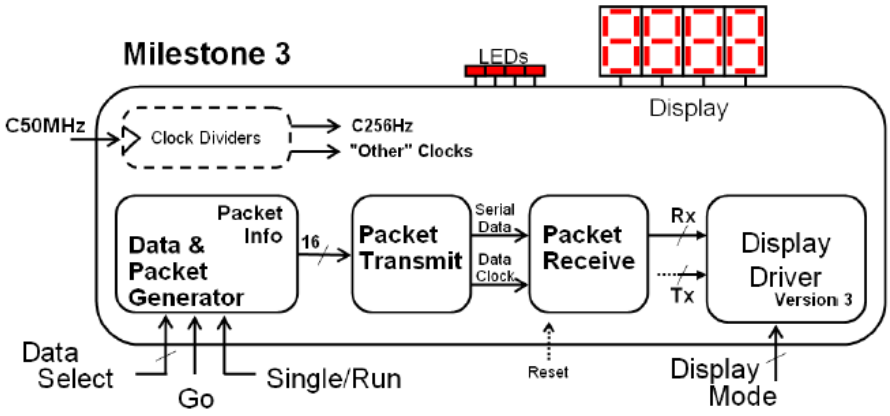
Milestone3

21/2/2017

Serial Packet Tx & Rx

Main requirement:

The milestone 3 is design of the packet system is to develop the serial packet transmit and packet receive subsystems.



Milestone 3 overview

Subtask 1: improvement on data & packet generator

Initial idea:

Since the Milestone 2 is finished in a hurry, there are some signal or process can be removed or improved.

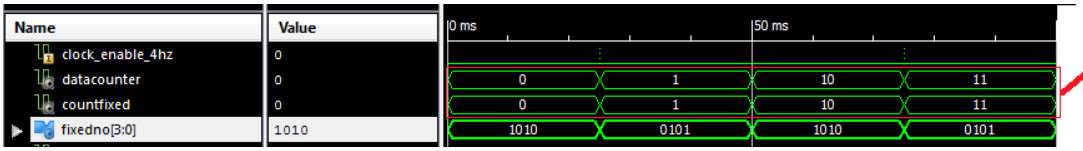
1. Signal countfixed is removed

The countfixed have the same trigger condition with datacounter, which mean they are identical. As a result, every condition based on countfixed is replaced by datacounter. The process to change countfixed is removed.

The countfixed process used in Milestone 2.

```
Fixed_Value_P1: process (Clock_Enable_4Hz, Reset, Clock, oldswitch)
begin
if Reset = '1' or oldswitch /= "00" then
   countfixed <= 0;  -- initialize counter to 0
elsif rising_edge (Clock) then
   if Clock_Enable_4HZ = '1'  and start = '1' then -- increase counter at a rate of 4Hz, therefore every 0.25 seconds
      if countfixed< 3 then   -- if below 3
         countfixed<=countfixed+1;  -- increase counter by 1
      else  -- roll over
         countfixed<=0; -- if at 3 it goes back to 0
      end if;
   end if;
end if;
end process;

Fixed_Value_P2: process (countfixed)   -- triggered by countfixed therefore every 0.25 seconds
begin
   if countfixed=0    then fixedno<="1010";  -- output will be A hex
   elsif countfixed=1 then fixedno<="0101";  -- output will be 5 hex
   elsif countfixed=2 then fixedno<="1010";  -- output will be A hex
   else                    fixedno<="0101";  -- else counter 3 --> output will be 5 hex
   end if;
end process;
```



The countfixed and datacounter have identical number

The changed process is in Milestone3.

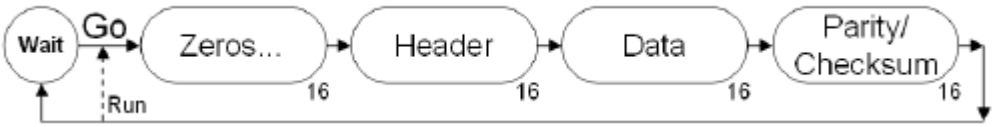
```
Fixed_Value: process (datacounter)  -- triggered by datacounter therefore every 0.25 seconds
begin
   if datacounter =0    then  fixedno<="1010";  -- output will be A hex
   elsif datacounter=1 then   fixedno<="0101";  -- output will be 5 hex
   elsif datacounter=2 then   fixedno<="1010";  -- output will be A hex
   else                       fixedno<="0101";  -- else counter 3 --> output will be 5 hex
   end if;
end process;
```

The countfixed is replaced by the datacounter

Impact: No impact to the data & packet generator, all output will remain the same

2. Format of packet

In Milestone 2, the packet follow the format below with 16 zero in the beginning of the packet.



In Milestone 3, the packet follow the format below with 8 zero in the beginning of the packet and 8 zero at the end of the packet.

Impact: the packet have to be rearranged to the required pattern. Because of this change, the display driver have to be changed to display the correct data which will be mentioned again in the display driver part.

3. Method and timing of changing the packet is changed.

Previously, we misunderstand that every four 4 bit data generated have to be stored into a 16 bit array before put it into the packet. This lead to the packet have to wait for extra 0.25s more to send it to the display.

In Milestone 3 Dataoutput which is the 16 bit array is removed from the datapacket generator. The packet is then updated when a new 4 bit data is generated. Since all data is stored into the packet immediately, no delay will be caused.

The following is the module in Milestone2 data & packet generator:

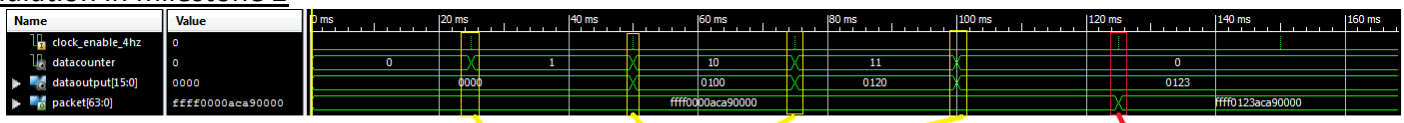
```
process (Reset, Clock)
begin
if Reset = '1' then
   dataoutput <= "0000000000000000";
elsif rising_edge (Clock) then
      if Clock_Enable_4HZ = '1' and start = '1' then
         case datacounter is
            when 0 => dataoutput(15 downto 12) <= BITоutput;
            when 1 => dataoutput(11 downto 8) <= BITоutput;
            when 2 => dataoutput(7 downto 4) <= BITоutput;
            when 3 => dataoutput(3 downto 0) <= BITоutput;
            when others => null;
         end case;
      end if;
end if;
end process;
```

The data is stored into the 16bit data (dataoutput) everytime the new 4 bit data is generated

```
Packet_Generator: process (Reset, Clock)
begin
if Reset = '1' then
   Packet(15 downto 0)  <= "0000000000000000"; -- leading 16 bit zeros of the data packet for Milestone 2
   Packet(31 downto 16) <= "1010110010101001"; -- header is always 'ACA5' hex
   Packet(47 downto 32) <= "0000000000000000"; -- this is the data, initialized to zero
   Packet(63 downto 48) <= "1111111111111111"; -- this contains the parity and checksum which is always 'FFFF' hex for Milestone 2
elsif rising_edge (Clock) then
   if Clock_Enable_4HZ = '1' then
      if datacounter = 0 then
         Packet(47 downto 32) <=  dataoutput;
      end if;
   end if;
end if;
end process;
```

This situation means all the data is updated into the dataoutput and will send to the packet before it get affected by the next newly generated data

Simulation in milestone 2



The generated data is stored into the 16 bit data (dataoutput) but not the packet every 4Hz

the data have to store into packet after the 16 bit data is updated

Packet in Milestone 3 data & packet generator:

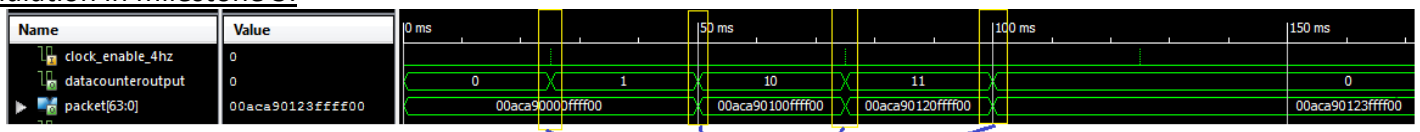
```
Packet_Generator: process (Reset, Clock)
begin
if Reset = '1' then

   Packet(7 downto 0)  <= "00000000"; -- leading 8 bit zeros of the data packet for Milestone 3
   Packet(23 downto 8) <= "1111111111111111"; --hex this contains the parity and checksum which is always 'FFFF' hex for Milestone 3
   Packet(39 downto 24) <= "0000000000000000"; -- this is the data, initialized to zero
   Packet(55 downto 40) <= "1010110010101001"; --  header is always 'ACA9'
   Packet(63 downto 56) <= "00000000";
elsif rising_edge (Clock) then
      if Clock_Enable_4HZ = '1' and start = '1' then
         case datacounter is
            when 0 => Packet(39 downto 36) <= BITоutput;
            when 1 => Packet(35 downto 32) <= BITоutput;
            when 2 => Packet(31 downto 28) <= BITоutput;
            when 3 => Packet(27 downto 24) <= BITоutput;
            when others => null;
         end case;
      end if;
end if;
end process;
```

start is added here.It is because datacounter will always stay at 0 even no data is generating.This make the packet will only update when new data is gnerated

This case will store the 4 bit data to the packet every 4Hz ,right after it is generated

Simulation in milestone 3:



when counter increases, corresponding data, is stored in the packet every 4 Hz

Subtask 2: packet transmit

Requirement:

- Before "Go" is pressed, or during any start-up delay, then the serial data line will constantly clock zeros. The clock has a rate that enables the 64 bit packet to be sent in one second.
- When "Go" is pressed there will need to be a delay to enable the data generator to complete the generation of a complete 16-bit data word before it is sent.
- A string of 8 zeros are first sent as serial data to make sure the receiver is looking for the header word that then follows. [note: the change from milestone 2 which had 16 initial zeros]
- Once the initial block of 8 zeros had been completed, it is immediately followed by the header, data and parity/checksum words sent most significant bit first.
- Finally, an additional block of 8 zeros is then sent to ensure that there is a suitable gap between packets.
- If the data generator is in single mode - then the transmit block returns to wait mode; else, when the generator is in continuous run mode - the next packet is sent.

Initial idea: Packet transmit will breakdown the packet into 64 bit. Then, send it to the packet receive bit by bit every 64 Hz. When no data is generated, the data transmitted to packet receive will always be zero.

The following is the final module of packet transmit.

