Combinational architecture :1 חלק

1. להלן מימוש היחידה FA ב VHDL, בקובץ fa.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;


entity FA is
  port (
    A, B, carry_in    : in  std_logic;
    F,    carry_out   : out std_logic );
end FA;


architecture behavioral of FA is
begin
     F <= A XOR B XOR carry_in after 30 ns;
     carry_out <= (A AND B) OR (A AND carry_in) OR (B AND
carry_in) after 40 ns;
end behavioral;
```

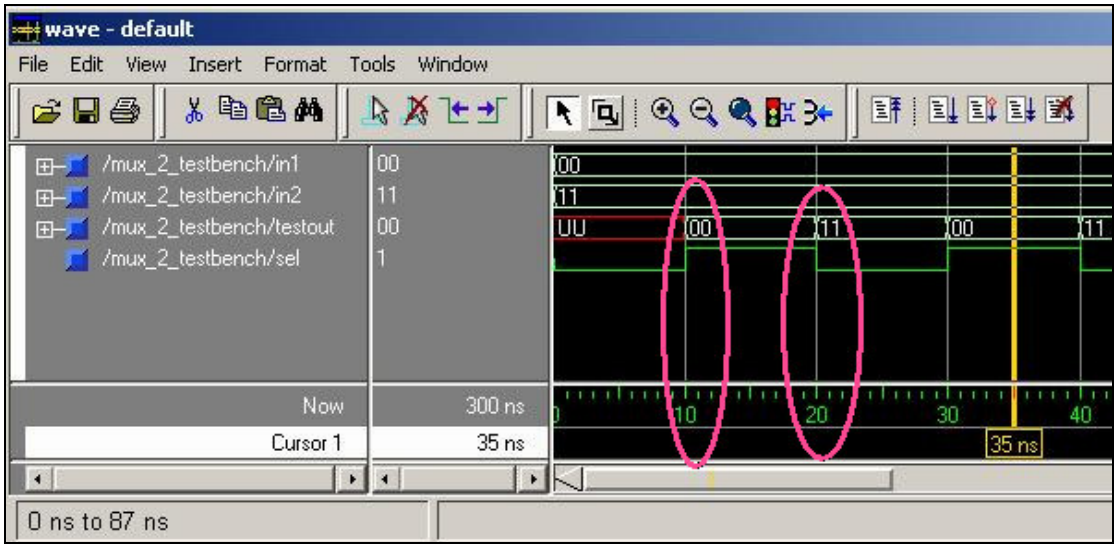
2. להלן מימוש היחידה mux_2 ב VHDL, בקובץ mux_2.vhd:

```
library ieee;
use IEEE.std_logic_1164.all;



entity mux_2 is
  port (
    A, B : in  std_logic_vector (1 downto 0);
    C_in : in std_logic;
    F    : out std_logic_vector (1 downto 0) );
end mux_2;

