ЦИФРОВА СХЕМОТЕХНИКА

ЛЕКЦИЯ #14

Системи с външна памет

Проектиране на контролер за управление на SRAM памети: блокова схема, управляващи сигнали, тайминг изисквания. VHDL код

Необходимост от използване:

- съхранение (хардуерно) на големи масиви информация ИЗВЪН FPGA чипа;
- невъзможност софтуерният регистров тип памет да поддържа работно големи обеми данни (ограничен обем).

❖Налага се включване / изграждане на интерфейсен **контролер** - посредник между Базовата Изчислителна Система - **БИС** и външната памет **SRAM**:

❑ асинхронно действие на SRAM спрямо БИС – изисква специални управляващи сигнали за извършване на операциите четене/запис;

❑ SRAM изисква повече от 1 периода /цикъла на тактовия сигнал за извършване на операциите.

Системи с външна памет. Проектиране на контролер за управление на SRAM памети: блокова схема, управляващи сигнали, тайминг изисквания. VHDL код

Избор на памет (асинхронна) с примерна организация:

- SRAM (асинхронна);
- обем 256K (18 адресни шини);
- дължина на думата – 16 bit (два отделно достъпни байта).

Примерно съответствие – IS61LV25616AL *Integrated Silicon Solutions Inc*. (ISSI).

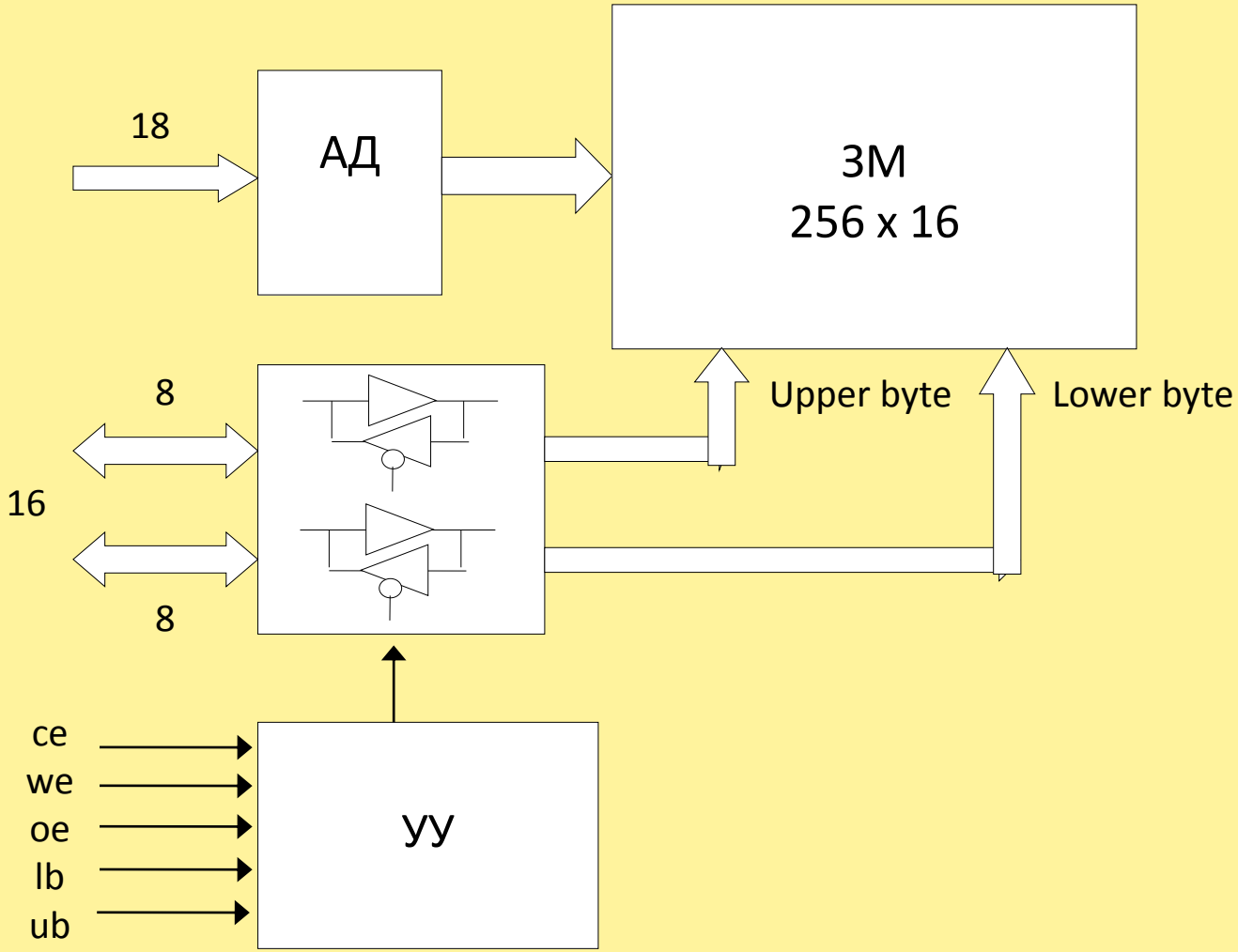
- Управляващи сигнали към паметта:

  - **ce** (chip enable) → разрешава/забранява чипа;
  - **we** (write enable) → разрешава запис в паметта;
  - **oe** (output enable) → разрешава изходите на паметта при четене;
  - **lb** (lower byte) → разрешава/забранява младшият байт по ШД;
  - **ub** (upper byte) → разрешава/забранява старшия байт по ШД.

Активно ниво на всички контролни сигнали – ниско!

Тук предполагаме R/W на 16-битови данни (lb=ub=0 едновременно).

