מעבדה לVHDL – ניסוי UART

תכנון משדר UART:

החלטנו לתכנן את המשדר בצורת מכונת מצבים.
המכונת מצבים תהיה במצב הנקרא IDLE כל עוד אנחנו לא שולחים מידע במצב זה אנו נשלח '1' לוגי קבוע.
למערכת הגדרנו כניסה הנקראת go שכאשר היא יורדת ל-'0' המערכת שולחת את המידע שנכנס אליה באופן מקבילי בצורה טורית לפי התקן של תקשורת UART.

תקן UART אומר שאנו שולחים מידע באופן טורי ואסינכרוני בהדק הנקרא TX בגודל 8 סיביות.
הPacket מורכב מסיבית התחלה שמוגדרת על פי התקן כ- '0' לוגי לאחר מכן אנו שולחים שמונת סיביות המידע כאשר הסיבית הראשונה שנשלחת היא סיבית ה-LSB והסיבית האחרונה היא סיבית ה-MSB.
לאחר שליחת המידע אנו שולחים סיבית הנקראת סיבית זוגיות (סיבית זו מבצעת השלמה לזוגיות לפי מספר '1' שנשלח במידע).
לאחר מכן אנו שולחים סיבית עצירה (שמוגדרת לפי התקן כ-'1') וחוזרים למצב שבו אנחנו לא שולחים מידע (IDLE) במצב זה אנו שולחים '1' לוגי קבוע.
מכיוון שמדובר בתקשורת אסינכרונית לכל סיבית יש זמן שליחה מוגדר לפי התדר שליחה שנקבע על ידי מתכנן המערכת.
אנו נדרשנו בניסוי זה לממש מערכת שמשדרת מידע בקצב שליחת סיביות: 9600[BPS].
מכיוון שבכרטיס יש כבר clk בתדר 50[MHz] היינו צריכים להשתמש ב- Generator כ-Component לצורך יחוקה בתדר.

חישוב מחלק תדר:

$$M = \frac{f_{in}}{f_{out}} = \frac{50[MHz]}{9.6[KHz]} = 5208$$

קיבלנו שצריך לחלק את התדר ב – 5208.
בשביל להגדיר D.C = 50[%] נצטרך להגדיר שכל זמן שהמונה ב-Generator לא הגיע לחצי מהזמן של 5208 הוא צריך להוציא '0' ואחרי שהמונה עובר את החצי מ-5208 הוא מוציא '1'.

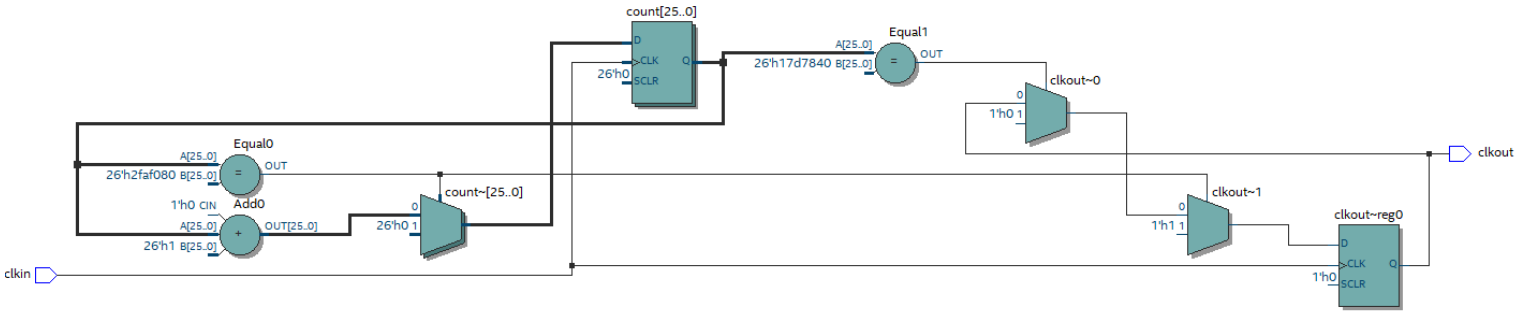
$$\frac{5208}{2} = 2604$$

לצורך קביעת תדר שעון למערכת השתמשנו בGenerator שתכננו בניסויים קודמים שמבצע את החלוקה בתדר

תוכנית ה-Generator:

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

entity Generator is
generic(N:  integer:= 50000000;
            DC: integer:= 25000000);
port(
clkin: in std_logic;
clkout: buffer std_logic:='1');
end;