architecture behavioral of mux_2 is
begin
     with C_in select
     f <= A    after 10 ns    when '0',
          B    after 10 ns    when '1',
          "ZZ" after 10 ns    when others;
end behavioral;
```

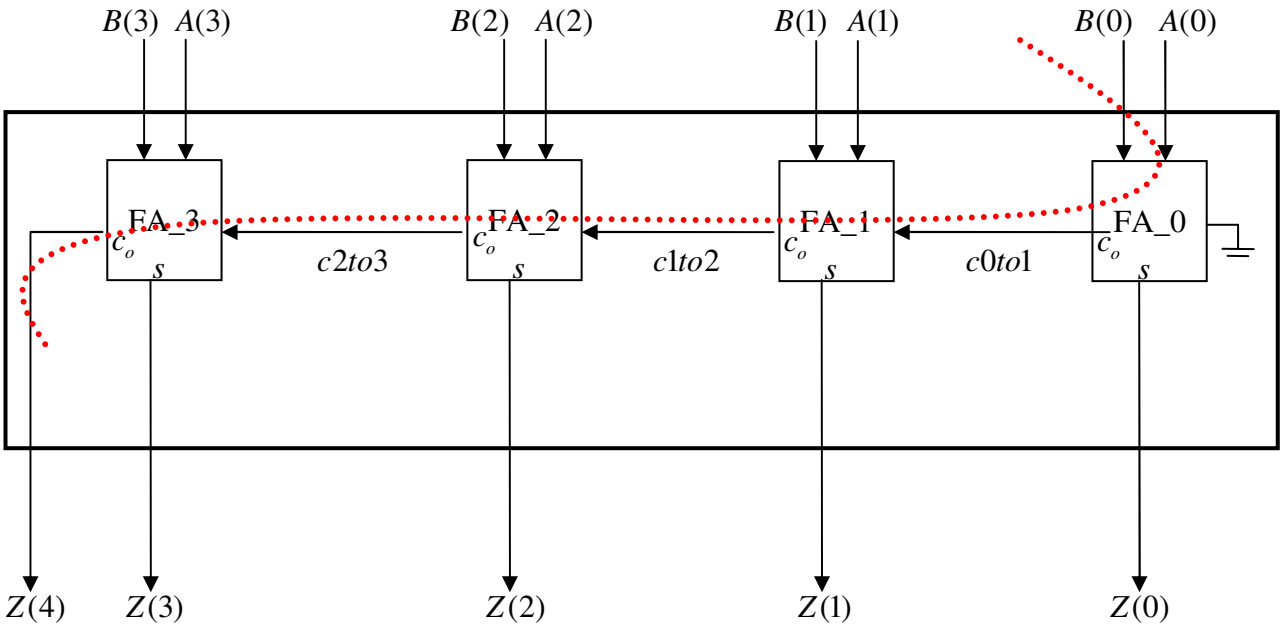
3. מימוש רכיב בדיקה לתת-היחידה mux_2, בקובץ mux_2_testbench.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux_2_testbench is
end mux_2_testbench;

architecture behavioral of mux_2_testbench is
     component mux_2
          port ( A, B : in  std_logic_vector (1 downto 0);
                 C_in : in std_logic;
                 F    : out std_logic_vector (1 downto 0)
          );
     end component;

     signal in1,in2,testout:std_logic_vector (1 downto 0);
     signal sel:std_logic;

begin
     u1: mux_2 port map (in1,in2,sel,testout);
     process
          variable err_cnt: integer := 0;
     begin

          in1 <= "00";
          in2 <= "11";
          sel <='0';
          wait for 10ns;
          assert(testout = "00") report "Error" severity error;
          if (testout /= "00") then
               err_cnt := err_cnt + 1;
          end if;
          sel <='1';
          wait for 10ns;
          assert(testout = "11") report "Error" severity error;
          if (testout /= "11") then
               err_cnt := err_cnt + 1;
          end if;

     end process;

end behavioral;
```

דוגמה ל waveform של בדיקה שהצליחה:



4. מימוש רכיב Add-4 בשיטת Ripple Carry, בעזרת רכיבי FA בלבד.
תאור סכמטי:



המסלול הקריטי מסומן באדום. השהיית המעגל היא $T_L = 4Tpd_{in\to Cout} = 4 \cdot 40 = 160ns$