Системи с външна памет. Контролер за управление на паметта
Блокова схема на SRAM



Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

Режими на работа на SRAM паметта

Операция	ce	we	oe	lb	ub	dio (upper)	dio (lower)
Неактивна	1	X	X	X	X	Z	Z
	0	1	1	X	X	Z	Z
	0	X	X	1	1	Z	Z
Четене	0	1	0	0	1	Data out	Z
	0	1	0	1	0	Z	Data out
	0	1	0	0	0	Data out	Data out
Запис	0	0	X	0	1	Data in	Z
	0	0	X	1	0	Z	Data in
	0	0	X	0	0	Data in	Data in

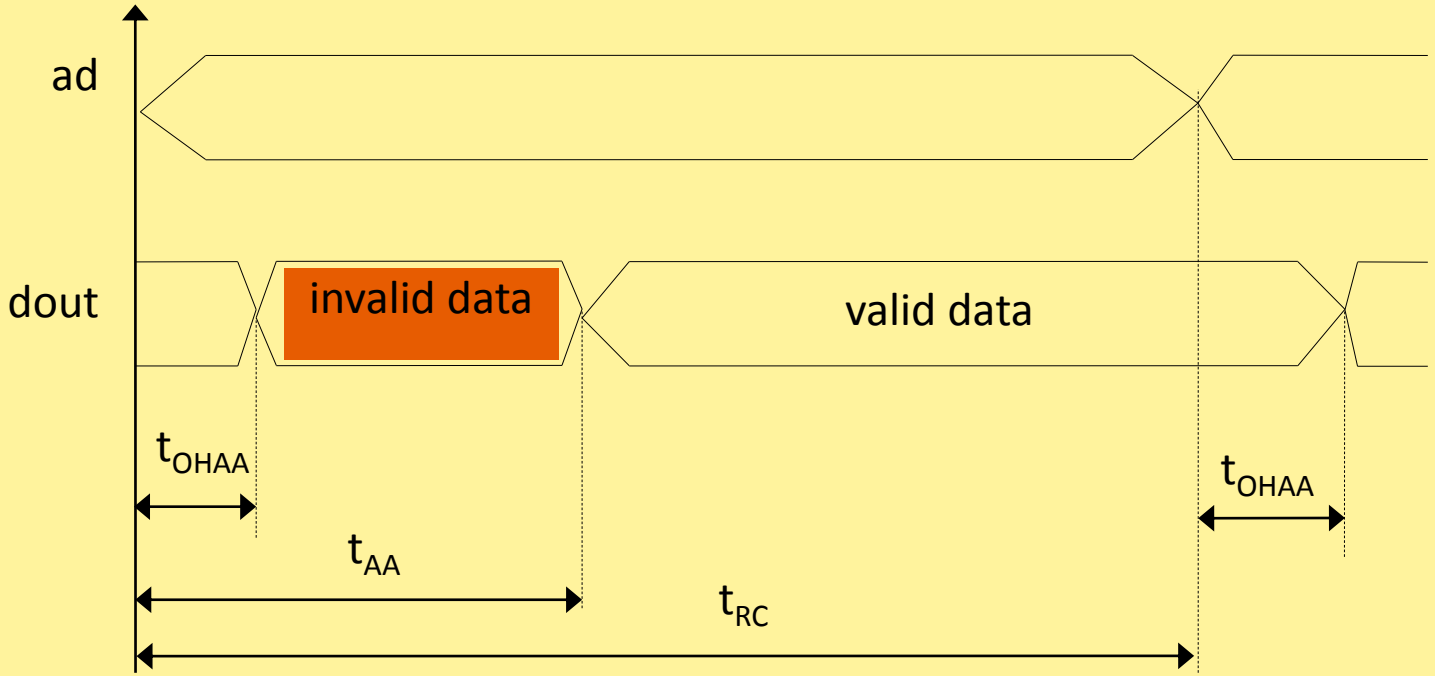
Операция	ce	we	oe	dio
Неактивна	1	X	X	Z
Четене	0	1	0	Valid data
Запис	0	0	X	Valid data

lb=ub=0 (едновременно разрешени, 16-битов обмен)
dio – 16 bit дума.

Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

Времедиаграми на четене/запис на SRAM от избрания тип

- **четене с отчитане само валидността на адреса (address controlled)**

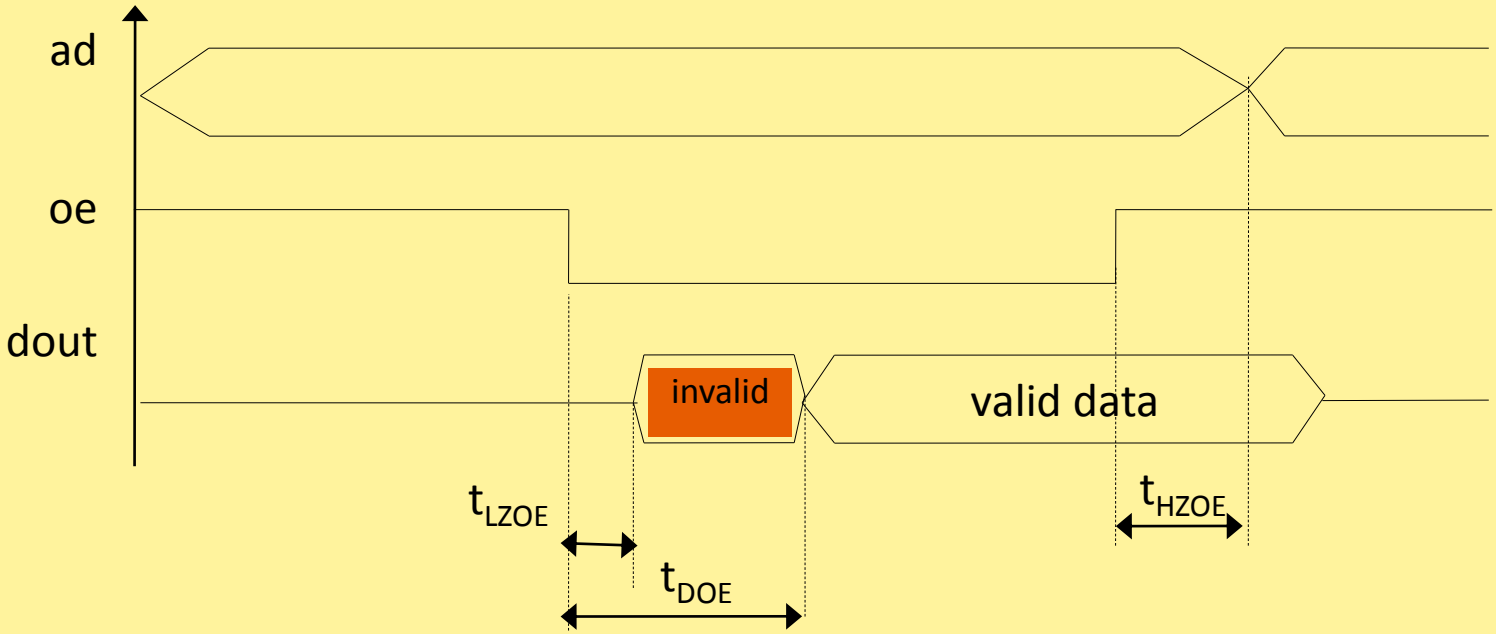


we=1, oe=0 – четене

Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

Времедиаграми на четене/запис на SRAM от избрания тип

- **четене с отчитане нивото на oe (т.нар. “oe controlled” режим)**



we=1 – четене с управление през oe

Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

- **Времеви параметри (тайминг) - ЧЕТЕНЕ**

t_{RC} – мин.време м/у две операции четене;

t_{AA} – мин. време за получаване на данни след подаване на адреса;

t_{OHA} – изходно hold време (данни валидни след подаване на адреса);

t_{DOE} – време за получаване стабилни данни след подаване на OE=L;

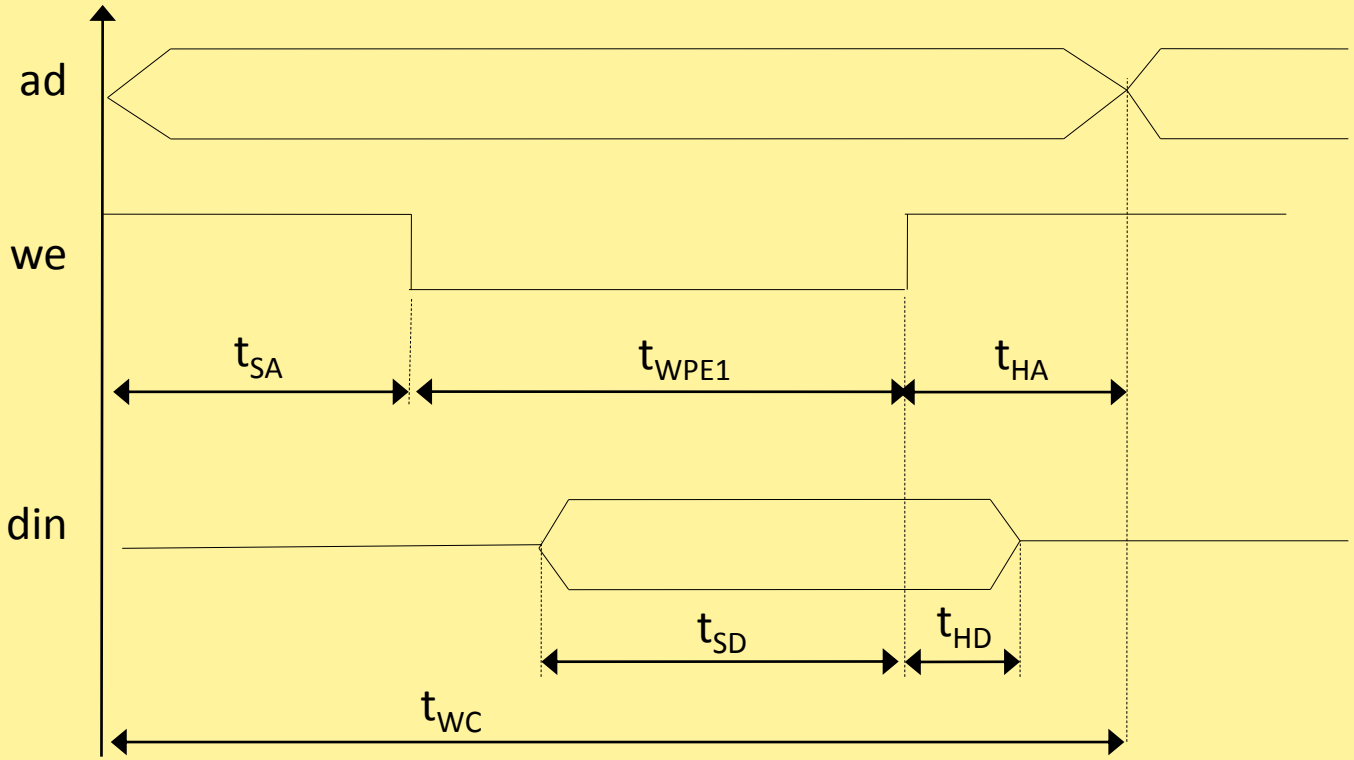
t_{LZOE} – време за излизане на изх.буфер от ВИС след подаване на OE=L;

t_{HZOE} – време за влизане на изх.буфер във ВИС след подаване на OE=H.

Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

Времедиаграми на четене/запис на SRAM от избрания тип

- **запис в паметта**



we=0 – запис

Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

- **Времеви параметри (тайминг) – ЗАПИС**

t_{WC} – мин.време м/у две операции запис;

t_{SA} – мин.време за задържане на адреса преди подаване на WE;

t_{HA} – мин.време за задържане на адреса след подаване на WE;

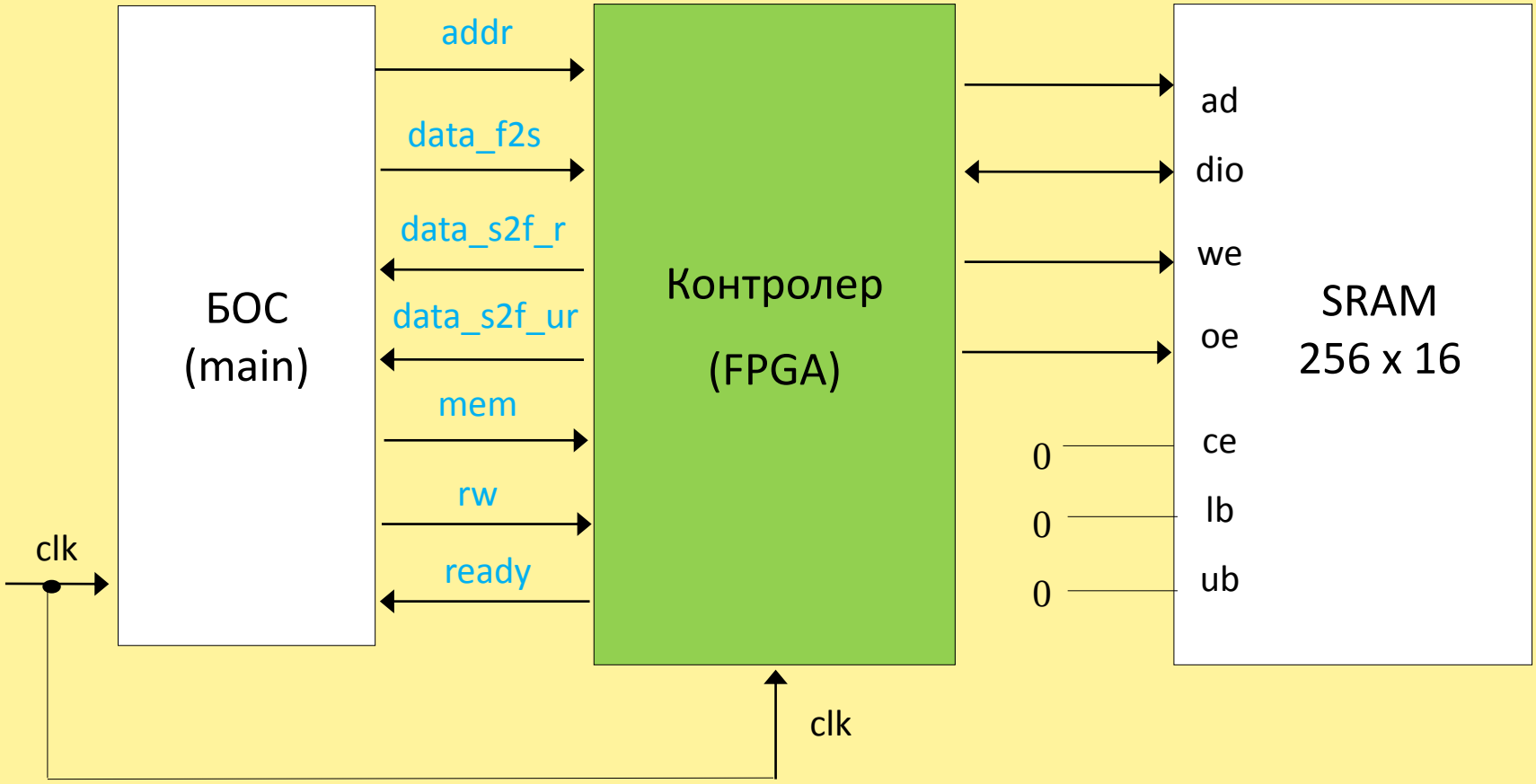
t_{PE1} – мин.продължителност на WE;

t_{SD} – мин.време за задържане на данните преди WE;

t_{HD} – мин.време за задъжане на данните след WE.

Системи с външна памет. Контролер за управление на паметта

Обмен на данните между БОС, контролера и SRAM - сигнали



Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

- **Управляващи сигнали мвжду БОС (main system) и FPGA (f)**

addr – 18-битов адрес;

data_f2s – разрешава запис на 16-битови данни в SRAM;

data_s2f_r – разрешава четене на 16-битови данни от SRAM (през буферен регистър - restricted);