Port used in this entity:

```
PORT(
        Reset : IN std_logic;
        Clock : IN std_logic;
        Clock_Enable_4Hz  : in std_logic;
        Clock_Enable_64Hz  : in std_logic;
        onebitoutput: out std_logic;
        packet:in std_logic_vector (63 downto 0);
        datacounteroutput: in integer;
        serialflag: out std_logic
    );
```

the one bit serial ouput data to packet recieve

packet coiming in from data packet generator

datacounter from packet generator to identify which data is generating

serial flag turn on to identify the data generated is sending out bit by bit

Signal used in this entity:

```
signal packet_transmit: std_logic_vector (63 downto 0);
signal counter64 : integer;
signal flagtransmit: std_logic;
```

signal used to store the whole packet after a full 16 bit data is generated

a counter count from 63 down to 0,which show the bit need to be sent out from the packet
Set as integer will have benefit of easier access to the specific bit to bit in the packet.

The flag which turn on when the packet is transmiting the data bit by bit

Process for flagtransmit:

This process introduce a flag to identify the time that start and end of transmitting the packet bit by bit.

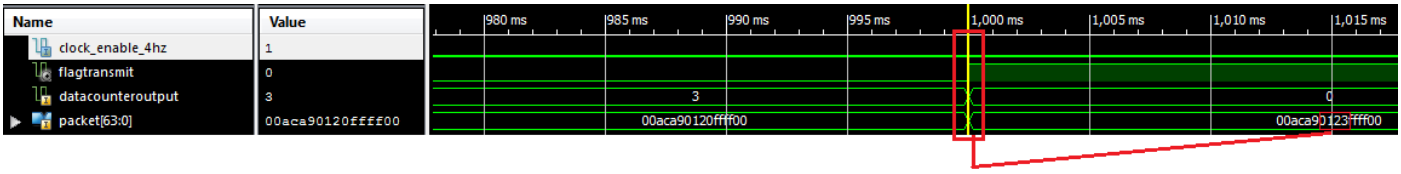
```
-- this process detects when the packet is completed then the flag for turning on the transmitter is made high.
-- the flag will go low when all the data from the packet has been transmitted to the receiver bit by bit
process (Reset, Clock_Enable_4HZ, Clock, datacounteroutput, counter64)
begin
if Reset = '1' then
   flagtransmit <= '0';
elsif rising_edge (Clock) then
   if datacounteroutput = 3 and Clock_Enable_4HZ = '1' then
      flagtransmit <= '1';
   elsif counter64 =0 and Clock_Enable_64HZ = '1' then
      flagtransmit <= '0';
   end if;
end if;
end process;
serialflag <= flagtransmit;
```

This situation mean the whole packet is generated and ready to be sent

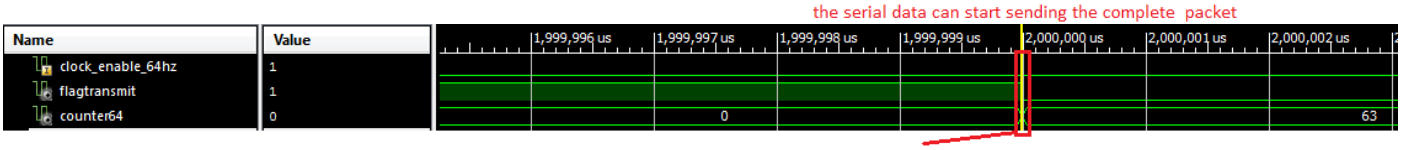
this situation mean the whole packet is already sent to transmit bit by bit.

This output the serial flag which mean the moment that sending the packet

Simulation:



At this point ,data counter is 3 which mean the last 4bit data of the 16 bit data is ready.The flagtransmit will set to 1, to identify the serial data can start sending the complete packet



At this point, counter =0 which mean all 64 bit in the packet is sent, the flagtransmit will then set to 0 beacuse of that

Problem occurred:

For the statement that bring the transmit flag back to 0(green box), the clock enable is intentionally set to 64 Hz instead of 4 Hz. Although the 64 Hz clock enable and the 4 Hz clock enable is supposed to be synchronize, when implementing it on the board without pressing the reset, they are not. Using 4Hz clock enable in that situation, will lead to that statement never true and the packet transmit will never stop transmitting the packet.

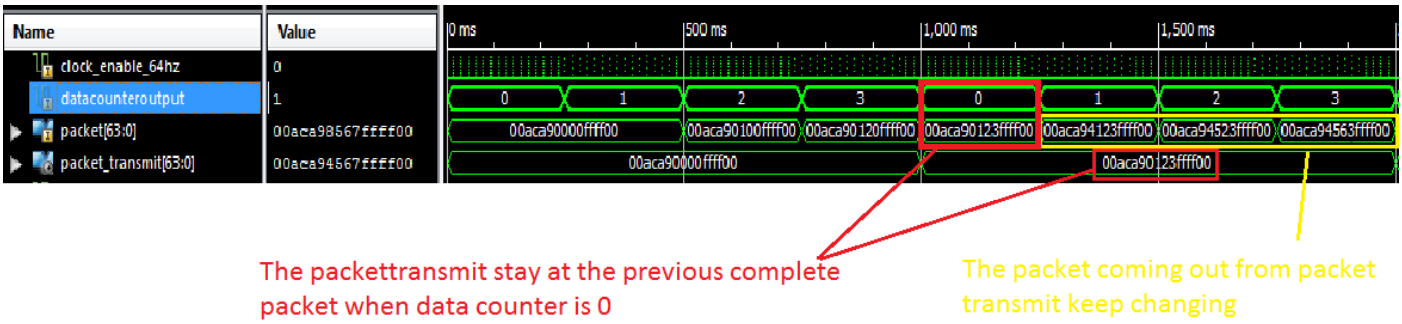
Process for packet transmit:

This process store the whole packet after all data is generated. Since the packet in our data packet generator is always changing, outputting the 1 bit data directly from packet will be incorrect. As a result, the packet have to be stored after the 16 bit data is generated. As a result, packet transmit is used to store the packet after all 16 bit data of 1 packet is generated.

```
-- the value of the packet is assigned to a variable of the same size whenever the packet is completed
-- this happens because the packet is being changed every 4Hz when data is being generated
-- therefore this condition is required
process (Reset, Clock, datacounteroutput)
begin
if Reset = '1' then
   packet_transmit <= "0000000000000000000000000000000000000000000000000000000000000000";
elsif rising_edge (Clock) then
   if datacounteroutput = 0 then -- packet is completed OR no new packet is being generated
      packet_transmit <= packet;
   end if;
end if;
end process;
```

in this situation, it mean all 16 bit data from a packet is generated

Simulation:



Process for counter64:

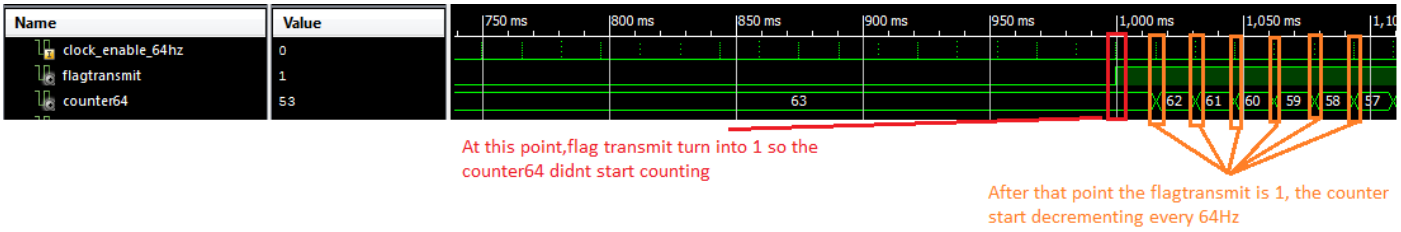
This process start counting down from 63 to 0 every 64 Hz when all the 16 bit data in a packet is ready. This counter is intended to be integer so that it can work as a pointer for the packet transmit to send out 1 bit data every 64 Hz.

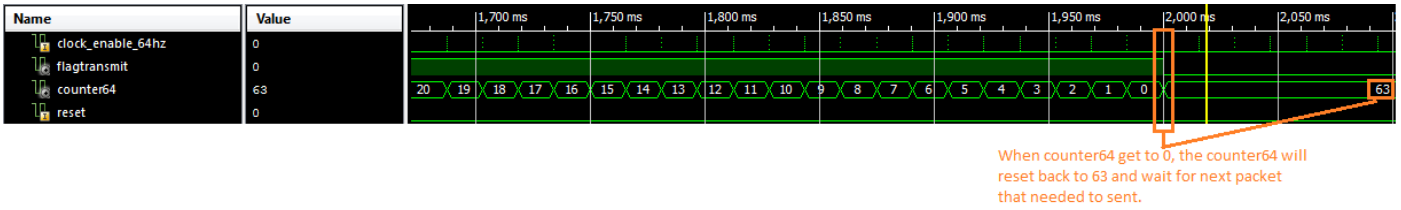
```
-- process detects when the flag for transmitting has been triggered.
-- when it has been triggered, the counter representing each bit of the packet starts to go down
-- one by one at a rate of 64Hz and it resets when it reaches value of 0
process(Reset, Clock, Clock_Enable_64Hz, flagtransmit, counter64)
begin
if Reset = '1' then
   counter64 <= 63;
elsif rising edge (Clock) then
   if Clock_Enable_64Hz = '1' and flagtransmit = '1' then
      if counter64 > 0 then
         counter64 <= counter64 - 1;
      else
         counter64 <= 63;
      end if;
   end if;
end if;
end process;
```

These statement will decrement the counter64 from 63 to 0 every 64Hz when the data is sending out bit by bit (flagtransmit = 1)

reset counter 64 to 63 to prepare for next packet to transmit

Simulation:





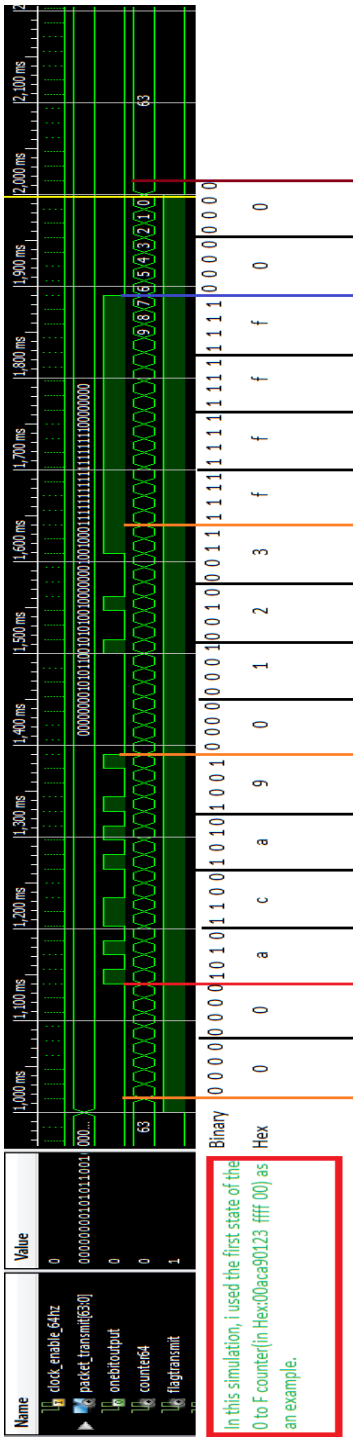
Process for onebitoutput:This process use flagtransmit as an indicator to start the transmition of the data bit by bit as onebitoutput. When flagtransmit is low, no data is sending. The onebitoutput will stay 0.

```
-- this process is sending out each bit of the packet at a rate of 64Hz when a compelted packet is received
-- when a packet is not received, it is sending out zeros at a rate of 64Hz
process(Reset, counter64, Clock, flagtransmit, Clock_Enable_64Hz)
begin
if Reset = '1' then
   onebitoutput <= '0';
elsif rising_edge (Clock) then
      if flagtransmit = '1' and Clock_Enable_64Hz = '1' then
         onebitoutput <= packet_transmit(counter64);
      elsif flagtransmit = '0' and Clock_Enable_64Hz = '1' then
         onebitoutput <= '0';
      end if;
end if;
end process;
```