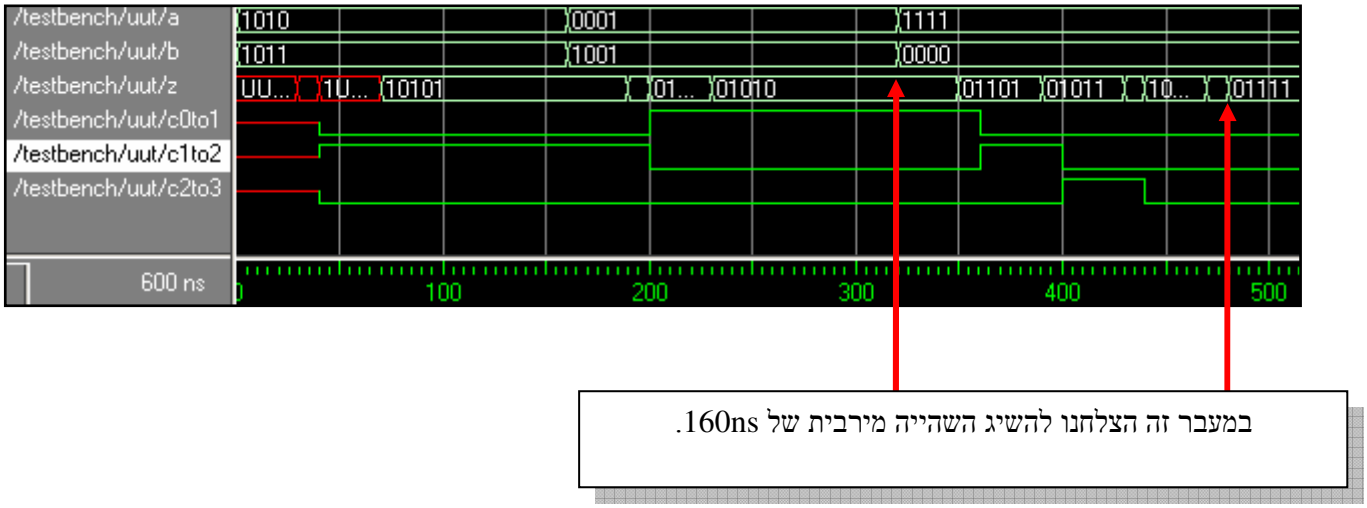
5. מימוש רכיב ה Add-4, בקובץ add_4.vhd:

```
library IEEE;

use IEEE.std_logic_1164.all;

entity add_4 is
  port(
    A, B : in  std_logic_vector (3 downto 0);
    Z    : out std_logic_vector (4 downto 0) );
end add_4;


architecture add_4_arc of add_4 is

component FA
      port ( A, B, carry_in : in std_logic;
             F,   carry_out : out std_logic
      );
end component;

signal c0to1, c1to2, c2to3 : std_logic;

begin
FA_0: FA port map (A=>A(0),B=>B(0),carry_in=>'0'  ,F=>Z(0),carry_out=>c0to1);

FA_1: FA port map (A=>A(1),B=>B(1),carry_in=>c0to1,F=>Z(1),carry_out=>c1to2);

FA_2: FA port map (A=>A(2),B=>B(2),carry_in=>c1to2,F=>Z(2),carry_out=>c2to3);

FA_3: FA port map (A=>A(3),B=>B(3),carry_in=>c2to3,F=>Z(3),carry_out=>Z(4));

end add_4_arc;
```