architecture one of Generator is
signal count: integer range 0 to N := 0;
begin
process(clkin)
begin
if (clkin 'event and clkin = '1') then
      if (count = N) then
            count <= 0;
            clkout <= '1';
      elsif (count = DC) then
            count <= count + 1;
            clkout <= '0';
      else
            count <= count + 1;
      end if;
end if;
end process;
end;
```

סכימת RTL:

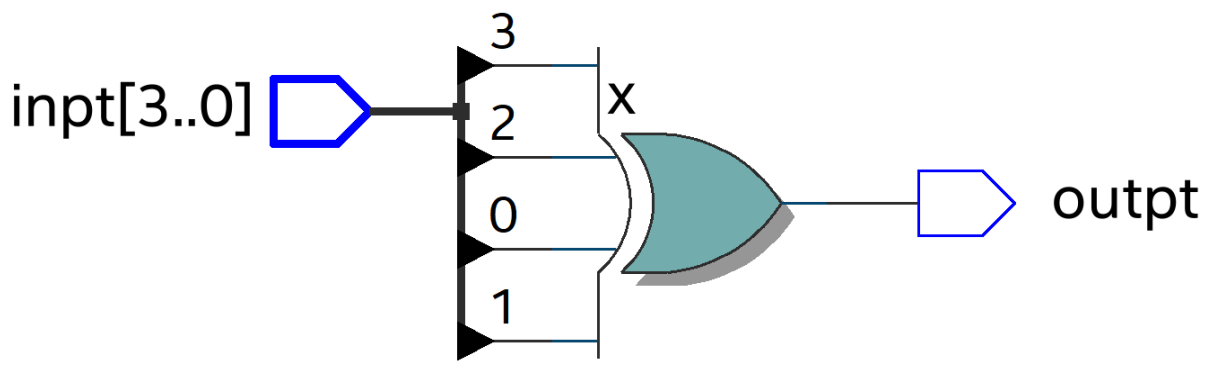


לצורך מימוש סיבית הזוגיות תכננו מערכת הנקראת Parity_Gen שמבצעת פעולת XOR בין כל סיביות המידע.

<u>תוכנית:</u>

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

entity Parity_Gen is
generic (N: integer:=4);
port(
inpt: std_logic_vector(N-1 downto 0);
outpt: out std_logic);
end;

architecture one of Parity_Gen is
begin
process(inpt)
variable x: std_logic;
begin
x:='0';
for i in 0 to N-1 loop
x := x xor inpt(i);
end loop;
outpt <= x;
end process;
end;
```

<u>סכימת RTL:</u>



כפי שניתן לראות הגדרנו באופן כללי את ווקטור הכניסה בגודל 4Bit.

כאשר נקרא לו כ-Component למערכת שלנו נגדיר ב – Generic map את גודל ווקטור הכניסה כ – 8Bit.

מימוש המשדר:

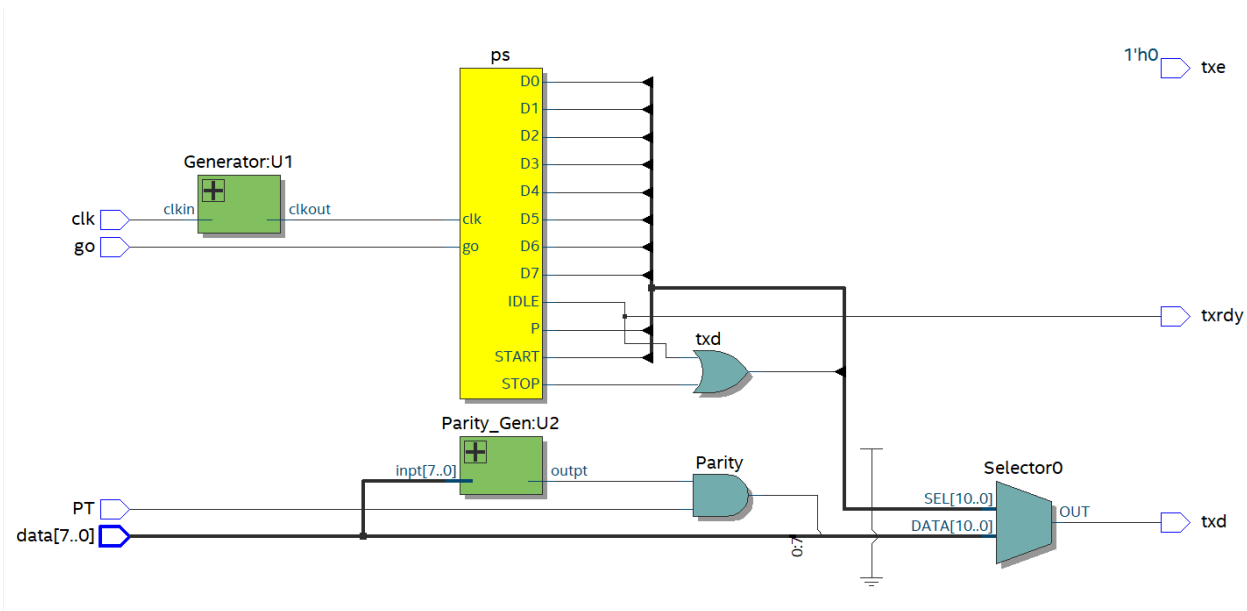
מימשנו את המשדר על סמך התקן שהסברתי בשלב התיאוריה.

התוכנית:

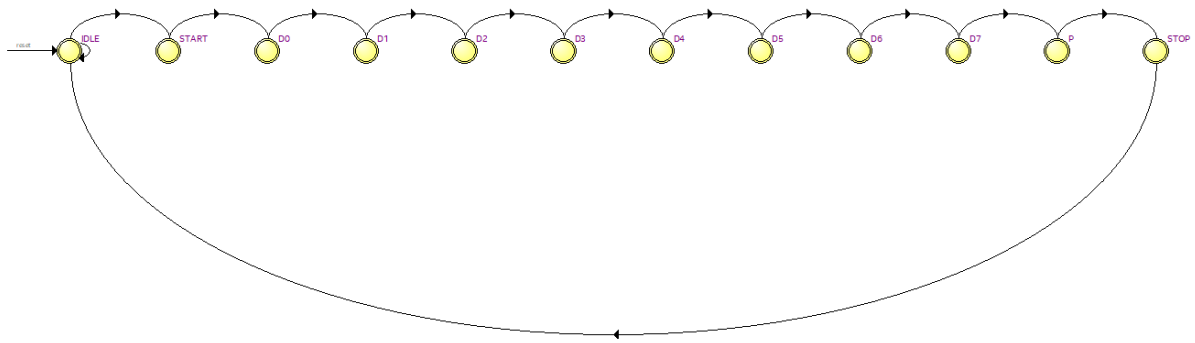
```
library ieee;
use ieee.std_logic_1164.all;

entity Transmitter is
port(
data: in std_logic_vector(7 downto 0);
clk,PT,go: in std_logic;
txd,txrdy,txe: out std_logic);
end;

architecture one of Transmitter is

component Parity_Gen
generic (N: integer:=4);
port(
inpt: std_logic_vector(N-1 downto 0);
outpt: out std_logic);
end component;

component Generator
generic(N:  integer:= 50000000;
             DC: integer:= 25000000);
port(
clkin: in std_logic;
clkout: buffer std_logic:='1');
end component;

type state is (IDLE,START,D0,D1,D2,D3,D4,D5,D6,D7,P,STOP);
signal ps,ns: state;
signal Parity,parity_xor: std_logic;

--for Prescaler the clk (50[MHz] ---> 9600[BPS])
signal Transclk: std_logic;

begin

process(Transclk)
begin
if (Transclk 'event and Transclk = '1') then ps <= ns; end if;
end process;

U1: Generator generic map (5208,2604) port map (clk,Transclk);

process (ps,go)
begin
case ps is
when IDLE  => txd <= '1';  txrdy <= '1'; if (go = '0') then ns <= START; else
ns <= IDLE; end if;
when START => txd <= '0';  txrdy <= '0'; ns <= D0;
when D0 => txd <= data(0); txrdy <= '0'; ns <= D1;
```

```
when D1 => txd <= data(1); txrdy <= '0'; ns <= D2;
when D2 => txd <= data(2); txrdy <= '0'; ns <= D3;
when D3 => txd <= data(3); txrdy <= '0'; ns <= D4;
when D4 => txd <= data(4); txrdy <= '0'; ns <= D5;
when D5 => txd <= data(5); txrdy <= '0'; ns <= D6;
when D6 => txd <= data(6); txrdy <= '0'; ns <= D7;
when D7 => txd <= data(7); txrdy <= '0'; ns <= P;
when P  => txd <= Parity;  txrdy <= '0';ns <= STOP;
when STOP => txd <= '1'; txrdy <= '0'; ns <= IDLE;
end case;
end process;

U2: Parity_Gen Generic map (8) port map (data,parity_xor); -- (Parity_Xor =
'1' --> even) , (Parity_Xor = '0' --> odd)
Parity <= '1' when ((parity_xor = '1') and (PT = '1')) else '0'; -- (PT = '1'
--> even) , (PT = '0' --> odd)
end;
```

סכימת RTL:



דיאגרמת מצבים:



טבלת מעברים:

	Source State	Destination State	Condition
1	D0	D1	
2	D1	D2	
3	D2	D3	
4	D3	D4	
5	D4	D5	
6	D5	D6	
7	D6	D7	
8	D7	P	
9	IDLE	START	(!go)
10	IDLE	IDLE	(go)
11	P	STOP	
12	START	D0	
13	STOP	IDLE	

סימולציה:

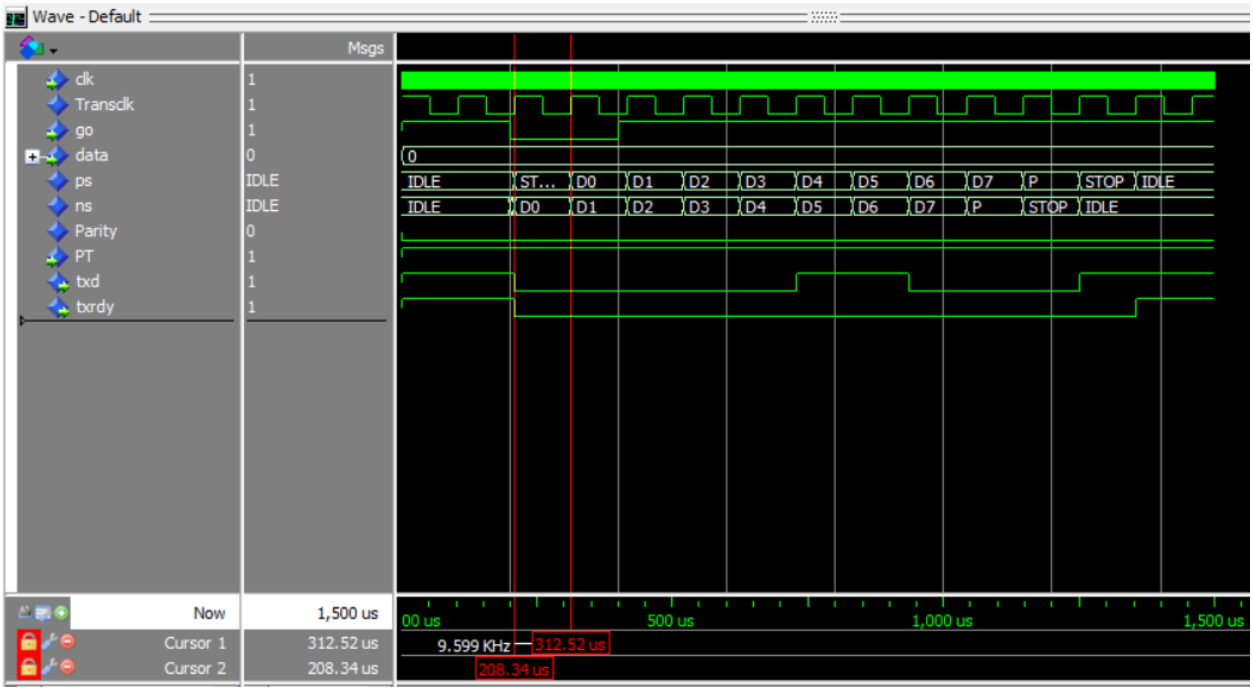
ביצענו קומפילציה בסימולציה של ModelSim בכדי לבדוק שהמערכת עובדת באופן תקין.
מטרתנו הייתה לבדוק שאנו שולחים את המידע 0x30 (בקידוד ASCII נתון זה שווה למספר 0) המידע נשלח בצורה תקינה.