This situation indicating the data should be sending out

This situation shows that the data of the packet should not be sending out,the onebitoutput will then set to be 0

Simulation:

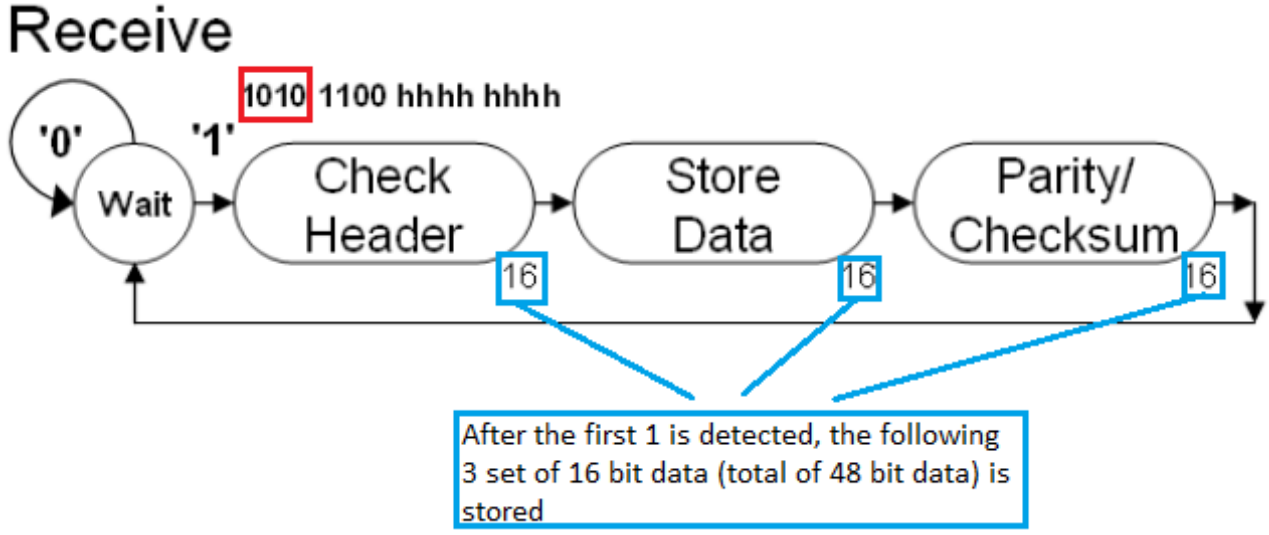


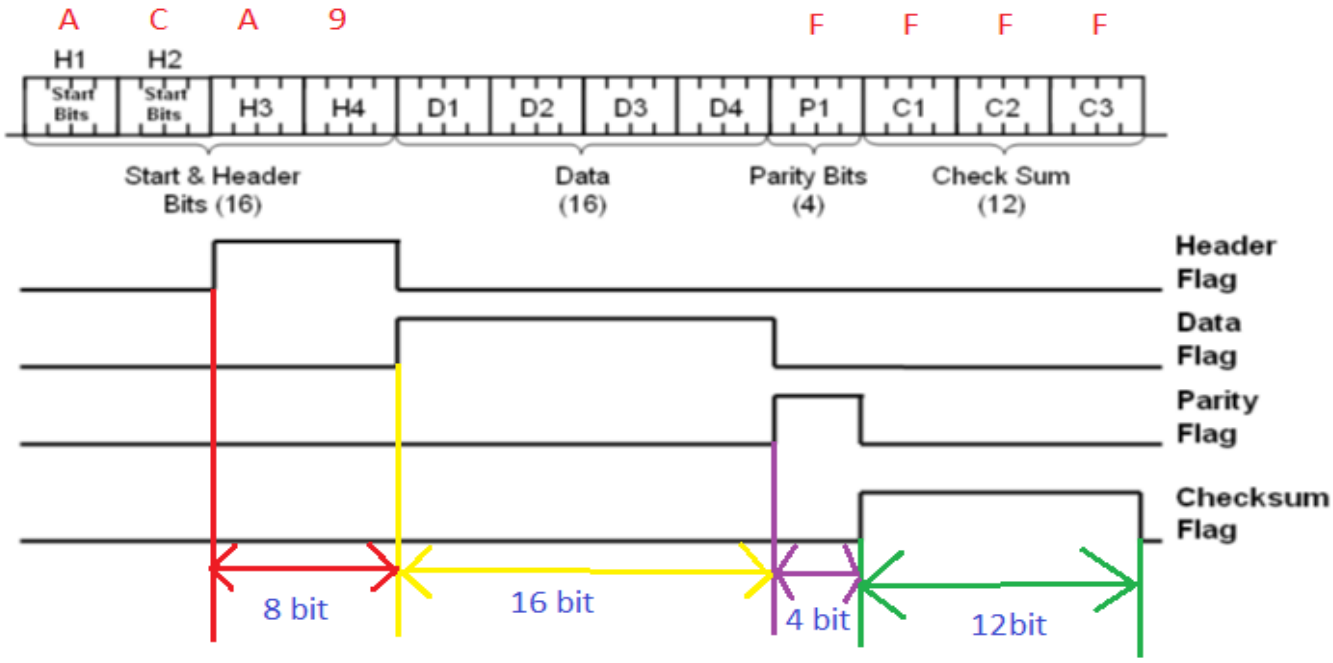
Subtask 3: packet receive

Requirement:

- The serial data line is constantly monitored on each active clock edge.
- When a leading **'0'** is found, then the system waits for the next clock edge.
- When a **'1'** is detected then it is assumed that this the first bit of the header, so this bit and the following 15 bits are stored.
- The data and the parity/checksum words that then follow are also stored
- The packet receive subsystem then again starts looking for the leading zeros / first header bit.
- As the packet is being received then various flags should be set to indicate the section being processed: The serial data line and four status flags will be assigned to specified LEDs on the Spartan 3 board.

Initial idea: Packet receive subsystem will look for the first 1 from serial data which is the first bit of the header "a" (1010). When the first 1 is found, the following 48 bit data is stored.





During different data is received, different flag have to be raised to indicate which data is storing at that moment .The time of those flag to be raised is the number of bit stored times 64Hz.

Port used in this entity:

```
PORT(    Reset : IN std_logic;
         Clock : IN std_logic;
         Clock_Enable_64Hz  : in std_logic;
         onebitoutput: in std_logic;
         receive:out STD_LOGIC_vector(47 downto 0);
         headerflag: out std_logic;
         dataflag: out std_logic;
         parityflag: out std_logic;
         checksumflag: out std_logic
         );
```

Serial data coming from packet transmit

logic vector to store the required 48 bit data in the packet

flags which indicate what data is storing at those moment.These flag will be used to turn on led in display driver

Signal used in this entity:

```
signal counter48 : integer;
signal flagreceive: std_logic;
```

a counter count from 47 down to 0, which show the bit storing from serial data input

A flag that turn on when the first 1 of the header is detected and will turn off after the following 48 bit data is recieved

Process for counter48:

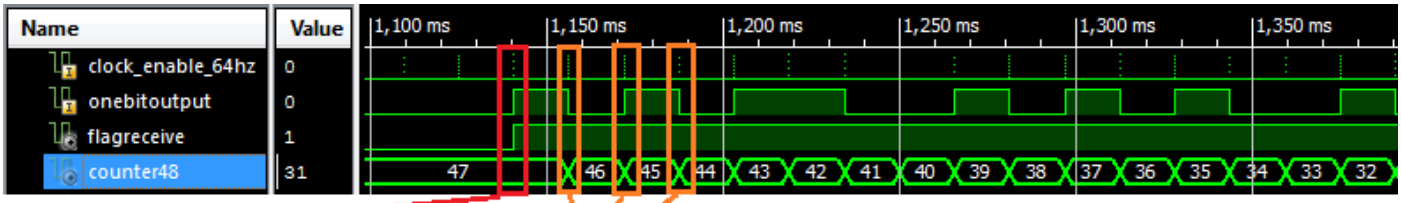
This process start counting down from 47 to 0 every 64 Hz when the packet receive detect the first 1 from the header. This counter is intended to be integer so that it can work as a pointer for the packet receive to store into the standard logic vector receive every 64 Hz.

```
-- a counter representing the header, data and checksum starts going down one by one
-- once the data starts arriving
process(Reset, Clock,Clock_Enable_64Hz, flagreceive, counter48)
begin
if Reset = '1' then
   counter48 <= 47;
elsif rising_edge (Clock) then
   if Clock_Enable_64Hz = '1' and flagreceive = '1' then
      if counter48 > 0 then
         counter48 <= counter48 - 1;
      else
         counter48 <= 47;
      end if;
   end if;
end if;
end process;
```

These statement will decrement the counter 48 from 47 to 0 every 64Hz when the 48 bit data is being received

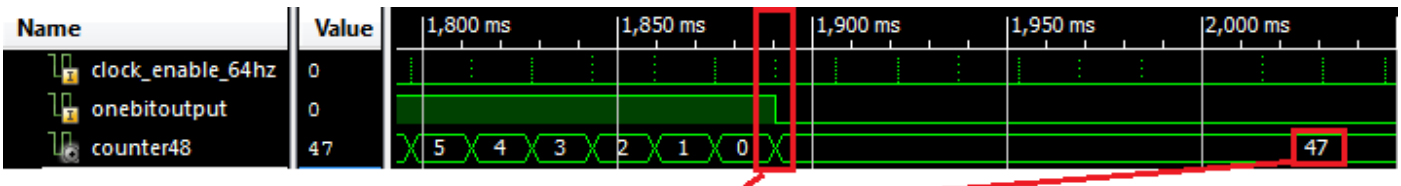
The counter will reset back to 47 after all 48 bit data is saved and wait for the next 1 from the header bit of a packet

Simulation:



When the first 1 from header is detected, flagreceive is changed to 1 .At this point , it was 0.THe counter will not start decrement

After that point, flagreceive is 1.The counter will start decreasing by 1 every 64 Hz.