6. מימוש רכיב הבדיקה בקובץ add_4_testbench.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity testbench is
  generic (Tpd : time := 160 ns);
end testbench;
architecture arc_testbench of testbench is
  component add_4
    port (
      A, B    : in  std_logic_vector (3 downto 0);
      Z       : out std_logic_vector (4 downto 0) );
  end component;
  signal A,B : std_logic_vector (3 downto 0);
  signal Z   : std_logic_vector (4 downto 0);
begin
  UUT: component add_4 port map (A,B,Z);  --Unit Under Test
  process
  begin
    A <= "1010";
    B <= "1011";
    wait for Tpd;
    A <= "0001";
    B <= "1001";
    wait for Tpd;
    A <= "1111";
    B <= "0000";
    wait;                -- needed to prevent entering the process again.
  end process;
end arc_testbench;
configuration cfg_testbench of testbench is
  for arc_testbench
    for UUT : add_4
      use entity work.add_4;
    end for;
  end for;
end cfg_testbench;
```

זוהי דוגמה ל waveform שמראה שהרכיב תקין. גרמנו לרכיב להוציא את הצירוף 10101 ואז את 01010 וכך וידאנו שאין תקלות בצמתים שגוררות תקלות stuck-at – כל ביט ביציאה מסוגל להתעדכן.

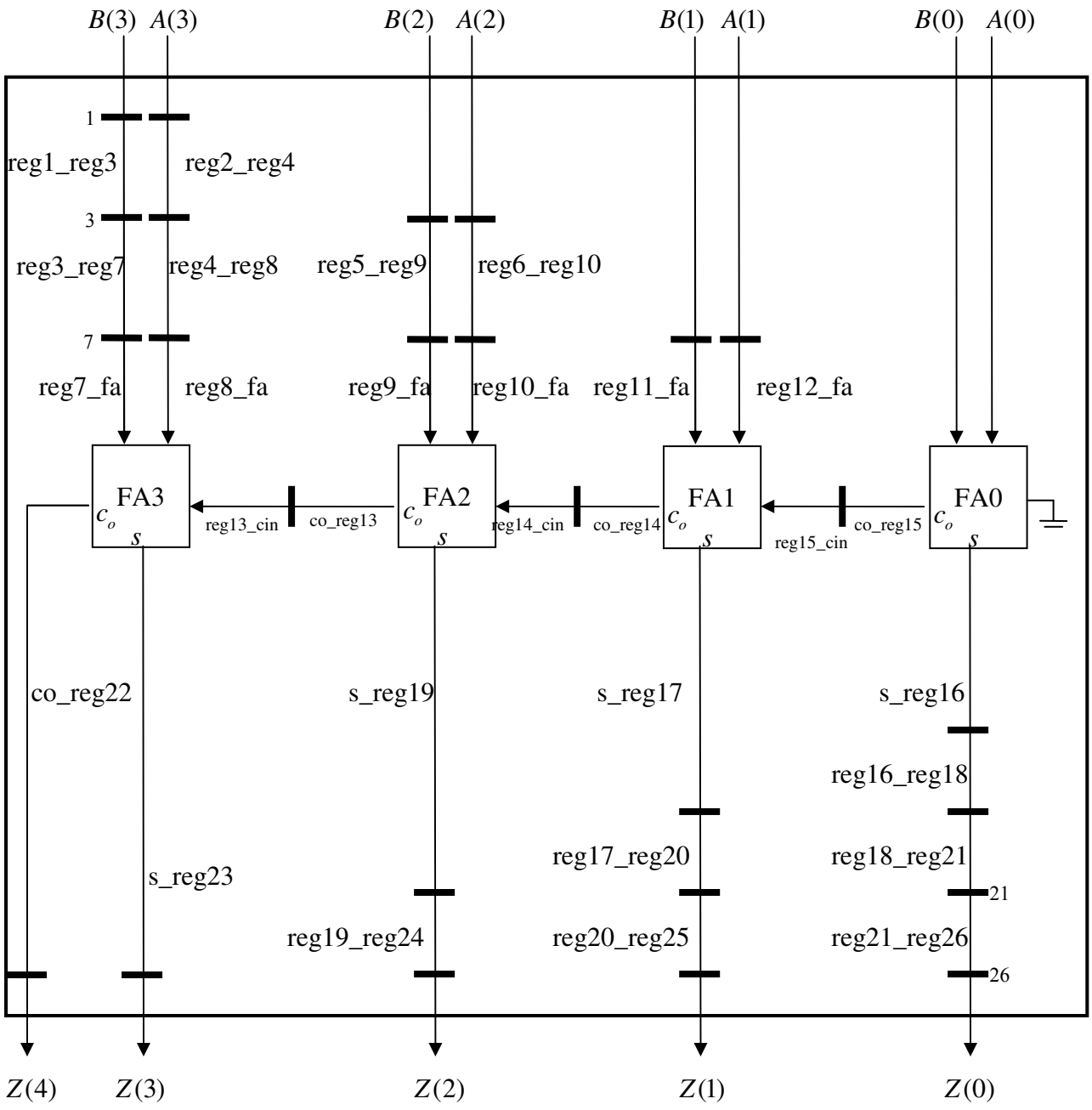


חלק 2: pipelined architecture

7. מימוש רגיסטר לביט בודד הדוגם בעליית שעון, בקובץ reg_bit.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity bit_reg is
  port(
    clk   : in  std_logic;
    d_in  : in  std_logic;
    d_out : out std_logic );
end bit_reg;

architecture bit_reg_arc of bit_reg is
begin
     process (clk)
     begin
           if (clk'event and clk='1') then
               d_out <= d_in after 15 ns;
           end if;
     end process;
end bit_reg_arc;
```

8. שירטוט של המחבר המצונר:



עומק הצינור: $k=4$, מספר אוגרים: $n=26$

זמן מחזור מינימאלי: $T_{cycle}=(40+15)ns=55ns$ ומכאן תדר העבודה המקסימאלי: $f_{\max}=\frac{1}{55ns}=18.2MHz$