הפקודות שהכנסו לScript:

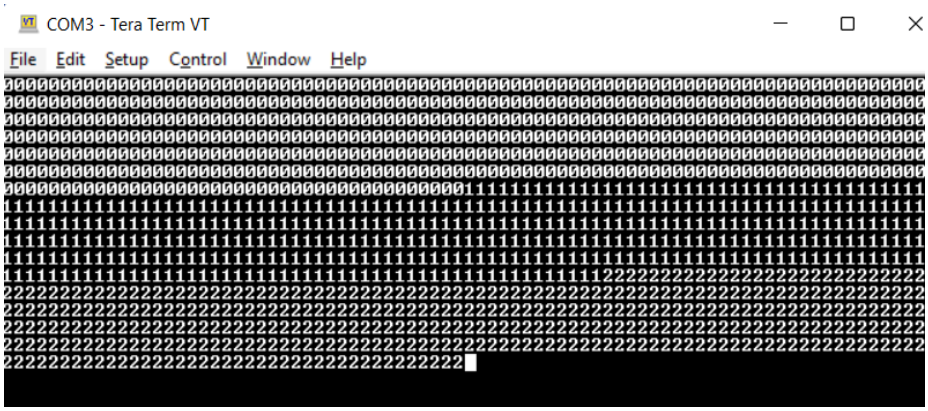
1. הגדרת clk בתדר 50MHz.
2. Force go 1,0 200us, 1 400us
3. Force data 00110000
4. Force pt 1
5. Run 1.5ms

ניתן לראות שבדקנו את המצב שבו אנו מוסיפים '1' בסיבית זוגיות רק כאשר מקבלים מספר אי-זוגי של '1' (כדי להשלים לזוגיות).

<u>תרשים זמנים (Waveform):</u>



לאחר הצריבה של המשדר ביצענו את הניסוי ושלחנו למחשב קודי ASCII (0,1,2...)



מימוש המקלט

כדי לממש את המקלט היינו צריכים לזהות את הרגע בו מתבצעת ירידה בקו הקבלת נתונים ומאותו הרגע להתחיל לספור זמן של שליחת 11 סיביות (סיבית התחלה + 8 סיביות מידע + סיבית זוגיות + סיבית עצירה).

לצורך מימוש המקלט ביצענו תכנון לשני יחידות:
יחידה אחת אשר אחראית על סנכרון המידע (היא הופכת את המידע שעל הקו מאסינכרוני לסינכרוני על ידי זיהוי הירידה בקו קבלת הנתונים).
ברגע שיחידה זו מזהה ירידה בקו היא מוציאה 11 פולסי שעון בתדר המתאים לקצב השליחה (במקרה שלנו 9600[BPS]).
היחידה השנייה יכולה לבצע קריאה של הנתונים בקלות מכיוון שהיא מקבלת כבר את המידע מסונכרן (כאן מימשנו סוג של אוגר הזזה שמקבל מידע טורי סינכרוני ומוציא מידע מקבילי).

תכנון יחידת הסנכרון:

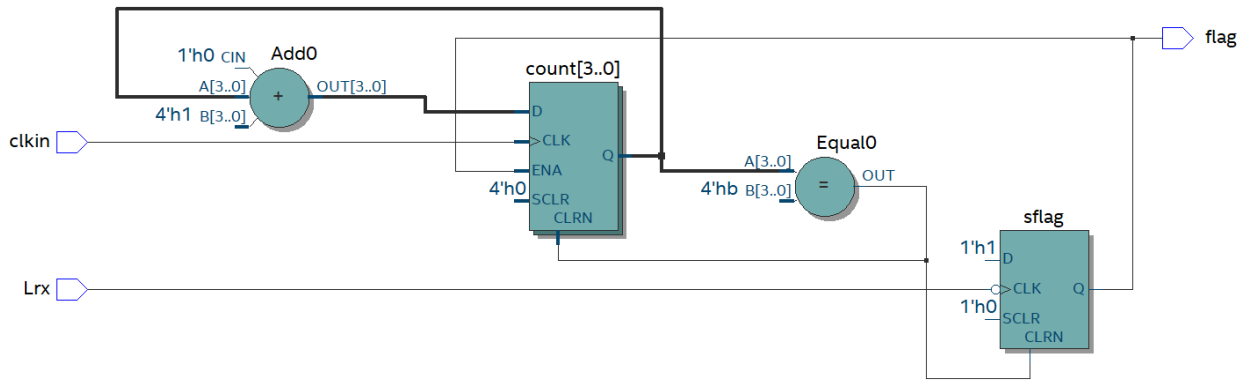
יחידה זו מורכבת גם כן משני יחידות העובדות במקביל.
היחידה הראשונה נקראת Receive_Enable.
יחידה זו אחראית על הזיהוי ירידה בקו, ברגע שמתבצעת ירידה אנחנו מוציאים במוצא מערכת זו '1'.
מהרגע שהמוצא עולה ל-'1' המערכת מתחילה לספור עליות שעון מהכניסה השנייה שלה, וברגע שהיא הגיעה ל-11 עליות שעון המוצא מתאפס.
יציאה זו בעצם נותנת לנו אות אפשור המייצג הזמן בו נשלח המידע. כלומר, כל עוד נשלח מידע מוצא המערכת יהיה ב-'1' אם לא נשלח מידע מוצא המערכת יורד ל-'0'.