When the whole 48 bit data is received(counter48=0),counter 48 is reset to 47 to wait for the next packet

Process for flagreceive:

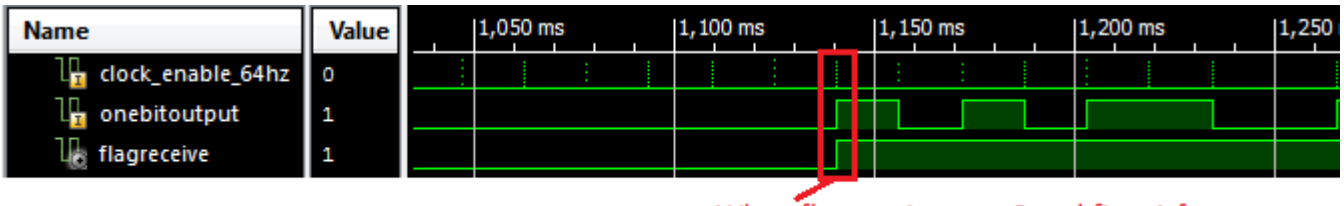
This process introduce a flag to identify the time that needed to store all 48 bit data into standard logic vector receive.

```
-- this process detects when the packet is being received then a flag goes high.
-- flag goes low once the required 48-bit data has been received
process (Reset, Clock_Enable_64HZ, Clock, onebitoutput, counter48, flagreceive)
begin
if Reset = '1' then
   flagreceive <= '0';
elsif rising_edge (Clock) then
   if onebitoutput = '1'  and flagreceive = '0' then
      flagreceive <= '1';
   elsif counter48 =0 and Clock_Enable_64Hz = '1' then -- required 48-bit data has been received
      flagreceive <= '0';
   end if;
end if;
end process;
```

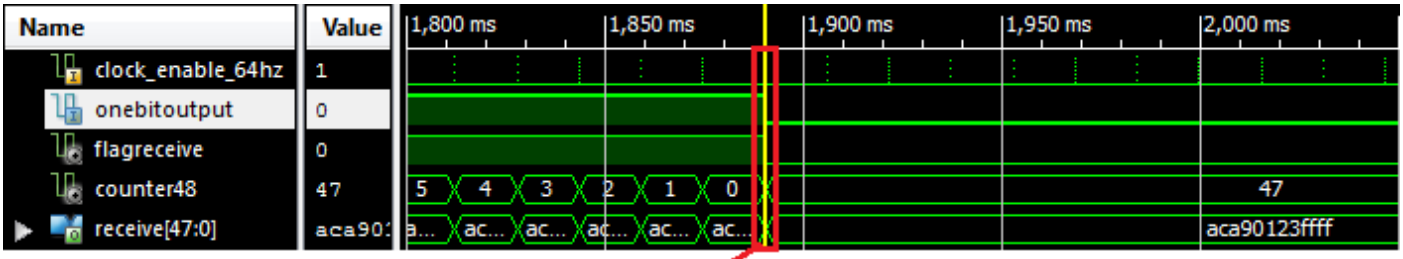
This situation turn the flag on when the first 1 from the header is detected.Flagreceive=0, means the data is not storing.The 1 after the first 1 from header will not trigger the flagreceive to change.

This situation means all 48 bit data had been stored(counter48=0).64Hz clock have to be used here .It is because without it, when the last serial data is still 1 and the flagreceive is changed to 0 becuase the counter48 is 0.It will retrigger the flagreceive to 1, and start counting for another time.

Simulation:



When flagreceive was 0 and first 1 from header is detected,flagreceive is set to 1



At this point, the last bit of the serial data is finished,all 48 bit data had been stored(counter48=0).The flagreceive will turn into 0.

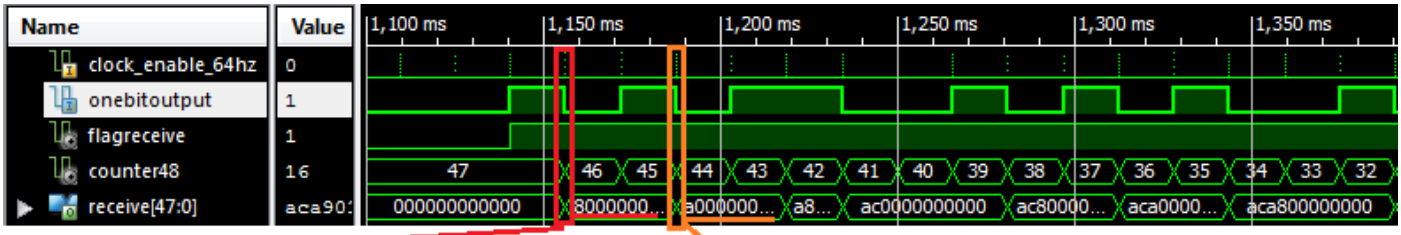
Process for receive:

This process store the serial data input to standard logic vector receive while using counter48 as pointer to store at correct location.

```
-- when the receiver starts receiving the 48-bit data, each bit is being saved
-- at a rate of 64Hz
process(Reset, counter48, Clock, Clock_Enable_64Hz, flagreceive)
begin
if Reset = '1' then
   receive(47 downto 0)<= "000000000000000000000000000000000000000000000000";
elsif rising_edge (Clock) then
   if flagreceive ='1' and Clock_Enable_64Hz = '1' then
      receive(counter48) <= onebitoutput;
   end if;
end if;
end process;
```

This statement store the incoming data into corresponding bit into the standard logic vector receive.

Simulation:



At this point, the first number is stored into the recieve(47).The first 4 number of recieve is 1000, so it show 8 in Hex.

At this point,The second 1(third number) is stored into the receive(45).The first 4 number of recieve is 1010,so it show a in Hex.

Process for flags:

This process turn on different flag for led when specific data is stored into standard logic vector receive.

```
-- a process sets diffirent flags to high when it is storing the representing data
process(Reset, counter48, Clock)
begin
if Reset = '1' then
   headerflag   <= '0';
   dataflag     <= '0';
   parityflag   <= '0';
   checksumflag <= '0';
elsif rising_edge (Clock) then
   if counter48 <= 39 and counter48 >= 32 then
      headerflag <= '1';
   else headerflag <= '0';
   end if;

   if counter48 <= 31 and counter48 >= 16 then
      dataflag <= '1';
   else dataflag <= '0';
   end if;

   if counter48 <= 15 and counter48 >= 12 then
      parityflag <= '1';
   else parityflag <= '0';
   end if;

   if counter48 <= 11 and counter48 >= 0 then
      checksumflag <= '1';
   else checksumflag <= '0';
   end if;

end if;
end process;
```

These statement turn the headerflag on when we are storing the last 8 bits of the header

These statement turn the dataflag on when we are storing the 16 bit data

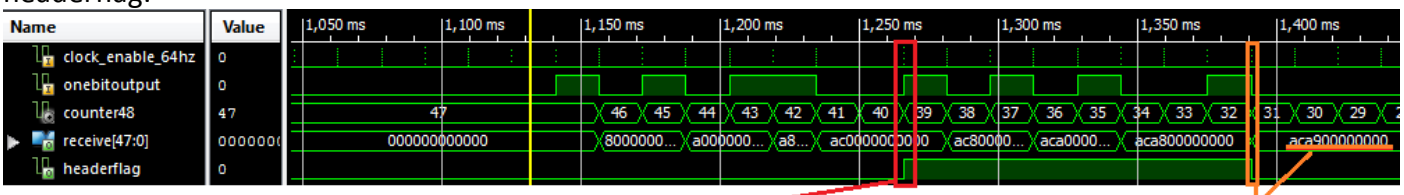
These statement turn the parityflag on when we are storing the 4 bit parity

These statement turn the checksumflag on when we are storing the 12 bit parity

Simulation:

For the first set data of the 0 to F counter (0x00aca90123ffff00)

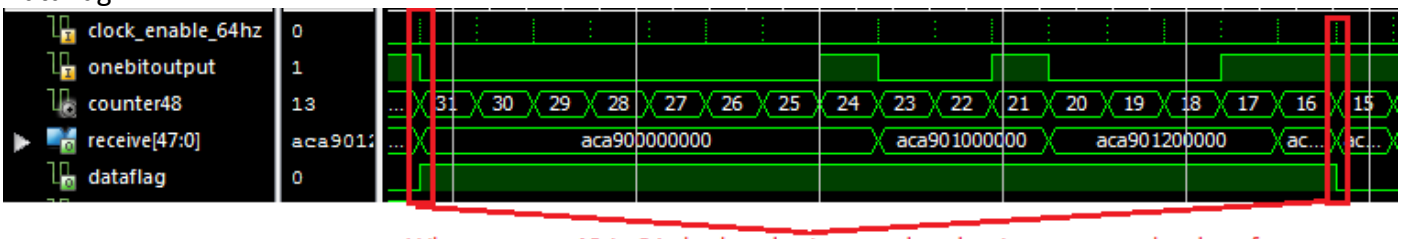
headerflag:



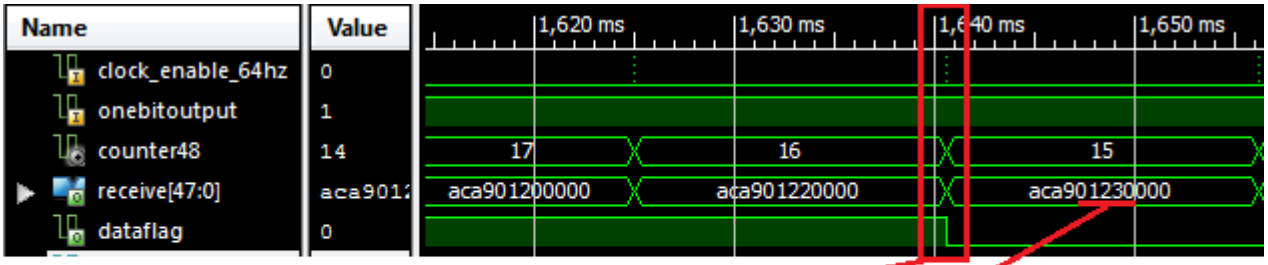
At this point, the first 8 bit of the header had been stored (first 2 hex number is ac),it will start storing the last 8 bit of the header.As a result, headerflag is turned on

At this point, the last 8 bit of the data is stored (first 4 hex number is aca9).THe header flag will turn off.

