Clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- insert stimulus here

Rst <='1', '0' after Clk_period;
x1 <='0', '1' after Clk_period*4;
Btn <='0', '1' after Clk_period*7, '0' after Clk_period*8;
```

```
        wait;
      end process;

  END;
```

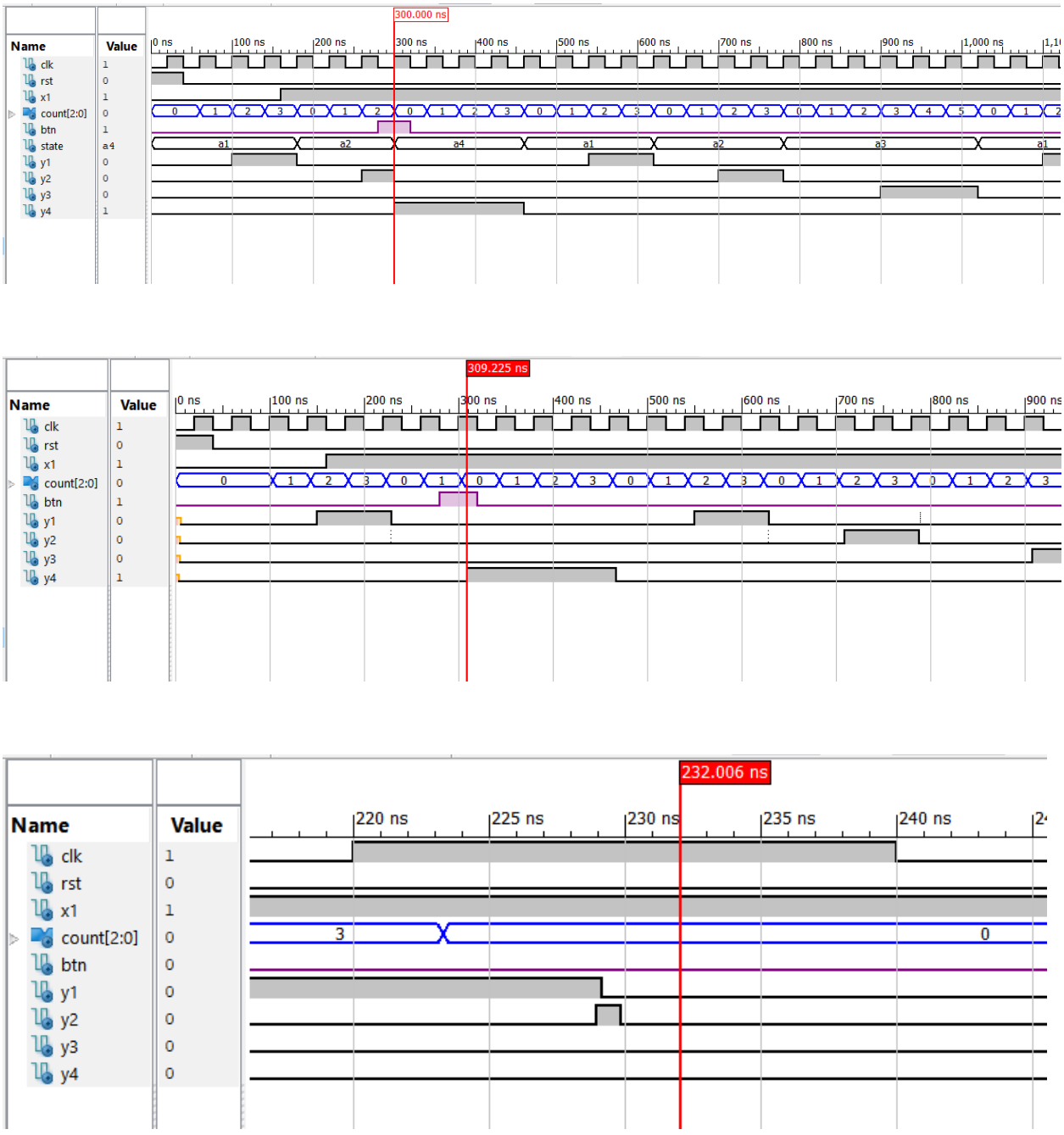


Рисунок Б.1 – Результати моделювання

Лістинг Б.3 – Протокол синтезу

```
Minimum period: 3.900ns (Maximum Frequency: 256.430MHz)
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <state/FSM> on signal <state[1:2]> with user encoding.
-------------------
 State | Encoding
-------------------
 a1    | 00
```

```
 a2     | 01
 a3     | 10
 a4     | 11
-------------------
```

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	5	9,312	1%	
Number of 4 input LUTs	18	9,312	1%	
Number of occupied Slices	9	4,656	1%	
Number of Slices containing only related logic	9	9	100%	
Number of Slices containing unrelated logic	0	9	0%	
Total Number of 4 input LUTs	18	9,312	1%	
Number of bonded IOBs	8	232	3%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	4.00			

Synthesis Report (Отчет по синтезу)

```
Release 14.7 - xst P.20131013 (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
--> Parameter TMPDIR set to xst/projnav.tmp