תוכנית Receive_Enable:

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

entity Recieve_Enable is
port(
Lrx,clkin: in std_logic;
flag: out std_logic);
end;

architecture one of Recieve_Enable is

signal count:  integer range 0 to 11   := 0;
signal sflag: std_logic := '0';


begin
process(Lrx,clkin)
begin

if (Lrx 'event and Lrx = '0') then
      if (sflag = '0') then
            sflag <='1';
      end if;
end if;

if ((clkin 'event and clkin = '1') and (sflag='1')) then count <= count + 1;
end if;
```

```
if (count = 11) then
      count <= 0;
      sflag <= '0';
end if;

end process;
flag <= sflag;
end;
```

סכימת RTL:



היחידה השנייה ביחידת הסינכרון נקראת SyncRecieveClk אחראית על קבלת אות השעון בתדר 50[MHz] והיא מוציאה אות שעון חדש המתאים לקצב שליחת הסיביות (9600[BPS]) כאשר אות השעון החדש מסונכרן גם עם הירידה של הקו ל-'0' בסיבית התחלה.

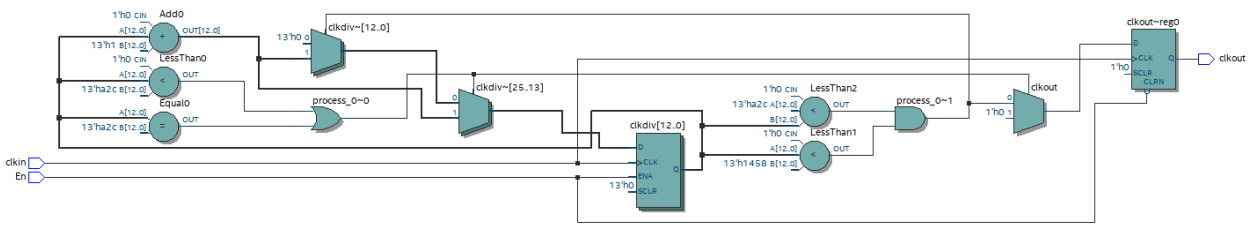
תוכנית זו בעצם מבצעת פעולה דומה לפעולה של ה-Generator (היא מבצעת חלוקה בתדר) אך הוספנו לה עוד שדרוג קטן שהעלייה במונה של המחלק תדר יתחיל ברגע שמתבצע ירידה בקו תקשורת.

תוכנית SyncRecieveClk:

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

entity SyncRecieveClk is
port(
clkin,En: in  std_Logic;
clkout:   out std_logic);
end;

architecture one of SyncRecieveClk is

signal clkdiv: integer range 0 to 5208 := 0;

begin
```

```
process(En,clkin)
begin
if (En = '1') then
      if (clkin 'event and clkin = '1') then
            if ((clkdiv < 2604) or (clkdiv = 2604)) then
                  clkdiv <= clkdiv+1;
                  clkout<='0';
            elsif ((clkdiv < 5208) and (clkdiv > 2604)) then
                  clkdiv <= clkdiv + 1;
                  clkout <= '1';
            else
                  clkdiv <= 0;
                  clkout <= '0';
            end if;
      end if;
else
      clkout <= '0';
end if;
end process;

end;
```

<u>סכימת RTL:</u>

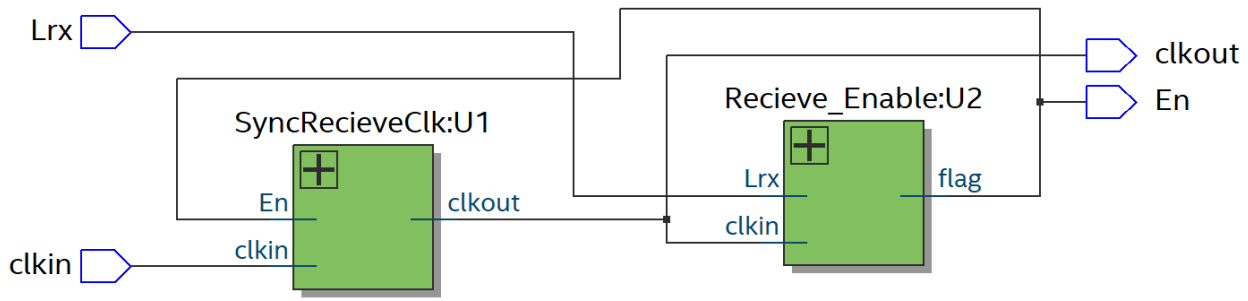


לאחר תכנון שני היחידות האלה חיברנו אותם ויצרנו יחידה אחת המבצעת סינכרון של המידע.
יחידה זו נקראת sync_Reciever.
תוכנית:

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

entity sync_Reciever is
port(
clkin,Lrx: in      std_logic;
En,clkout: buffer std_logic);
end;

architecture one of sync_Reciever is

component syncRecieveClk
port(
clkin,En: in  std_Logic;
clkout:   out std_logic);
```