Dataflag:

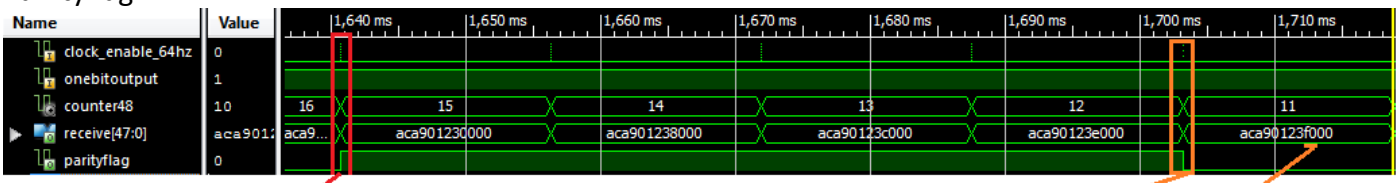


When counter48 is 31 the header is stored and going to store the data for the next 16 bit.As aresult, the dataflag is turned on when counter 48 is 31 to 16.



At this point, the 16 bit data(0123 in hex) is stored.The dataflag will be set back to 0.

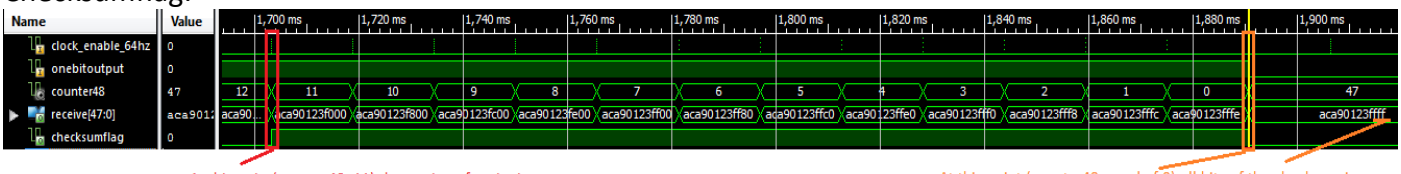
Parityflag:



At this point(counter =15),the pairity will start storing in the receive.The parityflag will be turned on.

At this point(the end of counter =12), all bit of pairity is stored.The pairity flag will turn off.

Checksumflag:



At this point(counter48=11),the storing of parity is finished.It will start the storing of checksum.AS a result the checksumflag is turned on

At this point (counter48 = end of 0),all bits of the checksum is stored.The checksumflag is turned off

Subtask 4: Display Driver

Requirement:

- The display driver will require updating to be able to display the header, data and parity/checksum for both the transmitted packet and the received packet.
- Because the data is being sent
  one bit at a time as a serial signal, then there will be a time delay between the information being transmitted and received.

Display_Mode	Action	
000	Data Generator M2	[D1..D4]
001	Headers Tx	[H1..H4]
010	Data Tx	[D1..D4]
011	Parity/Checksum Tx	[P1,C1..C3]
100	reserved for later milestone	
101	Headers Rx	[H1..H4]
110	Data Rx	[D1..D4]
111	Parity/Checksum Rx	[P1,C1..C3]

Initial idea: Display driver that was done in Milestone 2 will stay the same. Only add the new display mode into the generation which will display the data stored in the packet receive and the Led light for flags which show what data is storing in that packet receive entity.

Process for display mode:

The first thing in this process that need to be changed is display mode 010.The reason behind is the packet from data generator is always changing .In display mode 010 which the data need to be updated every 1 second after all 16 bit data is generated. With changing packet, it will keep changing the display just like display mode 000.As a result, the condition of counter =0 is added in that situation to make sure it only display the data after all 16 bit data is generated. This will have no impact to the actual display on the board.

```
if Display_mode="010" and counter ="00" then
   D4(3 downto 0)       <= Packet(39 downto 36); D4(4)       <= '0';
   D3(3 downto 0)       <= Packet(35 downto 32); D3(4)       <= '0';
   D2(3 downto 0)       <= Packet(31 downto 28); D2(4)       <= '0';
   D1(3 downto 0)       <= Packet(27 downto 24); D1(4)       <= '0';
end if;
```

The second thing is the new display mode added into display driver to display the data stored in the packet receive.

```
if Display_mode="101"then

   D4(3 downto 0)       <= receive(47 downto 44); D4(4)       <= '0';
   D3(3 downto 0)       <= receive(43 downto 40); D3(4)       <= '0';
   D2(3 downto 0)       <= receive(39 downto 36); D2(4)       <= '0';
   D1(3 downto 0)       <= receive(35 downto 32); D1(4)       <= '0';
end if;
if Display_mode="110"then

   D4(3 downto 0)       <= receive(31 downto 28); D4(4)       <= '0';
   D3(3 downto 0)       <= receive(27 downto 24); D3(4)       <= '0';
   D2(3 downto 0)       <= receive(23 downto 20); D2(4)       <= '0';
   D1(3 downto 0)       <= receive(19 downto 16); D1(4)       <= '0';
end if;
if Display_mode="111"then

   D4(3 downto 0)       <= receive(15 downto 12); D4(4)       <= '0';
   D3(3 downto 0)       <= receive(11 downto 8);  D3(4)       <= '0';
   D2(3 downto 0)       <= receive(7 downto 4);   D2(4)       <= '0';
   D1(3 downto 0)       <= receive(3 downto 0);   D1(4)       <= '0';
end if;
```

In this mode,it will show the stored header

In this mode, it will show the stored 16 bit data

In this mode, it will show the stored pairity and checksum

Process for led lights:

This process will turn on the led representing what data is actually storing in that moment. This will be following the flag coming out from the packet receive.

```
-- a process that turn on the led when it is storing the representing data
process (Reset, Clock, start_output, headerflag, dataflag, parityflag, checksumflag, serialflag)
begin
if Reset = '1' then
   led(7 downto 0) <= "00000000";
elsif rising_edge (Clock) then
   if serialflag = '1' then
      led(4) <= '1';
   else
      led(4) <= '0';
   end if;
   if headerflag = '1' then
      led(3) <= '1';
   else
      led(3) <= '0';
   end if;
   if dataflag = '1' then
      led(2) <= '1';
   else
      led(2) <= '0';
   end if;
   if parityflag = '1' then
      led(1) <= '1';
   else
      led(1) <= '0';
   end if;
   if checksumflag = '1' then
      led(0) <= '1';
   else
      led(0) <= '0';
   end if;
end if;
end process;
```

All the requireed led will be turned on with corresponding flag.

The corresponding flag to led are

serialflag => led(4)

headerflag =>led(3)

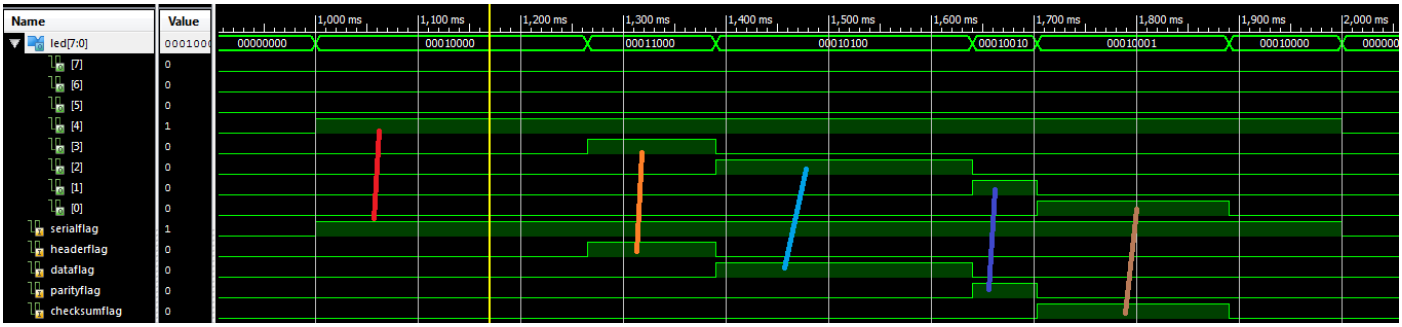
dataflag =>led(2)

parityflag =>led(1)

checksumflag =>led(0)

Simulation for leds:

All the led will turn on and off with the exact same time with the corresponding flag.



Pin Setting (ucf file):

```
 3  NET Reset                          LOC = M13;
 4  NET Clock                          LOC = T9;
 5  NET SingleRun                      LOC = L13;
 6  NET Go_Button                      LOC = L14;
 7  NET Switch<1>                      LOC = K13;
 8  NET Switch<0>                      LOC = K14;
 9  NET Segment_Cathodes<6>            LOC = E14;
10  NET Segment_Cathodes<5>            LOC = G13;
11  NET Segment_Cathodes<4>            LOC = N15;
12  NET Segment_Cathodes<3>            LOC = P15;
13  NET Segment_Cathodes<2>            LOC = R16;
14  NET Segment_Cathodes<1>            LOC = F13;
15  NET Segment_Cathodes<0>            LOC = N16;
16  NET Anode_Enable<3>                Loc = E13;
17  NET Anode_Enable<2>                Loc = F14;
18  NET Anode_Enable<1>                Loc = G14;
19  NET Anode_Enable<0>                Loc = D14;
20  NET Display_mode<0>                Loc = F12;
21  NET Display_mode<1>                Loc = G12;
22  NET Display_mode<2>                Loc = H14;
23  NET led<7>                         Loc = P11;
24  NET led<6>                         Loc = P12;
25  NET led<5>                         Loc = N12;
26  NET led<4>                         Loc = P13;
27  NET led<3>                         Loc = N14;
28  NET led<2>                         Loc = L12;
29  NET led<1>                         Loc = P14;
30  NET led<0>                         Loc = K12;
```

Added the led into the ucf file. In order to make Milestone 4 easier, all led is added into the ucf file, instead of just the led needed in the milestone 3.