Total REAL time to Xst completion: 1.00 secs
Total CPU time to Xst completion: 0.49 secs

--> Parameter xsthdpdir set to xst


Total REAL time to Xst completion: 1.00 secs
Total CPU time to Xst completion: 0.49 secs

--> Reading design: Moor_Btn.prj

TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) Design Hierarchy Analysis
  4) HDL Analysis
  5) HDL Synthesis
     5.1) HDL Synthesis Report
  6) Advanced HDL Synthesis
     6.1) Advanced HDL Synthesis Report
  7) Low Level Synthesis
  8) Partition Report
  9) Final Report
      9.1) Device utilization summary
      9.2) Partition Resource Summary
      9.3) TIMING REPORT

=========================================================================
*                      Synthesis Options Summary                        *
```

```
=========================================================================
---- Source Parameters
Input File Name                    : "Moor_Btn.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "Moor_Btn"
Output Format                      : NGC
Target Device                      : xc3s500e-5-fg320

---- Source Options
Top Module Name                    : Moor_Btn
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : LUT
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : Yes
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : Yes
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Multiplier Style                   : Auto
Automatic Register Balancing       : No

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 500
Add Generic Clock Buffer(BUFG)     : 24
Register Duplication               : YES
Slice Packing                      : YES
Optimize Instantiated Primitives   : NO
Use Clock Enable                   : Yes
Use Synchronous Set                : Yes
Use Synchronous Reset              : Yes
Pack IO Registers into IOBs        : Auto
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Keep Hierarchy                     : No
Netlist Hierarchy                  : As_Optimized
RTL Output                         : Yes
Global Optimization                : AllClockNets
Read Cores                         : YES
Write Timing Constraints           : NO
Cross Clock Analysis               : NO
Hierarchy Separator                : /
Bus Delimiter                      : <>
Case Specifier                     : Maintain
Slice Utilization Ratio            : 100
BRAM Utilization Ratio             : 100
Verilog 2001                       : YES
Auto BRAM Packing                  : NO
Slice Utilization Ratio Delta      : 5
```

```
=========================================================================
*                          HDL Compilation                              *
=========================================================================
Compiling vhdl file
"D:/MyDesigns/Moor_c_Btn_for_artical_RIA_Mealy/Moor_Btn.vhd" in Library work.
Entity <moor_btn> compiled.
Entity <moor_btn> (Architecture <behavioral>) compiled.

=========================================================================
*                     Design Hierarchy Analysis                         *
=========================================================================
Analyzing hierarchy for entity <Moor_Btn> in library <work> (architecture
<behavioral>).


=========================================================================
*                           HDL Analysis                                *
=========================================================================
Analyzing Entity <Moor_Btn> in library <work> (Architecture <behavioral>).
Entity <Moor_Btn> analyzed. Unit <Moor_Btn> generated.


=========================================================================
*                           HDL Synthesis                               *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <Moor_Btn>.
    Related source file is
"D:/MyDesigns/Moor_c_Btn_for_artical_RIA_Mealy/Moor_Btn.vhd".
    Found finite state machine <FSM_0> for signal <state>.
    -----------------------------------------------------------------------
    | States             | 4                                              |
    | Transitions        | 10                                             |
    | Inputs             | 4                                              |
    | Outputs            | 4                                              |
    | Clock              | Clk                       (rising_edge)        |
    | Reset              | Rst                       (positive)           |
    | Reset type         | asynchronous                                   |
    | Reset State        | a1                                             |
    | Power Up State     | a1                                             |
    | Encoding           | automatic                                      |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 3-bit register for signal <count>.
    Found 3-bit adder for signal <next_count$add0000> created at line 77.
    Found 3-bit comparator less for signal <state$cmp_lt0000> created at line
67.
    Found 4-bit comparator less for signal <state$cmp_lt0001> created at line
80.
    Found 3-bit comparator greatequal for signal <y2$cmp_ge0000> created at
line 98.
    Found 3-bit comparator greatequal for signal <y3$cmp_ge0000> created at
line 99.
    Summary:
      inferred   1 Finite State Machine(s).
      inferred   3 D-type flip-flop(s).
      inferred   1 Adder/Subtractor(s).
      inferred   4 Comparator(s).
Unit <Moor_Btn> synthesized.
```

```
=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 1
 3-bit adder                                           : 1
# Registers                                            : 1
 3-bit register                                        : 1
# Comparators                                          : 4
 3-bit comparator greatequal                           : 2
 3-bit comparator less                                 : 1
 4-bit comparator less                                 : 1

=========================================================================

=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================

Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <state/FSM> on signal <state[1:2]> with user encoding.
-------------------
 State | Encoding
-------------------
 a1    | 00
 a2    | 01
 a3    | 10
 a4    | 11
-------------------

=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# FSMs                                                 : 1
# Adders/Subtractors                                   : 1
 3-bit adder                                           : 1
# Registers                                            : 3
 Flip-Flops                                            : 3
# Comparators                                          : 4
 3-bit comparator greatequal                           : 2
 3-bit comparator less                                 : 1
 4-bit comparator less                                 : 1

=========================================================================

=========================================================================
*                         Low Level Synthesis                           *
=========================================================================

Optimizing unit <Moor_Btn> ...