ספיקה: $TP=\frac{1}{T_{cycle}}=\frac{1}{55}$

שיהוי: $T_L=kT_{cycle}=220ns$

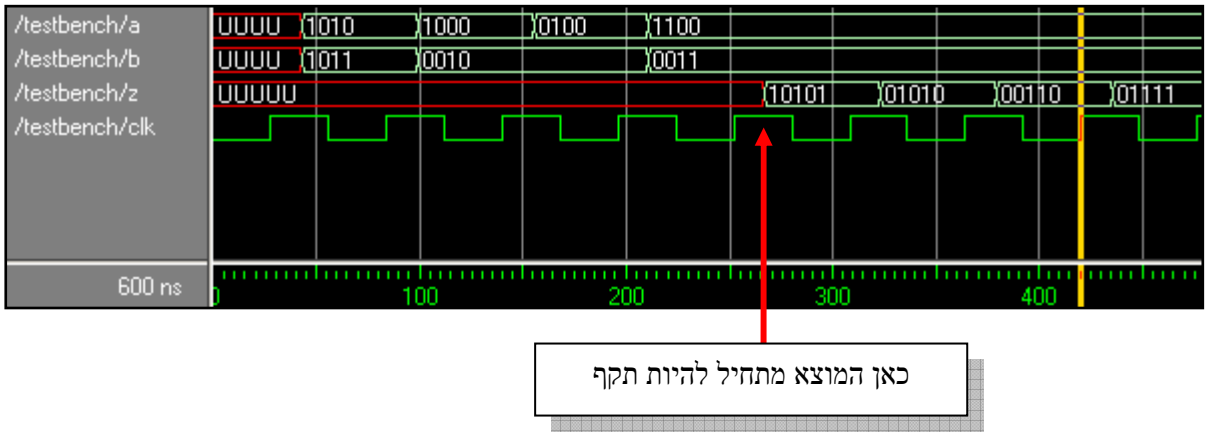
9. מימוש המחבר המצונר בקובץ add_4_pipe.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity add_4_pipe is
  port(
    A, B    : in  std_logic_vector (3 downto 0);
    Z       : out std_logic_vector (4 downto 0);
    clk     : in std_logic);
end add_4_pipe;

architecture add_4_pipe_arc of add_4_pipe is
component FA
       port ( A, B, carry_in : in std_logic;
              F,   carry_out : out std_logic
       );
end component;
component bit_reg
       port (  clk,d_in: in std_logic;
              d_out: out std_logic);
end component;
signal reg1_reg3,reg3_reg7,reg7_fa,
       reg2_reg4,reg4_reg8,reg8_fa,
       reg5_reg9,reg9_fa,
       reg6_reg10,reg10_fa,
       reg11_fa,
       reg12_fa,
       co_reg15,reg15_cin,
       co_reg14,reg14_cin,
       co_reg13,reg13_cin,
       co_reg22,
       s_reg16,reg16_reg18,reg18_reg21,reg21_reg26,
       s_reg17,reg17_reg20,reg20_reg25,
       s_reg19,reg19_reg24,
       s_reg23 : std_logic;

begin
       FA_0: FA port map(A(0)    ,B(0)    ,'0'      ,s_reg16,co_reg15);
       FA_1: FA port map(reg11_fa,reg12_fa,reg15_cin,s_reg17,co_reg14);
       FA_2: FA port map(reg9_fa ,reg10_fa,reg14_cin,s_reg19,co_reg13);
       FA_3: FA port map(reg7_fa ,reg8_fa ,reg13_cin,s_reg23,co_reg22);

       reg_01: bit_reg port map (clk, B(3)      , reg1_reg3);
       reg_02: bit_reg port map (clk, A(3)      , reg2_reg4);
       reg_03: bit_reg port map (clk, reg1_reg3 , reg3_reg7);
       reg_04: bit_reg port map (clk, reg2_reg4 , reg4_reg8);
       reg_05: bit_reg port map (clk, B(2)      , reg5_reg9);
       reg_06: bit_reg port map (clk, A(2)      , reg6_reg10);
       reg_07: bit_reg port map (clk, reg3_reg7 , reg7_fa);
       reg_08: bit_reg port map (clk, reg4_reg8 , reg8_fa);
       reg_09: bit_reg port map (clk, reg5_reg9 , reg9_fa);
       reg_10: bit_reg port map (clk, reg6_reg10, reg10_fa);
       reg_11: bit_reg port map (clk, B(1)      , reg11_fa);
       reg_12: bit_reg port map (clk, A(1)      , reg12_fa);

       reg_13: bit_reg port map (clk, co_reg13  , reg13_cin);
       reg_14: bit_reg port map (clk, co_reg14  , reg14_cin);
       reg_15: bit_reg port map (clk, co_reg15  , reg15_cin);