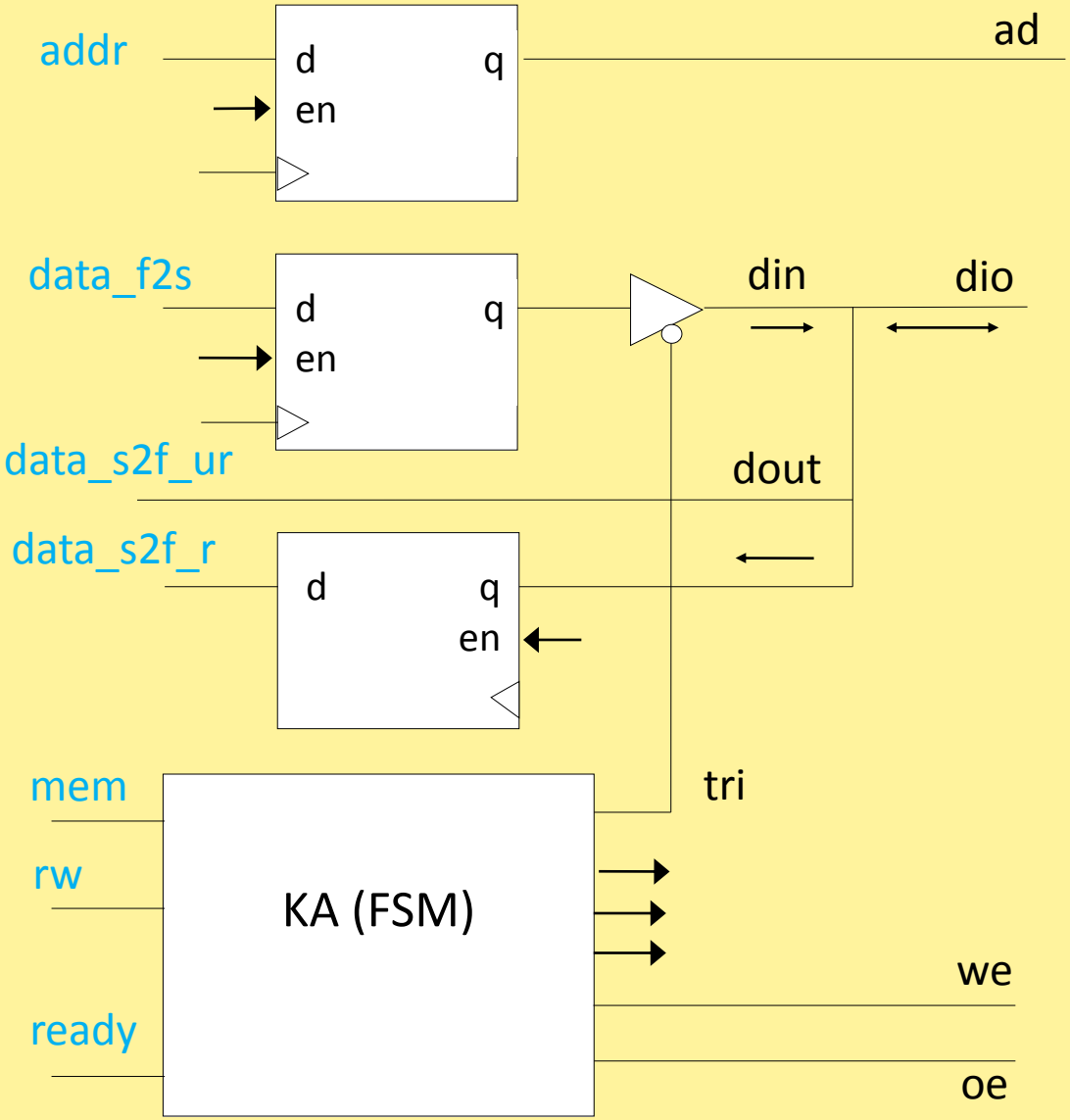
data_s2f_ur – разрешава четене на 16-битови данни от SRAM (директно, без буферен регистър - unrestricted);

mem – за извършване на операциите mem='1';

rw – определя типа операция rw='1' – четене, rw='0' - запис;

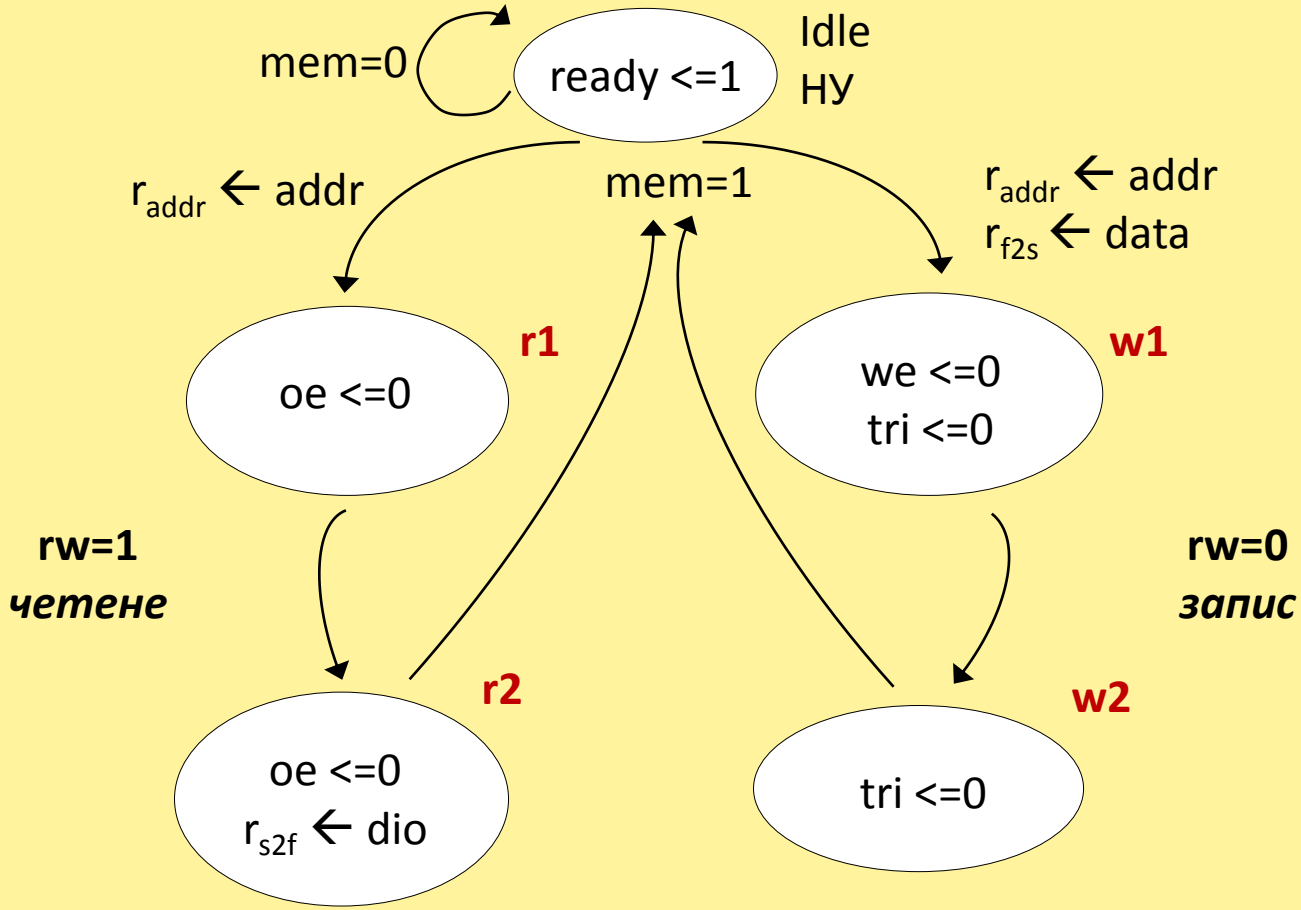
ready – готовност за приемане на нова команда от контролера.