Top entity:

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_arith.ALL;

entity T22_Milestone_3_Top_Entity is
    Port ( Reset : in STD_LOGIC;
           Switch: in std_logic_vector (1 downto 0);
           Clock : in  STD_LOGIC;
           SingleRun : in  STD_LOGIC;
           Go_Button: in  STD_LOGIC;
           Fast: in  STD_LOGIC;
           Anode_Enable : out STD_LOGIC_VECTOR (3 downto 0);
           Segment_Cathodes: out STD_LOGIC_VECTOR (6 downto 0);
           Display_mode: in std_logic_vector (2 downto 0);
           led:out STD_LOGIC_VECTOR (7 downto 0)
           );
end T22_Milestone_3_Top_Entity;


architecture Behavioral of T22_Milestone_3_Top_Entity is

signal Clock_Enable_4Hz: std_logic;
signal Clock_Enable_256Hz: std_logic;
signal Clock_Enable_64Hz: std_logic;
signal Digit : STD_LOGIC_VECTOR (3 downto 0);
signal Packet :  std_logic_vector (63 downto 0);
signal start_output :  STD_LOGIC;
signal fourbitoutput: STD_LOGIC_VECTOR(3 downto 0);
signal onebitoutput: STD_LOGIC;
signal receive:STD_LOGIC_vector(47 downto 0);
signal headerflag: std_logic;
signal dataflag: std_logic;
signal parityflag: std_logic;
signal checksumflag: std_logic;
signal datacounteroutput: integer;
signal serialflag: std_logic;

COMPONENT T22_Milestone_3_Clock_Dividers is
   PORT(
         Reset : IN std_logic;
         Clock : IN std_logic;
         Fast: in STD_LOGIC;
         Clock_Enable_4Hz  : OUT std_logic;
         Clock_Enable_256Hz  : OUT std_logic;
         Clock_Enable_64Hz  : OUT std_logic
       );
END COMPONENT;
```

```
COMPONENT T22_Milestone_3_Data_Packet_Generator is
   PORT( Reset : in  STD_LOGIC;
         Clock : IN std_logic;
         Go_Button: in  STD_LOGIC;
         SingleRun : in  STD_LOGIC;
         start_output : out STD_LOGIC;
         Clock_Enable_256HZ : IN std_logic;
         Clock_Enable_4HZ : IN std_logic;
         fourbitoutput:out  STD_LOGIC_VECTOR(3 downto 0);
         Packet : OUT std_logic_vector (63 downto 0);
         Switch : in std_logic_vector (1 downto 0);
         datacounteroutput:OUT integer
       );
END COMPONENT;

COMPONENT T22_Milestone_3_Packet_Transmit is
   PORT(
         Reset : IN std_logic;
         Clock : IN std_logic;
         Clock_Enable_4Hz  : in std_logic;
         Clock_Enable_64Hz  : in std_logic;
         onebitoutput:out STD_LOGIC;
         packet:in std_logic_vector (63 downto 0);
         datacounteroutput:in integer;
         serialflag: out std_logic
       );
END COMPONENT;

COMPONENT T22_Milestone_3_Packet_receive is
   PORT( Reset : IN std_logic;
         Clock : IN std_logic;
         Clock_Enable_64Hz  : in std_logic;
         onebitoutput:in STD_LOGIC;
         receive:out STD_LOGIC_vector(47 downto 0);
         headerflag: out std_logic;
         dataflag: out std_logic;
         parityflag: out std_logic;
         checksumflag: out std_logic
       );
END COMPONENT;
```

```
COMPONENT T22_Milestone_3_Display is
   PORT( Reset : in  STD_LOGIC;
         Clock : IN std_logic;
         Clock_Enable_256HZ : IN std_logic;
         Clock_Enable_4HZ : IN std_logic;
         Packet : in STD_LOGIC_VECTOR (63 downto 0);
         Display_mode:in std_logic_vector (2 downto 0);
         Anode_Enable : out STD_LOGIC_VECTOR (3 downto 0);
         fourbitoutput:in  STD_LOGIC_VECTOR(3 downto 0);
         start_output : in STD_LOGIC;
         Segment_Cathodes: out STD_LOGIC_VECTOR (6 downto 0);
         led:out STD_LOGIC_VECTOR (7 downto 0);
         receive:in STD_LOGIC_VECTOR (47 downto 0);
         headerflag: in std_logic;
         dataflag: in std_logic;
         parityflag: in std_logic;
         checksumflag: in std_logic;
         serialflag: in std_logic
       );
END COMPONENT;

begin
C1: T22_Milestone_3_Clock_Dividers PORT MAP( Reset => Reset,
                                             Clock => Clock,
                                             Fast => Fast,
                                             Clock_Enable_64Hz => Clock_Enable_64Hz,
                                             Clock_Enable_4Hz => Clock_Enable_4Hz,
                                             Clock_Enable_256Hz => Clock_Enable_256Hz
                                            );

C2: T22_Milestone_3_Data_Packet_Generator PORT MAP( Reset=>Reset,
                                                    Switch => Switch,
                                                    Clock => Clock,
                                                    Clock_Enable_256Hz => Clock_Enable_256Hz,
                                                    Packet  => Packet,
                                                    start_output => start_output,
                                                    fourbitoutput => fourbitoutput,
                                                    SingleRun => SingleRun,
                                                    Go_Button => Go_Button,
                                                    Clock_Enable_4Hz => Clock_Enable_4Hz,
                                                    datacounteroutput =>datacounteroutput
                                                   );
```

```
C3: T22_Milestone_3_Packet_Transmit PORT MAP( Reset => Reset,
                                              Clock => Clock,
                                              Clock_Enable_4Hz => Clock_Enable_4Hz,
                                              Clock_Enable_64Hz => Clock_Enable_64Hz,
                                              onebitoutput=>onebitoutput,
                                              packet=>packet,
                                              datacounteroutput =>datacounteroutput,
                                              serialflag => serialflag
                                             );

C4: T22_Milestone_3_Packet_receive PORT MAP( Reset => Reset,
                                             Clock => Clock,
                                             Clock_Enable_64Hz => Clock_Enable_64Hz,
                                             onebitoutput=>onebitoutput,
                                             receive=>receive,
                                             headerflag=>headerflag,
                                             dataflag=>dataflag,
                                             parityflag=>parityflag,
                                             checksumflag=>checksumflag);

C5: T22_Milestone_3_Display PORT MAP(        Reset=>Reset,
                                             Clock => Clock,
                                             Clock_Enable_256Hz => Clock_Enable_256Hz,
                                             Clock_Enable_4Hz => Clock_Enable_4Hz,
                                             Packet => Packet,
                                             start_output=>start_output,
                                             fourbitoutput=>fourbitoutput,
                                             Display_mode=>Display_mode,
                                             Anode_Enable=>Anode_Enable,
                                             Segment_Cathodes=>Segment_Cathodes,
                                             led=>led,
                                             receive=>receive,
                                             headerflag=>headerflag,
                                             dataflag=>dataflag,
                                             parityflag=>parityflag,
                                             checksumflag=>checksumflag,
                                             serialflag => serialflag
                                            );