```
end component;

component Recieve_Enable
port(
Lrx,clkin: in std_logic;
flag: out std_logic);
end component;

begin
U1: syncRecieveClk port map (clkin, En, clkout);
U2: Recieve_Enable  port map (Lrx, clkout, En);
end;
```

<u>סכימת RTL:</u>



במערכת זו יש שני כניסות:
Lrx – RX Line: קו התקשורת שאחראי על קבלת המידע.
clkin – כניסת clk בתדר 50[MHz].

יציאות המערכת:
clkout – אות שעון שאחראי על סנכרון המידע (בכל סיבית שנשלחת הוא מבצע עליית שעון) אות השעון הזה מותאם לתדר של 9.6[KHz] שזה בדיוק קצב שליחת הנתונים (9600[BPS]).
En – יציאה זו מוציאה '1' לוגי כל זמן שהמערכת מזהה מצב של קבלת מידע.

<u>תכנון אוגר הזזה לקריאת נתונים בצורה סינכרונית:</u>

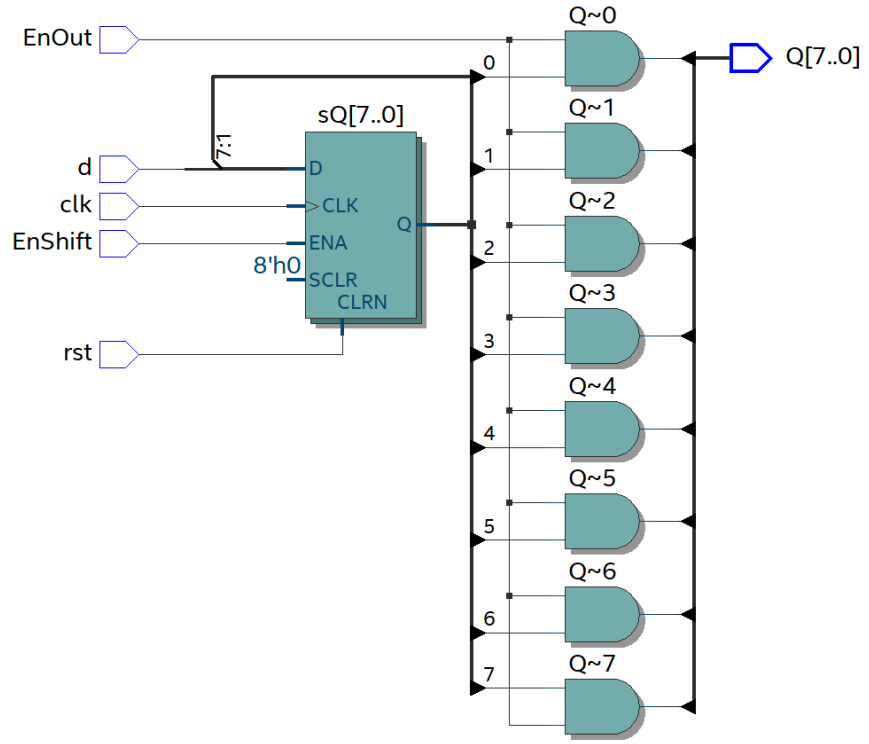
בשלב זה תכננו אוגר הזזה שיבצע קריאת נתונים בצורה סינכרונית.
מאוד דומה לניסוי של אוגרים.

הגדרנו את האוגר הזזה באופן גנרי כדי שנוכל לשלוט על כמות הסיביות שאנחנו רוצים לקבל.

תוכנית:

```
library ieee;
use ieee.std_Logic_1164.all;

entity Shift_Register is
generic(Bits: integer := 8);
port(
EnOut, EnShift, rst, clk, d: in std_logic;
Q: out std_logic_vector(Bits-1 downto 0));
end;