Блокова схема на контролера



Системи с външна памет. Контролер за управление на паметта: сигнали, времедиаграми, параметри (тайминг)

Граф на КА на контролера за извършване операциите четене/запис в SRAM



Системи с външна памет. Сигнали, времедиаграми за извършване на операциите, параметри (тайминг)

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
library ieee;
use ieee.std.logic.1164.all;
entity SRAM_CTRL  is
      port (
            clk, reset: in std_logic;  -- от страната на БОС (system side)
            mem: in std_logic;
            rw: in std_logic;
            addr: in std_logic_vector(17 downto 0);
            data_f2s: in std_logic_vector(15 downto 0);
            ready: out std_logic;
            data_s2f_r, data_s2f_ur: out std_logic_vector(15 downto 0);
            ad: out std_logic_vector(17 downto 0);  -- от страната на FPGA (chip side)
            we, oe: out std_logic;
            dio: inout std_logic_vector(15 downto 0);  -- присъщи на SRAM
            ce, ub, lb: out std_logic
end SRAM_CTRL;
```

Системи с външна памет. Сигнали, времедиаграми за извършване на операциите, параметри (тайминг)

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
architecture  arch  of  SRAM_CTRL  is
        type state_type is (idle,rd1,rd2,wr1,wr2);   -- дефинира изброен тип за ВС на КА
        signal state_reg, state_next: state_type;     -- текущо/следващо ВС
        signal data_f2s_reg, data f2s_next: std_logic_vector(15 downto 0);  --данни от БОС
        signal data_s2f_reg, data_s2f_next: std_logic_vector(15 downto 0); -- данни от SRAM
        signal addr_reg, addr_next: std_logic_vector(17 downto 0); -- адреси текущ/следващ
        signal we_buf, oe_buf, tri_buf: std_logic;
        signal we_reg, oe_reg, tri_reg: std_logic;

process  (clk, reset)   -- регистри state & data
begin
   if (reset='1') then  -- в състояние idle /НУ/
        state_reg <= idle;
        addr_reg  <= (others=>'0');
        data f2s_reg <= (others=>'0');  -- нулиране данни
        data s2f_reg <= (others=>'0');  -- нулиране данни
        tri_reg <= '1';   -- неактивно ниво
        we_reg <= '1';  -- неактивно ниво
        oe_reg <= '1';   -- неактивно ниво
```

Системи с външна памет. Сигнали, времедиаграми за извършване на операциите, параметри (тайминг)

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
elsif (clk'event and clk='1') then   -- присвояване на регистрите при преход към следващо ВС
            state_reg <= state_next;
            addr_reg  <= addr_next;
            data f2s_reg <= data f2s_next;
            data s2f_reg <= data s2f_next;
            tri_reg <= tri_buf;
            we_reg <= we_buf;
            oe_reg <= oe_buf;
 end if;
end process;

process  (state_reg, mem, rw, dio_a, addr,
          data_f2s, data_f2s_reg, data_s2f_reg, addr_reg)
begin       -- обхождане на вътрешните състояния
   addr_next  <= addr_reg;
   data_f2s_next  <= data_f2s_reg;
   data_s2f_next  <= data_s2f_reg;
   ready <= '0';
```

Системи с външна памет. Сигнали, времедиаграми за извършване на операциите

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
case state_reg is
          when idle =>
           if mem='0' then
               state_next <= idle;   -- при НУ
          else
              addr_next <= addr;
              if rw='0'  then                    -- запис съгласно графа
                 state_next <= wr1;
                 data_f2s_next <= data_f2s;
              else                               -- четене съгласно графа
                 state_next <= rd1;
              end if;
          end if;
            ready <= '1';
            when  wr1  =>  state_next <= wr2;
            when  wr2  =>  state_next <= idle;
            when  rd1  =>  state_next <= rd2;
            when  rd2  =>  state_next <= idle;
   end case;
end process;
```

Системи с външна памет. Сигнали, времедиаграми за извършване на операциите

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
process  (state_next)    -- описание на изходните състояния
begin
    tri_buf <= '1';    -- в idle /НУ/
    we_buf <= '1';
    oe_buf <= '1';
case state_next  is      -- извън idle
            when idle =>
            when wr1 => tri_buf <= '0';
                        we_buf <= '0';
            when wr2 => tri_buf <= '0';
            when rd1 => oe_buf <= '0';
            when rd2 => oe_buf <= '0';
  end case;
end process;
```

Системи с външна памет. Сигнали, времедиаграми за извършване на операциите

Изходен код на VHDL с описание действието на контролера за работа със SRAM