end Behavioral;
```

Warnings:

xst	⚠	Xst:647 - Input <Packet<63:56>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
xst	⚠	Xst:647 - Input <Packet<7:0>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
xst	⚠	Xst:1780 - Signal <Digit> is never used or assigned. This unconnected signal will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_63> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_62> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_61> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_60> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_59> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_58> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_57> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_56> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_54> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_52> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_49> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_48> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_46> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_44> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_42> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_41> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_7> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_6> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_5> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_4> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_3> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_2> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_1> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <packet_transmit_0> (without init value) has a constant value of 0 in block <C3>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1710 - FF/Latch <C3/packet_transmit_0> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_1> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_2> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_3> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_4> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_5> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_6> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_7> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_41> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_42> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_44> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_46> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_48> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_49> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_52> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_54> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_56> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_57> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_58> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_59> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_60> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_61> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_62> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.
xst	⚠	Xst:1895 - Due to other FF/Latch trimming, FF/Latch <C3/packet_transmit_63> (without init value) has a constant value of 0 in block <T22_Milestone_3_Top_Entity>. This FF/Latch will be trimmed during the optimization process.

Reason behind each warning

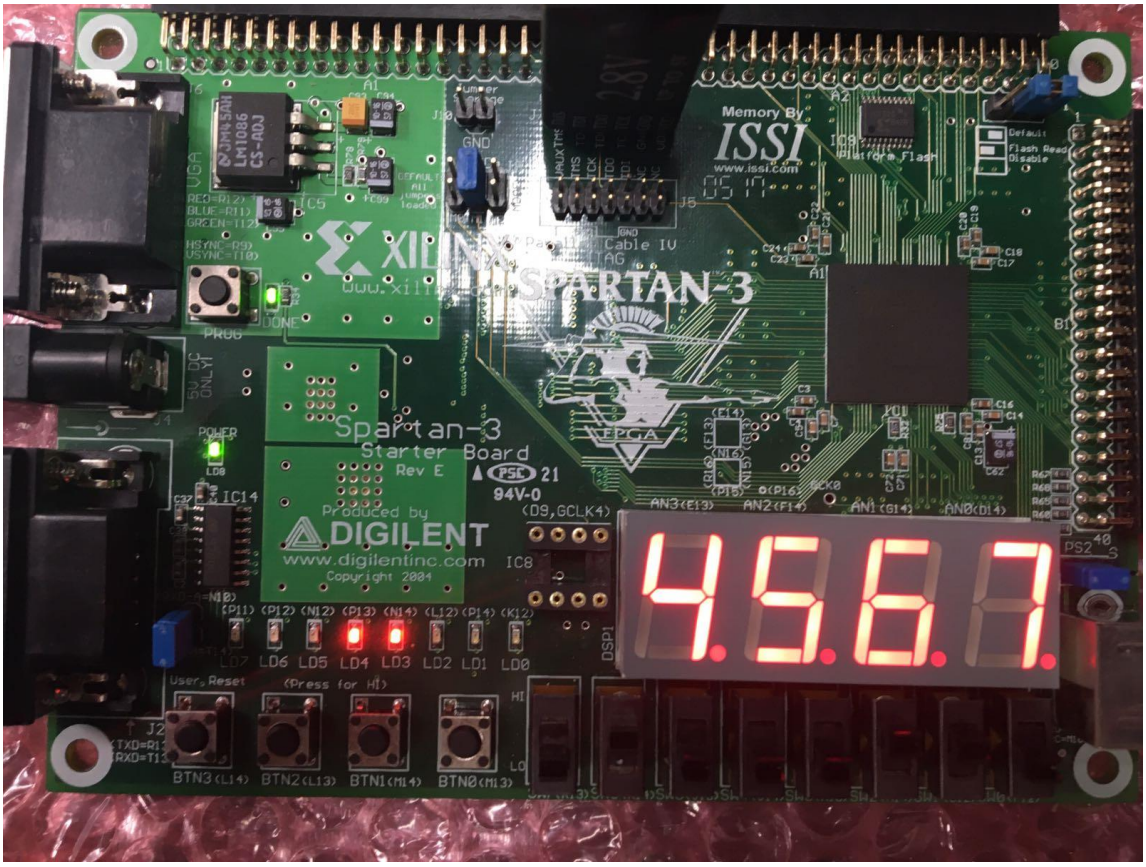
Warning xst-647: It is because the first eight 0 and last eight zero in the packet is never used at any point. However, it is needed in the packet. These warning cannot be fixed.

Warning xst-1780: it is because digit only used when the display need to be changed. We don't want to initialize a value for digit. As a result, these warning cannot be fixed.

Warning xst-1710 and xst-1895: It is because we need to initialize the number for packet transmit as all zero. However, in the packet the first eight 0, last eight zero and header are constant. The 0 in all those constant will always be the same. As are result, the system think those bit are not useful because of they never get changed. Since all those bits are needed, these warning cannot be fixed.

Implementation:

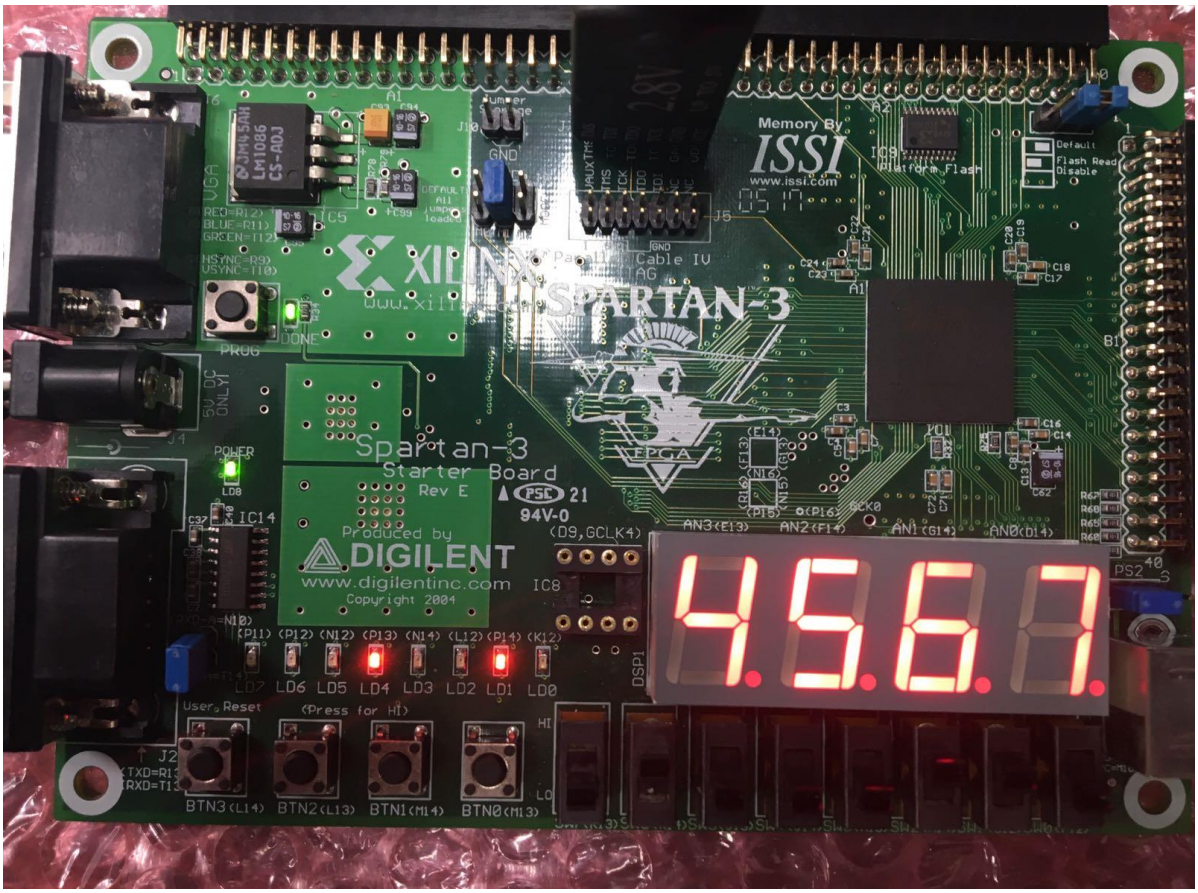
Storing header



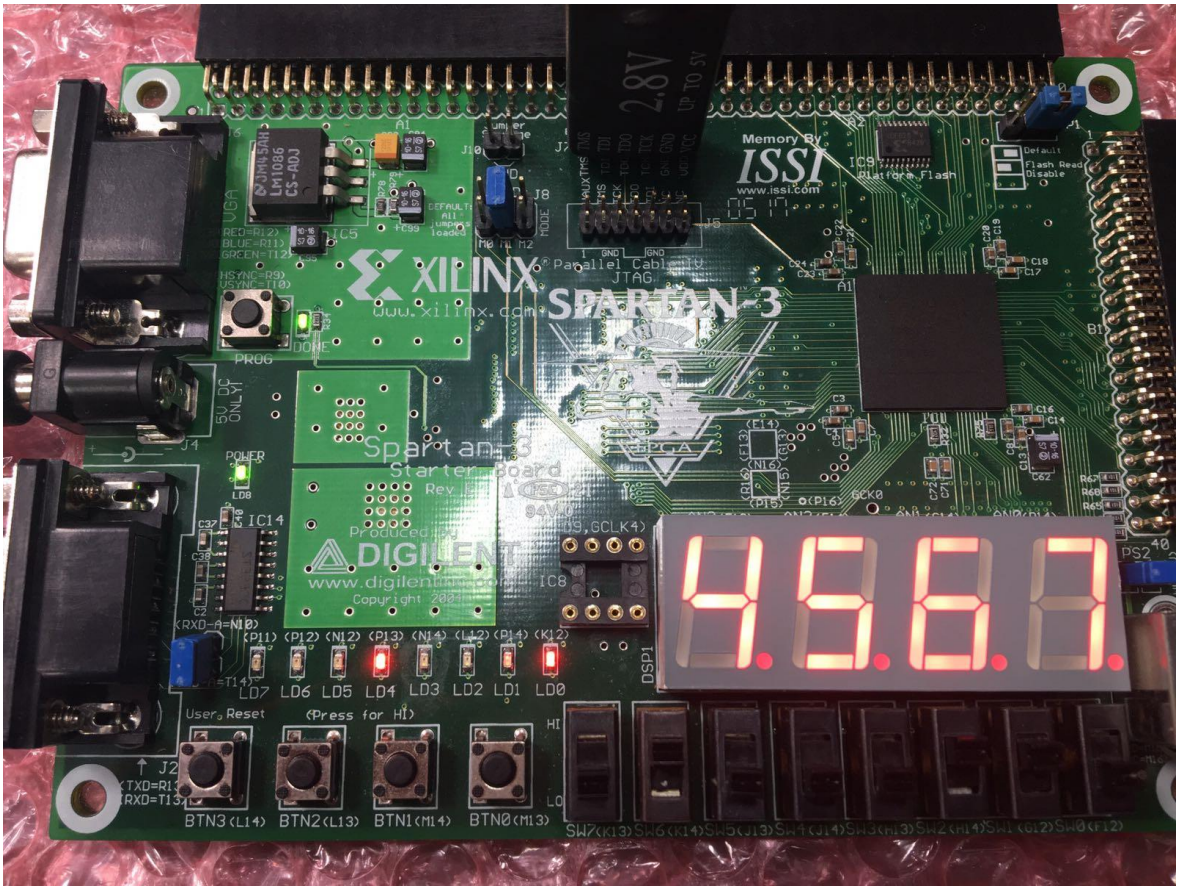
Storing data

Since it is storing the data of the fourth number, the display of the fourth number is still stay at 3 which is not changed from previous stage.

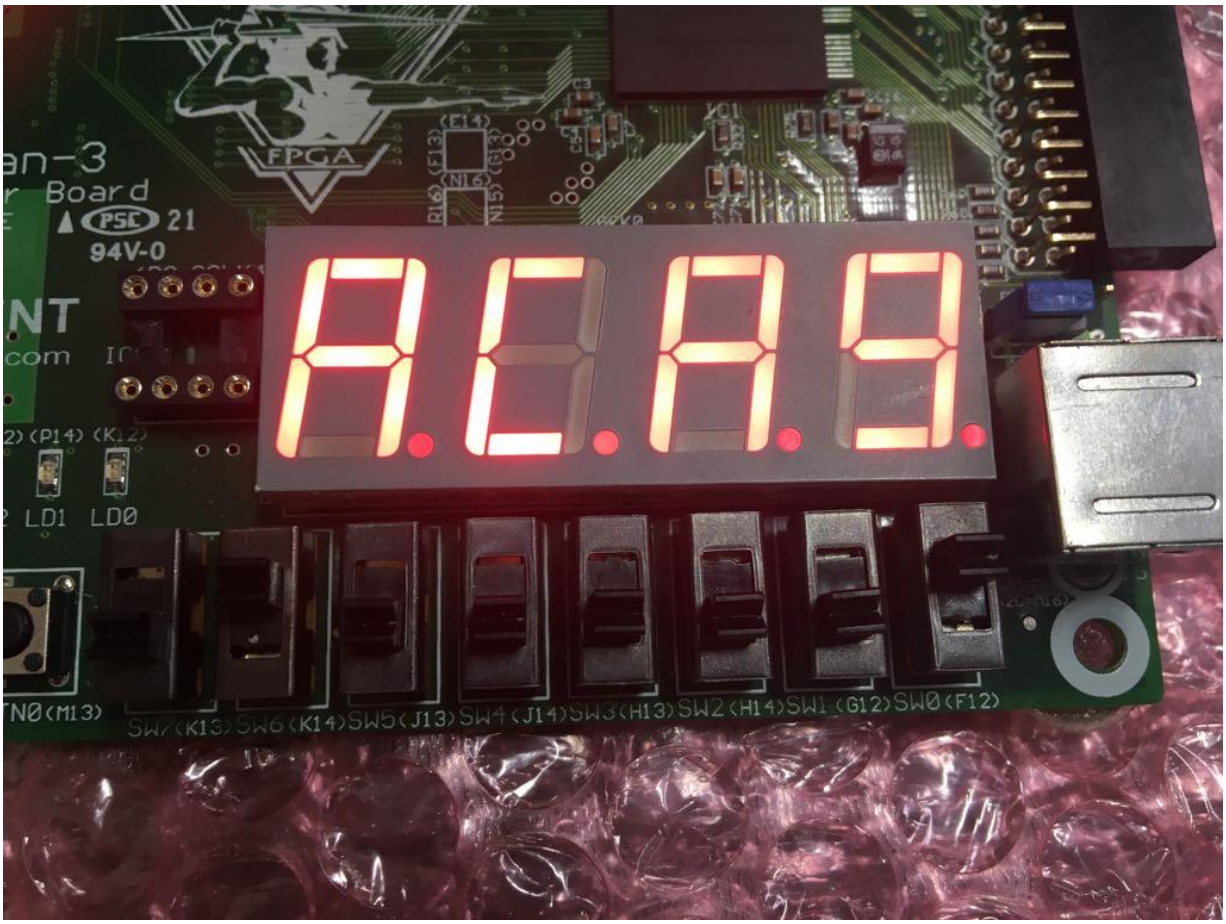
Storing parity



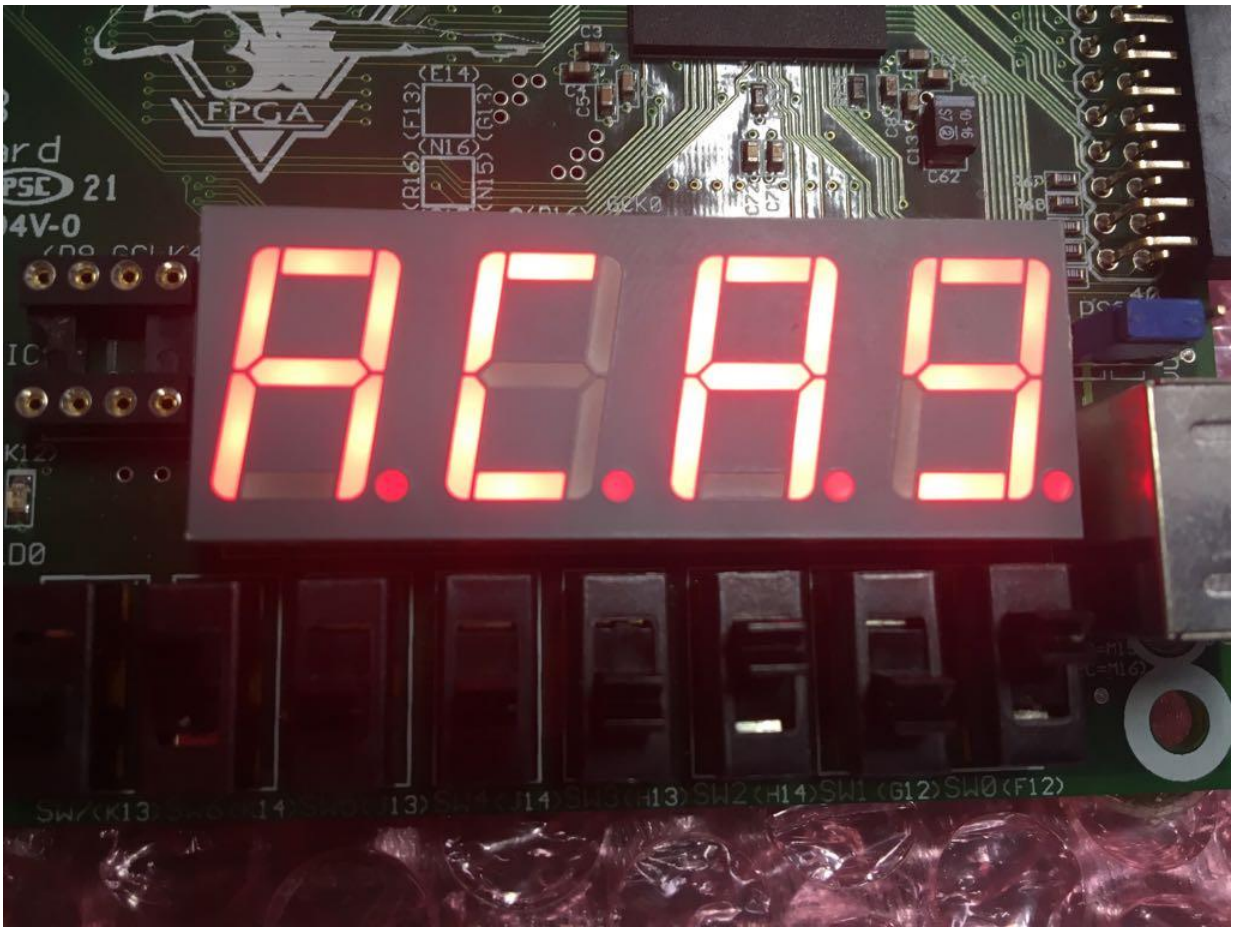
Storing checksum



Header of transmitter

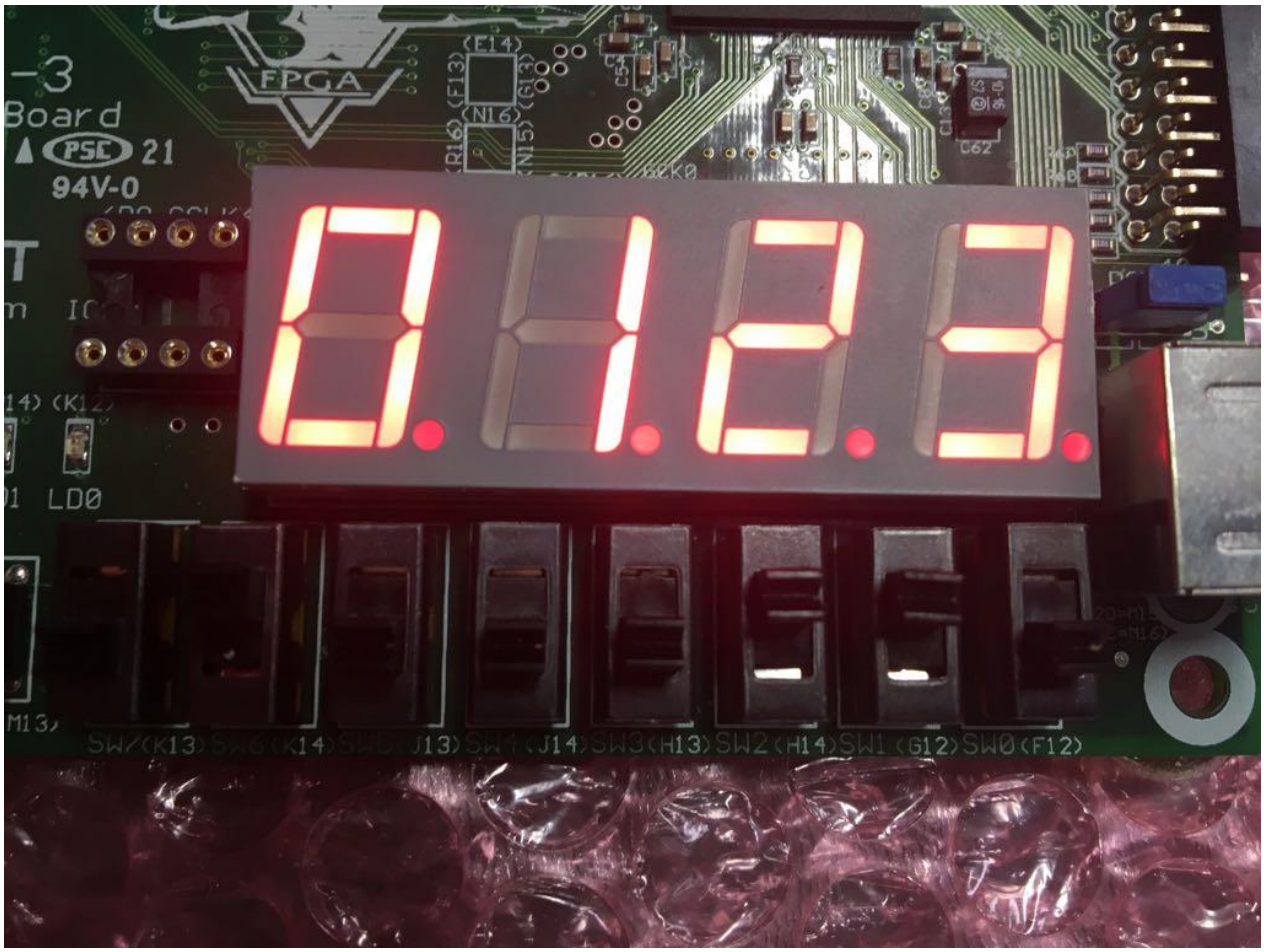


Header in receive

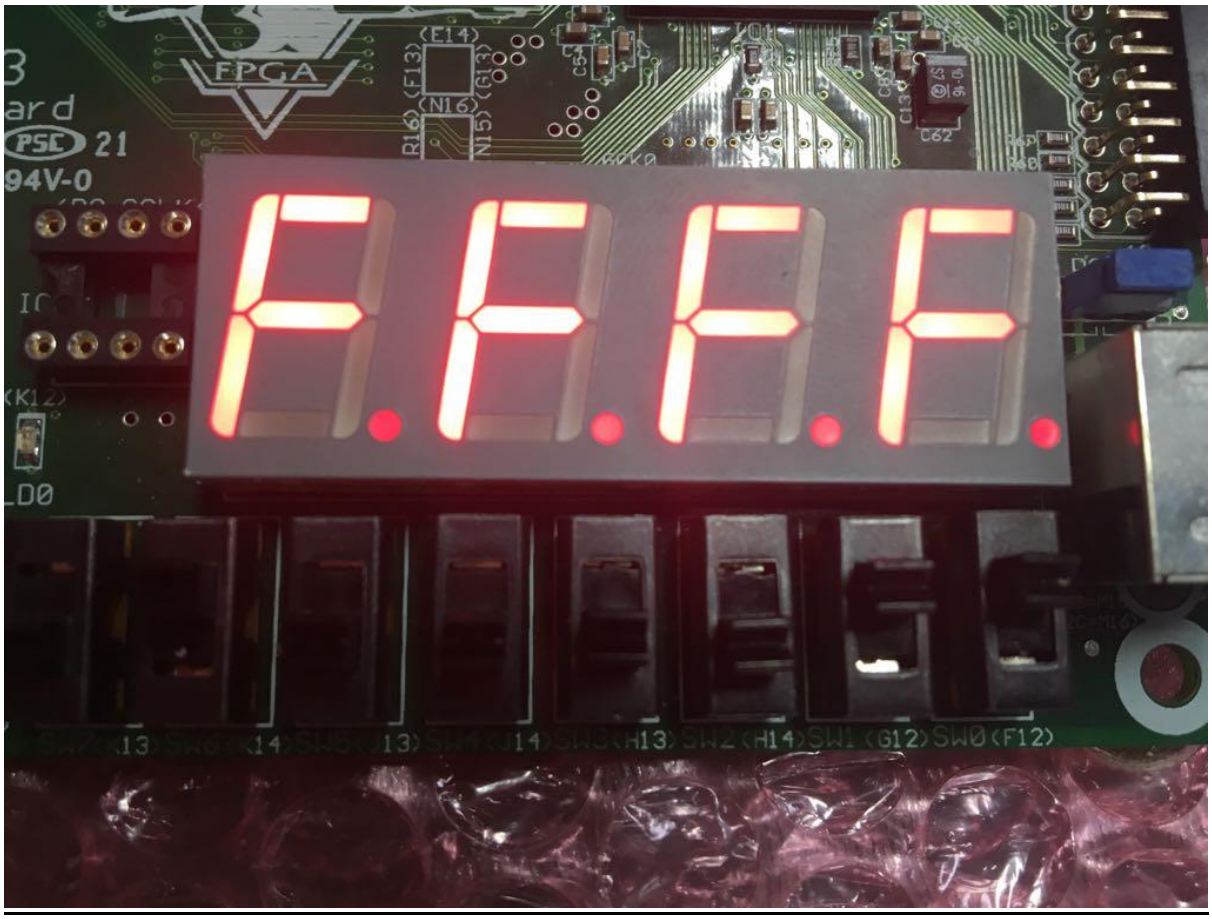


Data of transmitter

Data of receive



Parity and checksum of transmitter



Parity and checksum of receive