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block Moor_Btn, actual ratio is
0.

Final Macro Processing ...

=========================================================================
Final Register Report

Macro Statistics
```

```
# Registers                                          : 5
 Flip-Flops                                         : 5

=========================================================================

=========================================================================
*                            Partition Report                           *
=========================================================================

Partition Implementation Status
-------------------------------

  No Partitions were found in this design.

-------------------------------

=========================================================================
*                             Final Report                              *
=========================================================================
Final Results
RTL Top Level Output File Name     : Moor_Btn.ngr
Top Level Output File Name         : Moor_Btn
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 8

Cell Usage :
# BELS                             : 21
#      GND                         : 1
#      LUT2                        : 2
#      LUT3                        : 2
#      LUT3_L                      : 2
#      LUT4                        : 10
#      LUT4_L                      : 1
#      MUXF5                       : 3
# FlipFlops/Latches                : 5
#      FDC                         : 5
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 7
#      IBUF                        : 3
#      OBUF                        : 4
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s500efg320-5

 Number of Slices:                        9  out of   4656     0%
 Number of Slice Flip Flops:              5  out of   9312     0%
 Number of 4 input LUTs:                 17  out of   9312     0%
 Number of IOs:                           8
 Number of bonded IOBs:                   8  out of    232     3%
 Number of GCLKs:                         1  out of     24     4%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.
```

```
--------------------------


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
Clk                                | BUFGP                  | 5     |
-----------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
-----------------------------------+------------------------+-------+
Control Signal                     | Buffer(FF name)        | Load  |
-----------------------------------+------------------------+-------+
Rst                                | IBUF                   | 5     |
-----------------------------------+------------------------+-------+

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 3.900ns (Maximum Frequency: 256.430MHz)
   Minimum input arrival time before clock: 4.324ns
   Maximum output required time after clock: 5.832ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'Clk'
  Clock period: 3.900ns (frequency: 256.430MHz)
  Total number of paths / destination ports: 36 / 5
-------------------------------------------------------------------------
Delay:               3.900ns (Levels of Logic = 3)
  Source:            state_FSM_FFd1 (FF)
  Destination:       state_FSM_FFd2 (FF)
  Source Clock:      Clk rising
  Destination Clock: Clk rising

  Data Path: state_FSM_FFd1 to state_FSM_FFd2
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q             10   0.514   0.819  state_FSM_FFd1 (state_FSM_FFd1)
     LUT2:I1->O            1   0.612   0.360  state_FSM_FFd2-In_SW1_SW0 (N20)
     LUT4_L:I3->LO         1   0.612   0.103  state_FSM_FFd2-In_SW1 (N18)
     LUT4:I3->O            1   0.612   0.000  state_FSM_FFd2-In
(state_FSM_FFd2-In)
     FDC:D                     0.268          state_FSM_FFd2
    ----------------------------------------
    Total                      3.900ns (2.618ns logic, 1.282ns route)
                                       (67.1% logic, 32.9% route)
```

```
================================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'Clk'
  Total number of paths / destination ports: 6 / 4
--------------------------------------------------------------------------------
Offset:              4.324ns (Levels of Logic = 4)
  Source:            Btn (PAD)
  Destination:       state_FSM_FFd2 (FF)
  Destination Clock: Clk rising

  Data Path: Btn to state_FSM_FFd2
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             4   1.106   0.651  Btn_IBUF (Btn_IBUF)
     LUT2:I0->O            1   0.612   0.360  state_FSM_FFd2-In_SW1_SW0 (N20)
     LUT4_L:I3->LO         1   0.612   0.103  state_FSM_FFd2-In_SW1 (N18)
     LUT4:I3->O            1   0.612   0.000  state_FSM_FFd2-In
(state_FSM_FFd2-In)
     FDC:D                     0.268          state_FSM_FFd2
    ----------------------------------------
    Total                      4.324ns (3.210ns logic, 1.114ns route)
                                       (74.2% logic, 25.8% route)

================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'Clk'
  Total number of paths / destination ports: 15 / 4
--------------------------------------------------------------------------------
Offset:              5.832ns (Levels of Logic = 3)
  Source:            state_FSM_FFd1 (FF)
  Destination:       y3 (PAD)
  Source Clock:      Clk rising

  Data Path: state_FSM_FFd1 to y3
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q             10   0.514   0.902  state_FSM_FFd1 (state_FSM_FFd1)
     LUT4:I0->O            1   0.612   0.000  y3_and000011 (y3_and00001)
     MUXF5:I0->O           1   0.278   0.357  y3_and00001_f5 (y3_OBUF)
     OBUF:I->O                 3.169          y3_OBUF (y3)
    ----------------------------------------
    Total                      5.832ns (4.573ns logic, 1.259ns route)
                                       (78.4% logic, 21.6% route)

================================================================================

Total REAL time to Xst completion: 6.00 secs
Total CPU time to Xst completion: 5.90 secs

-->

Total memory usage is 4521488 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```