```
  data_s2f_r <= data_s2f_reg;     -- данни към БОС
  data_s2f_u <= dio;
  we <= we_reg;       -- данни към SRAM
  oe <= oe_reg;
  ad <= addr_reg;
  ce <= '0';  -- I/O сигнали за SRAM чипа
  ub <= '0';
  lb <= '0';
  dio <= data_f2s_reg  when tri_reg='0'  else (others =>'Z');    -- при запис
end arch;
```

Синтез на брояч с програмируем модул на броене

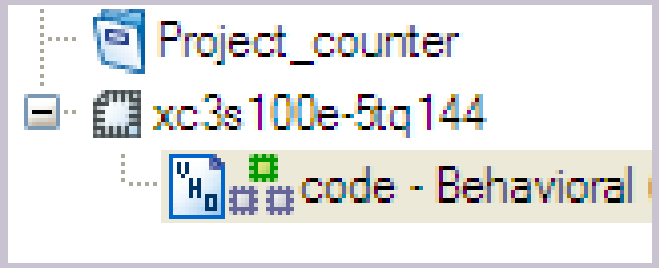
- 6-битова разрядност;
- задаване модула на броене;
- възможност за въвеждане на тактовия сигнал ръчно или от вградения тактов генератор (100MHz);
- реверсивен брояч (режим на сумиране или изваждане).

>> делене на честотата от вградения ТГ;

>> вектори на изходите и на максималния модул на броене;

>> отделни входове за избор на сорса на тактовия сигнал (ръчно/автом.);

>> вход за избор режима на броене.



Синтез на брояч с програмируем модул на броене

Входно-изходни (портови) връзки/променливи

```
entity code is
port(
   CLK: in STD_LOGIC; -- hand-driven clock pulses
   DIR: in STD_LOGIC;  -- '1' subtraction, '0' accumulation
   PROGRAM: in STD_LOGIC; -- when '1' accepts the max.count from PROGRAM exept for MAX(5:0)=0
   AUTOCLOCK: in STD_LOGIC; -- accepts generator pulses (100MHz source)
   AUTO: in STD_LOGIC; -- on-board generator '1' or hand-driven '0'
   CLR: in STD_LOGIC; -- high level asynchronous reset
   MAX:in STD_LOGIC_VECTOR (5 downto 0); --input for maximum count
   Q:inout STD_LOGIC_VECTOR (5 downto 0):="ZZZZZZ" -- current counter value
   );
end code;
```

Синтез на брояч с програмируем модул на броене: архитектура

```
architecture Behavioral of code is
   signal tmp:std_logic_vector(5 downto 0):="000000";
   signal MAXtmp:std_logic_vector(5 downto 0):="000001";
   signal clock:std_logic:='0';

begin

-- 100MHz to 1Hz when AUTO='1'
process (CLK,AUTOCLOCK,CLR,AUTO)
  constant modulus: integer:=100000000;
  variable count: natural range 0 to modulus-1:=0;

begin

if CLR='0' then
    count:=0;
    clock <='0';
elsif AUTO='1' then

    if (AUTOCLOCK'event and AUTOCLOCK='1') then
        if count=modulus-1 then
            count:=0;
            clock<='1';
        else
                count:=count+1;
                clock<='0';
        end if;
    end if;
```

Синтез на брояч с програмируем модул на броене: архитектура

```
else
   clock <=CLK;
end if;

end process;

process (clock,CLR,AUTO,PROGRAM)
  constant delay_time:integer:=3;
  variable delay: natural range 0 to delay_time:=0;
begin

if (PROGRAM'event and PROGRAM='1') then  -- protects the counter in case of modulus=0
   if MAX>"000000" then
      MAXtmp<=MAX;
   end if;
end if;

if CLR='1' then
   tmp<="000000";
elsif clock='1' and clock'event then
```

Синтез на брояч с програмируем модул на броене: архитектура

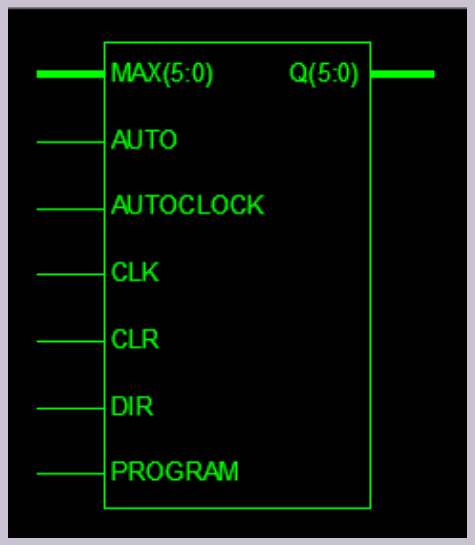
```
if(DIR='0') then  -- addition when DIR=0