       reg_16: bit_reg port map (clk, s_reg16     , reg16_reg18);
       reg_17: bit_reg port map (clk, s_reg17     , reg17_reg20);
       reg_18: bit_reg port map (clk, reg16_reg18 , reg18_reg21);
       reg_19: bit_reg port map (clk, s_reg19     , reg19_reg24);
       reg_20: bit_reg port map (clk, reg17_reg20 , reg20_reg25);
       reg_21: bit_reg port map (clk, reg18_reg21 , reg21_reg26);
       reg_22: bit_reg port map (clk, co_reg22    , Z(4));
       reg_23: bit_reg port map (clk, s_reg23     , Z(3));
       reg_24: bit_reg port map (clk, reg19_reg24 , Z(2));
       reg_25: bit_reg port map (clk, reg20_reg25 , Z(1));
       reg_26: bit_reg port map (clk, reg21_reg26 , Z(0));
end add_4_pipe_arc;
```

10. מימוש רכיב הבדיקה למחבר המצונר, בקובץ add_4_pipe_testbench.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity testbench is
  generic (CLK_CYCLE : time := 56 ns);
end testbench;
architecture arc_testbench of testbench is
  component add_4_pipe is
  port (
    A,B     : in  std_logic_vector (3 downto 0);
    Z       : out std_logic_vector (4 downto 0);
    clk     : in std_logic);
  end component;
  signal A,B : std_logic_vector (3 downto 0);
  signal Z   : std_logic_vector (4 downto 0);
  signal clk : std_logic := '0';        -- clk signal, reset to '0' on
 begin
   UUT: component add_4_pipe port map (A,B,Z,clk);  --Unit Under Test
   process
   begin
     wait until ( (clk'event) and (clk = '1') );
     wait for 15 ns;
     A <= "1010";
     B <= "1011";
     wait until ( (clk'event) and (clk = '1') );
     wait for 15 ns;
     A <= "1000";
     B <= "0010";
     wait until ( (clk'event) and (clk = '1') );
     wait for 15 ns;
     A <= "0100";
     B <= "0010";
     wait until ( (clk'event) and (clk = '1') );
     wait for 15 ns;
     A <= "1100";
     B <= "0011";
         wait;
   end process;
   process
   begin
     wait for CLK_CYCLE/2;
     clk <= not clk;
   end process;
end arc_testbench;
configuration cfg_testbench of testbench is
  for arc_testbench
    for UUT : add_4_pipe
      use entity work.add_4_pipe;
    end for;
  end for;
end cfg_testbench;
```

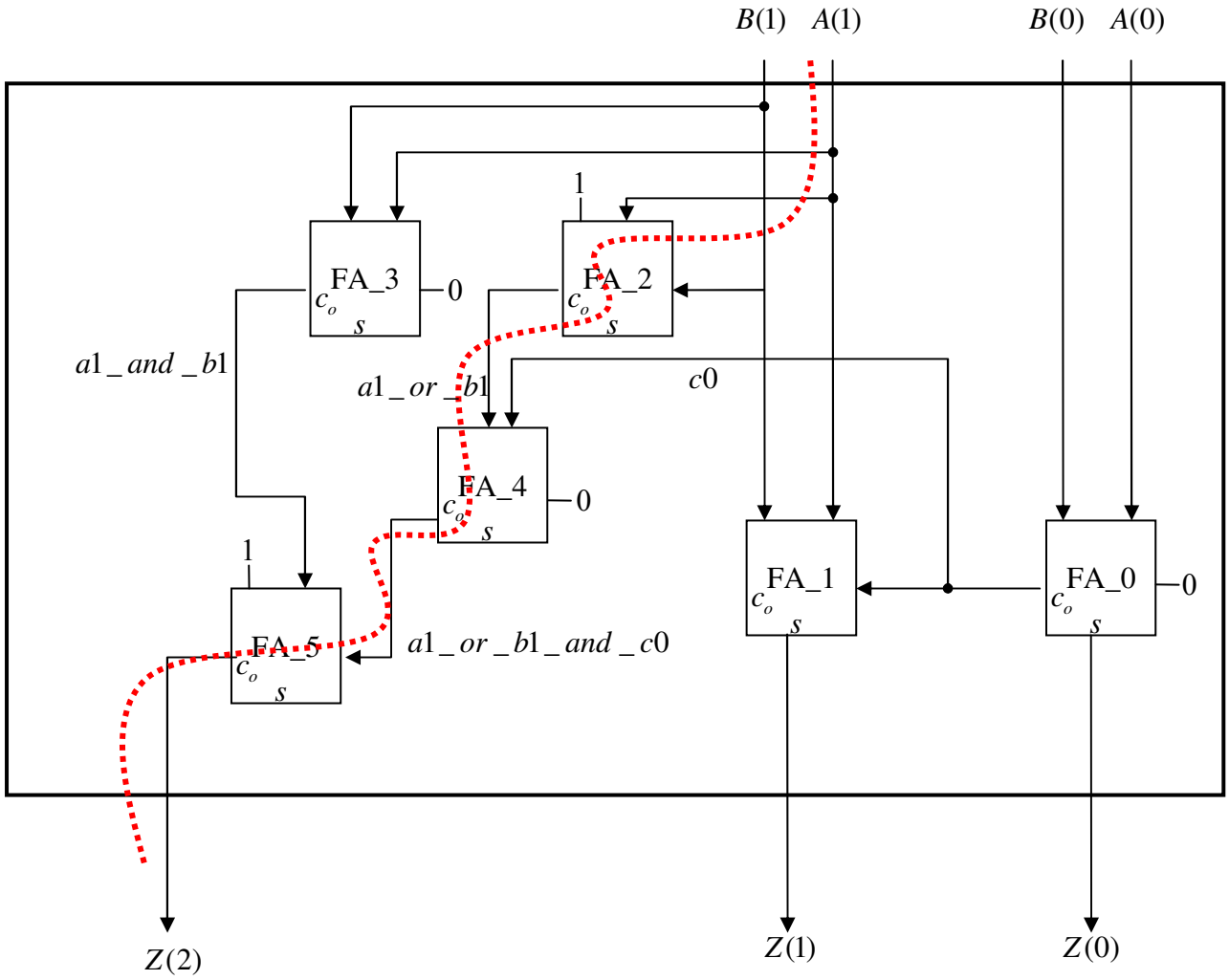
דוגמה ל waveform שמראה שהבדיקה הצליחה:



* היתה בעיה עם זמן מחזור של 55ns, כניראה בגלל החלוקה ב 2 של מספר איזוגי זה, ולכן שיננו ל 56ns.

11. ראשית, הסינתיסייזר של VHDL לא מכיר במשתני time וגם לא בפקודת wait for וְ- after, כך שפורמלית פקודות אלו הן לא סינתזביליות.
אתחול השעון, כפי שהוא מתבצע אינו סינתזבילי. ההשהיה המלאכותית wait cycle/2 לא ניתנת למימוש ע"י שערים לוגיים, לעומת ההשהיות wait 15 שלאחר כל עליית שעון שהן כן ניתנות למימוש – הן בעצם ההשהיות האמיתיות של הלוגיקה שבונה את האוגרים שבכניסה לרכיב ה Add 4 המצונר.
כל מה שפקודות ההשייה עושות הוא דימוי של ההשהייה של השערים הלוגיים, לטובת סימולציה של הרכיבים שבנינו.

בונוס



ע"פ המודל של תכנון ה Carry Look Ahead, חישוב הנשא מתבצע ע"י: $c_{i+1} = A_i B_i + (A_i + B_i) C_i$
השהיית המעגל: $T_{pd} = 3 \cdot (Tpd_{in \to cout}) = 3 \cdot 40ns = 120ns$
המסלול בעל ההשהייה הגדולה ביותר צבוע באדום.
חסרון גדול למימוש זה – שיהוי גדול מאוד.
יתרון פוטנציאלי: שיטת החישוב של ה Carry יעילה ותתן זמן שיהוי קטן יותר מהמחבר בשיטת Ripple Carry, עבור חיבור שני מספרים בעלי n סיביות, כאשר n גדול.

מימוש רכיב ה Add-2, בקובץ add_2_cla.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity add_2_cla is
  port(
    A, B    : in  std_logic_vector (1 downto 0);
    Z       : out std_logic_vector (2 downto 0));
end add_2_cla;

architecture add_2_cla_arc of add_2_cla is

component FA
      port ( A, B, carry_in : in std_logic;
             F,   carry_out : out std_logic
      );
end component;

signal
      c0,
      a1_and_b1,
      a1_or_b1,
      a1_or_b1_and_c0,
      useless : std_logic;

begin
      FA_0: FA port map(A(0),B(0)     ,'0'            ,Z(0),c0);
      FA_1: FA port map(A(1),B(1)     ,c0             ,Z(1));
      FA_2: FA port map('1' ,A(1)     ,B(1)           ,useless,a1_or_b1);
      FA_3: FA port map(A(1),B(1)     ,'0'            ,useless,a1_and_b1);
      FA_4: FA port map(a1_or_b1,c0,'0'      , useless,a1_or_b1_and_c0);
      FA_5: FA port map('1' ,a1_and_b1,a1_or_b1_and_c0,useless,Z(2));
end add_2_cla_arc;
```