architecture one of Shift_Register is
signal sEnOut, sQ: std_logic_vector (Bits-1 downto 0);
begin
process(rst,clk)
begin
if (rst = '1') then sQ <= (others => '0');
elsif (clk 'event and clk = '1') then
      if (EnShift = '1') then
            sQ <= d & sQ(Bits-1 downto 1);
      end if;
end if;
end process;
sEnOut <= (others => EnOut);
Q <= sQ and sEnOut;
end;
```

<u>סכימת RTL:</u>



בשלב זה היה עלינו לתכנן יחידה המזהה את סיבית הזוגיות.

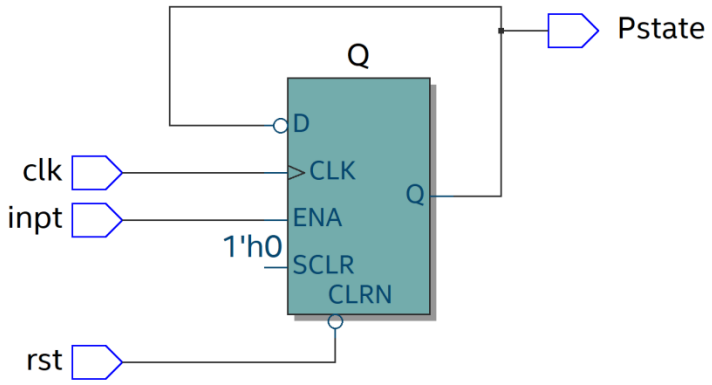
כדי לחשב את סיבית הזוגיות בזמן אמת תכננו מערכת המקבלת את סיבית המידע בכל דגימה וחישוב הזוגיות שלה עם הסיבית הקודמת שנדגמה.

העיקרון מבוסס על דלגלג מסוג TFF אשר ממומש באמצעות דלגלג מסוג DFF ויש לו משוב מהמוצא לכניסה דרך שער XOR (כאשר המשוב אחראי על חישוב הזוגיות).

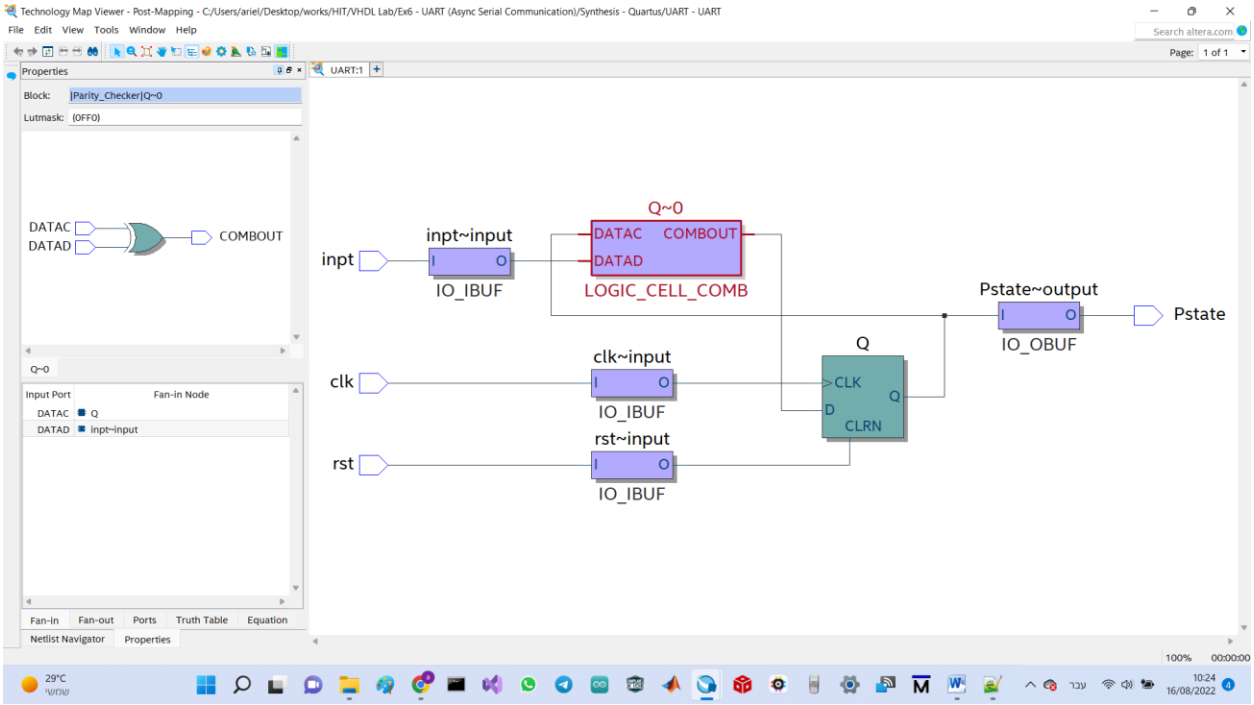
<u>תוכנית:</u>

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

entity Parity_Checker is
port(
inpt,clk,rst: in std_logic;
Pstate: out std_logic);
end;

architecture one of Parity_Checker is
signal d,Q: std_logic;
begin
process(clk)
begin
if (rst = '0') then Q<='0';
elsif (clk 'event and clk = '1') then Q<=d; end if;
end process;
d <= Q xor inpt;
Pstate <= Q;
end;
```

<u>סכימת RTL:</u>



כפי שניתן לראות השרטוט לא הכי ברור ולכן נסתכל על הסכימה שאנו מקבלים בסינתזה המלאה דרך Technology Map Viewer.



ניתן לראות כאן בבירור שתוצאת הסינתזה נותנת לנו שער XOR במשוב של הדלגלג DFF.

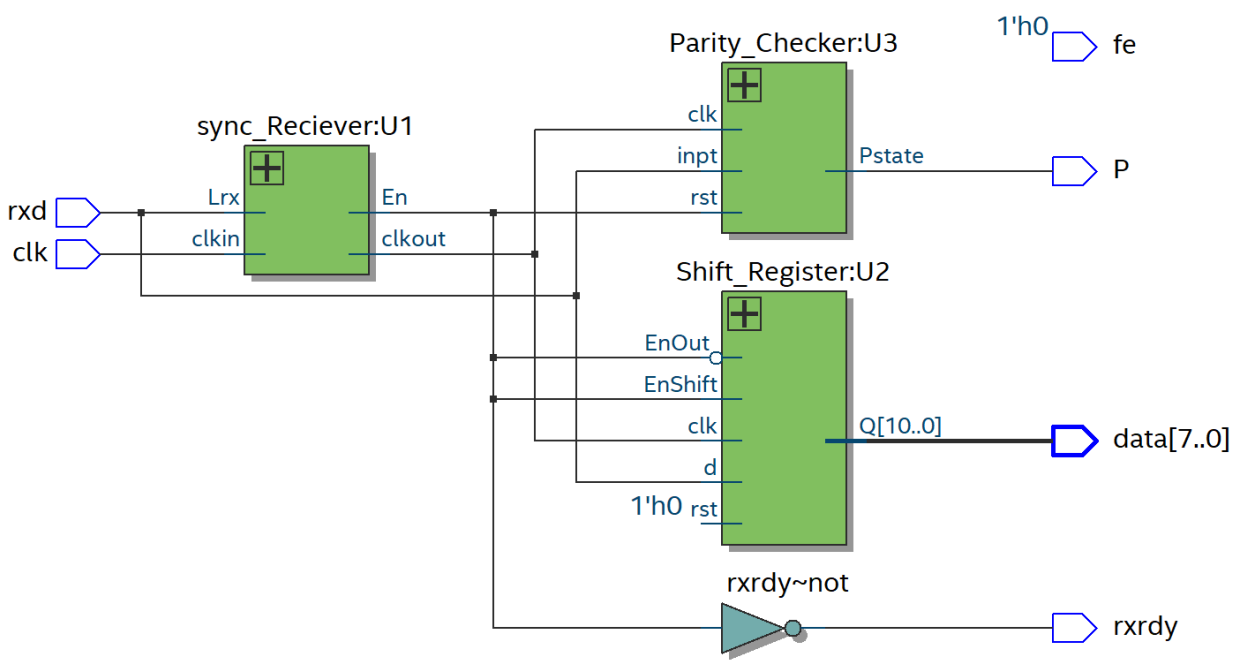
כעת אפשר עם כל יחידות אלה להשלים את התכנון של המקלט.