    tmp<=tmp+1;
    if tmp>=MAXtmp-1 then

    if AUTO='1' then  -- waits when the max.modul is achieved
        DELAY:=DELAY+1;
        if DELAY>=DELAY_TIME then
            DELAY:=0;
            tmp<="000000";
        else
            tmp<=MAXtmp-1;
        end if;
            elsif AUTO='0' then
                tmp<="000000";
        end if;
    end if;
```

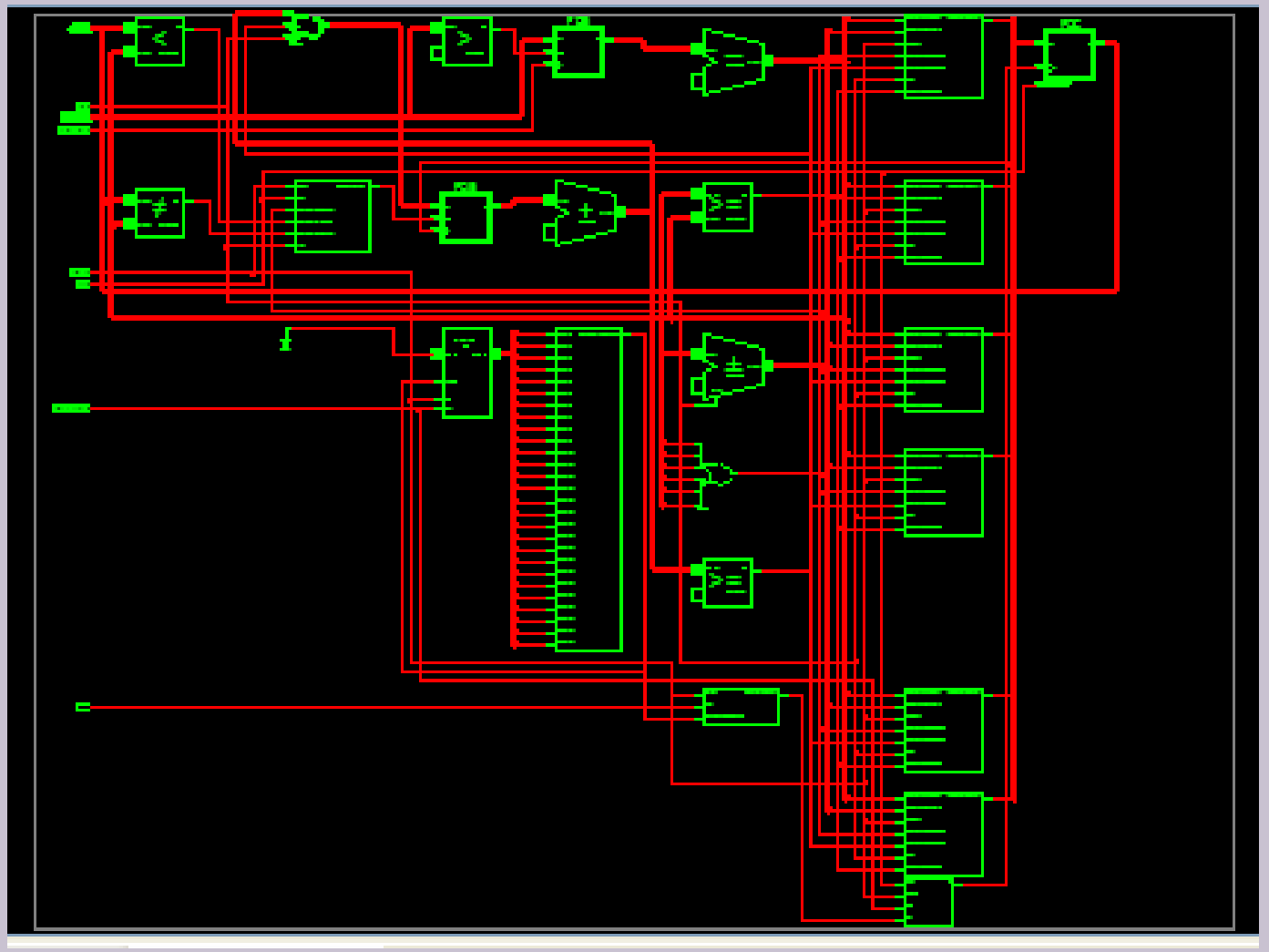
Синтез на брояч с програмируем модул на броене: архитектура

```
   else                    -- subtraction when DIR=1

      if tmp="000000" then
         tmp<=MAXtmp-1;
      elsif tmp=MAXtmp-1 and AUTO='1' then    -- waits

         DELAY:=DELAY+1;
         if DELAY<=DELAY_TIME then
            DELAY:=0;
            tmp<=tmp-1;
         else
            tmp<=MAXtmp-1;
         end if;

      else
         tmp<=tmp-1;
      end if;

   end if;
end if;
end process;
Q<=tmp;

end Behavioral;
```

Синтез на брояч с програмируем модул на броене

Синтезирана RTL-структура





Синтез на брояч с програмируем модул на броене

Използвани ресурси. Тайминг

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

Selected Device : 3s100etq144-5

 Number of Slices:                       68  out of     960      7%
 Number of Slice Flip Flops:             42  out of    1920      2%
 Number of 4 input LUTs:                127  out of    1920      6%
 Number of IOs:                          18
 Number of bonded IOBs:                  18  out of     108     16%
 Number of GCLKs:                         2  out of      24      8%
```

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

   Minimum period: 6.672ns (Maximum Frequency: 149.891MHz)
   Minimum input arrival time before clock: 5.987ns
   Maximum output required time after clock: 4.533ns
```

Синтез на брояч с програмируем модул на броене Implementation Constrains File (.ucf) за тестване на прототипа

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "CLK"  CLOCK_DEDICATED_ROUTE=FALSE ;
NET "AUTO"  LOC = "p38"  ;
NET "AUTOCLOCK"  LOC = "p54"  ;
NET "CLK"  LOC = "p69"  ;
NET "CLR"  LOC = "p47"  ;
NET "DIR"  LOC = "p36"  ;
NET "MAX<0>"  LOC = "p29"  ;
NET "MAX<1>"  LOC = "p24"  ;
NET "MAX<2>"  LOC = "p18"  ;
NET "MAX<3>"  LOC = "p12"  ;
NET "MAX<4>"  LOC = "p10"  ;
NET "MAX<5>"  LOC = "p6"  ;
NET "PROGRAM"  LOC = "p48"  ;
NET "Q<0>"  LOC = "p15"  ;
NET "Q<1>"  LOC = "p14"  ;
NET "Q<2>"  LOC = "p8"  ;
NET "Q<3>"  LOC = "p7"  ;
NET "Q<4>"  LOC = "p5"  ;
NET "Q<5>"  LOC = "p4"  ;

#PACE: Start of PACE Area Constraints
```

Синтез на брояч с програмируем модул на броене:

Резултати от симулация (ModelSim XE 6.0d симулатор на Mentor Graphics)