<u>תוכנית:</u>

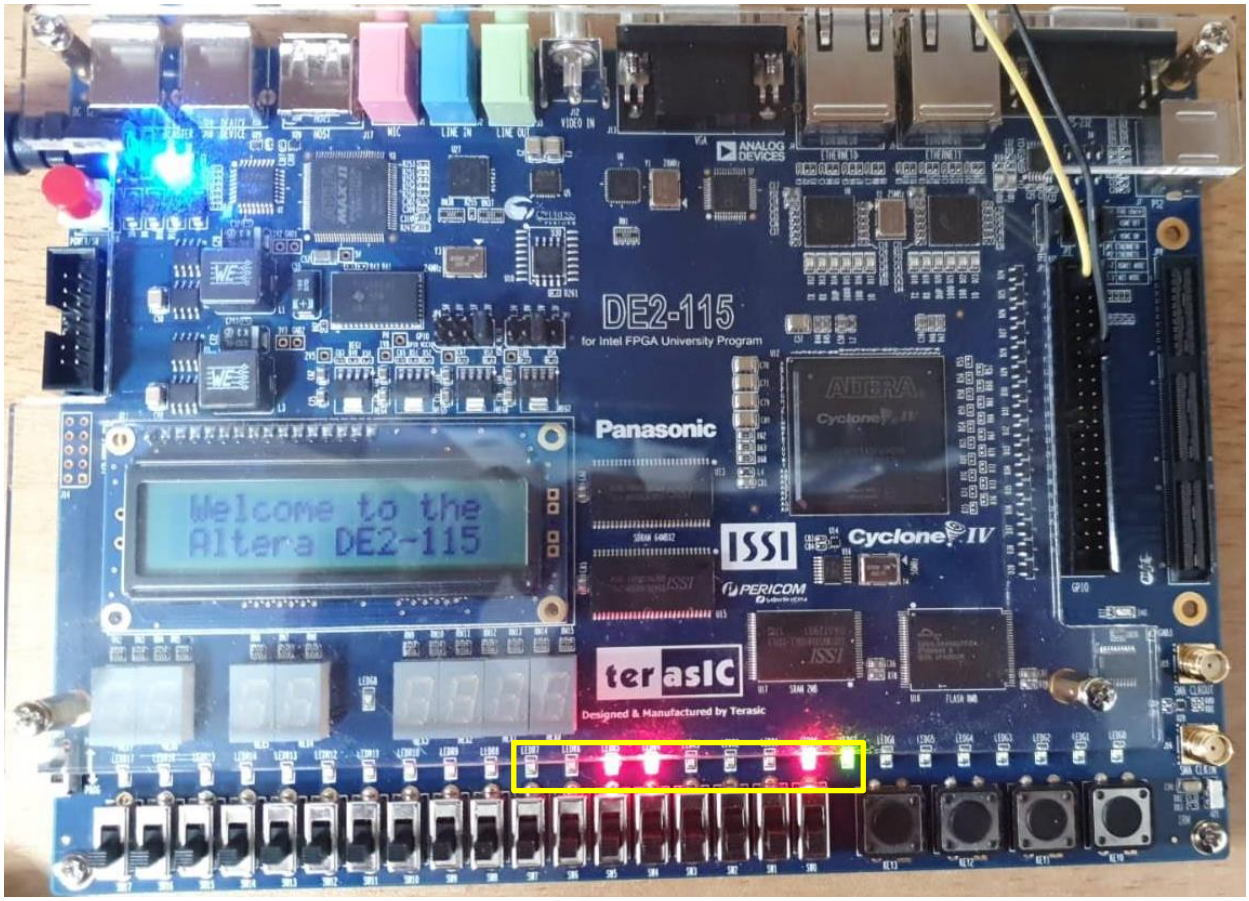
```
library ieee;
use ieee.std_logic_1164.all;

entity Reciever is
port(
rxd,clk: in std_logic;
data: out std_logic_vector(7 downto 0);
rxrdy,P,fe: out std_logic);
end;

architecture one of Reciever is

component sync_Reciever
port(
clkin,Lrx: in      std_logic;
En,clkout: buffer std_logic);
end component;

component Shift_Register
generic(Bits: integer := 11);
port(
EnOut, EnShift, rst, clk, d: in std_logic;
Q: out std_logic_vector(Bits-1 downto 0));
end component;
```

```
component Parity_Checker
port(
inpt,clk,rst: in std_logic;
Pstate: out std_logic
);
end component;

signal internclk: std_logic;
signal Parallel:  std_Logic_vector(10 downto 0);
signal Enflag:    std_logic;


begin
U1: sync_Reciever  port map (clk,rxd,Enflag,internclk);
U2: Shift_Register generic map (11) port map (not Enflag, Enflag,
'0',internclk,rxd,Parallel);
U3: Parity_Checker port map (rxd, internclk, Enflag, P);
data <= Parallel(8 downto 1);
rxrdy <= not Enflag;
end;
```

סכימת RTL:



לאחר הצריבה שלחנו את הקוד ASCII של המספר 1 לכרטיס והתקבל התוצאה הבאה:



ניתן לראות בתוצאת המערכת (הלדים האדומים) שהתקבל המידע 0x31 שהוא בדיוק הקוד ASCII של המספר 1.