BAB 7 UART

I. Tujuan Praktikum :

- Praktikan Dapat Memahami Fungsi UART.
- Praktikan Dapat Mengimplementasikan UART pada FPGA.

II. Dasar Teori :

- Pengenalan UART.
- Penerapan UART pada FPGA.

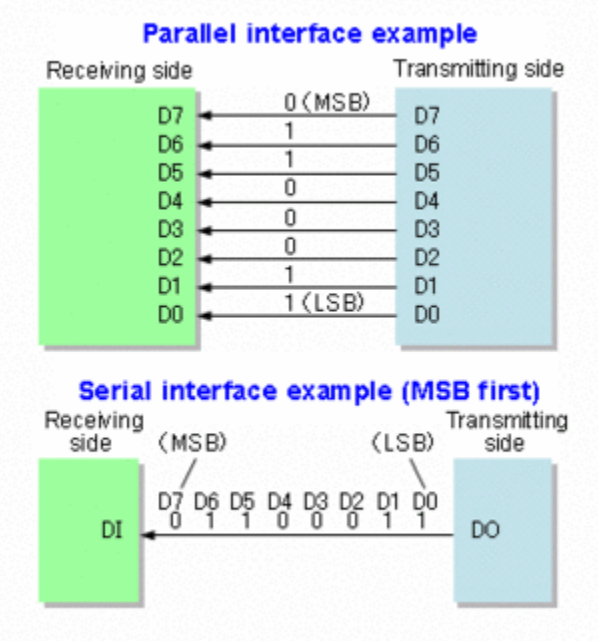
III. Peralatan :

- FPGA Xilinx Artix 7 Board Nexys 4.
- Kabel Power USB.
- 1 buah PC.
- Software ISE Design Suite 14.5.

7.1 Pengertian

7.1.1 Komunikasi Serial

Komunikasi serial adalah proses pengiriman bit data satu per satu, secara berurutan, melaui sebuah saluran komunikasi. Hal ini berbeda dengan komunikasi parallel, dimana beberapa bit dikirim secara keseluruhan, link dengan beberapa saluran parallel. Metode komunikasi serial sering disebut dengan TTL Serial (transistor-transistor logic). Komunikasi serial pada tingkat TTL akan selalu tetap antara batas 0V dan Vcc, yang sering digunakan yaitu 5V atau 3.3V. Sebuah logika tinggi (1) diwakili oleh Vcc sedangkan logika rendah (0) adalah 0V.

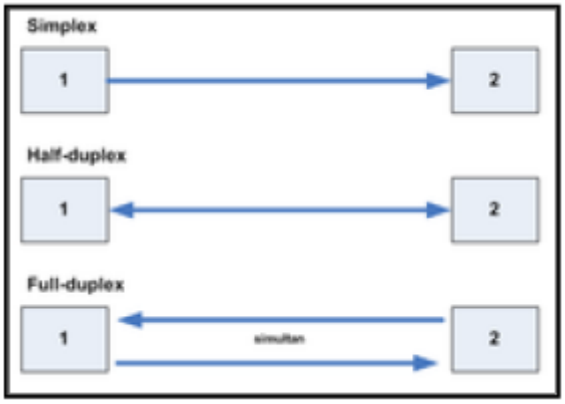


Gambar 7.1 Komunikasi serial dan parallel.

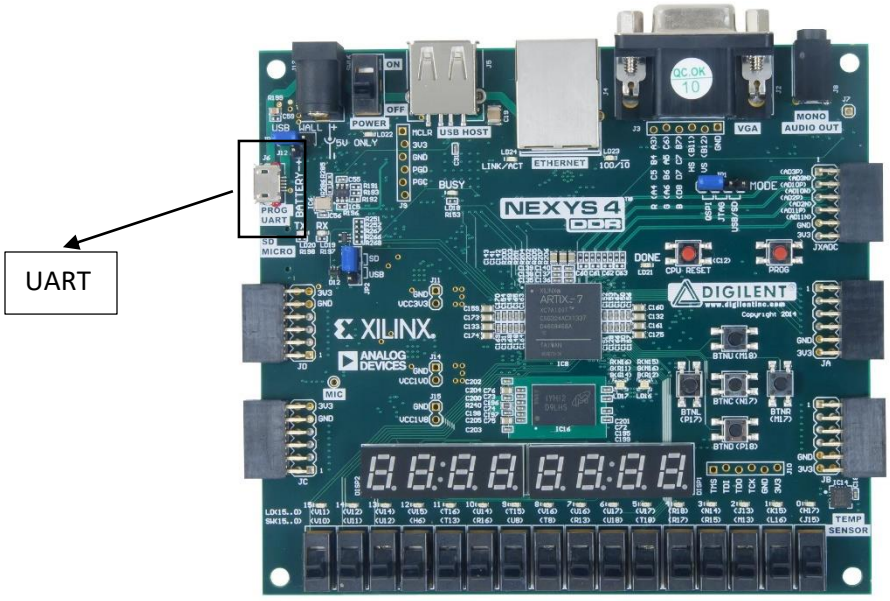
Pada saat berkomunikasi dengan semakin tingginya frekuensi pengiriman data, semakin tinggi juga gangguan elektromagnetik. Setiap kabel dapat diperlakukan sebagai antenna, menangkap noise yang ada di sekitarnya dan mengganggu data yang sedang ditransmisikan. Dalam komunikasi parallel, karena banyaknya kabel yang digunakan, masalah gangguan elektromagnetik menjadi lebih serius. Komunikasi serial dibutuhkan jumlah kabel yang lebih sedikit, bisa hanya menggunakan tiga kabel, yaitu saluran Transmit Data (Tx), sauran Receive Data (Rx) dan saluran Ground (GND).

Komunikasi serial memiliki beberapa jenis, yaitu :

a. ***Synchronous***, yaitu sebuah pengiriman data serial yang disertai dengan clock sebagai pengatur waktu untuk mengindikasi bahwa ada bit siap untuk dibaca. Contoh : I2C, USRT, SPI, dll.
b. ***Asynchronous***, yaitu sebuah pengiriman data dimana clock tidak dikirim bersamaan dengan data sehingga masing-masing perangkat keras yang berkomunikasi harus membuat clocknya sendiri yang sama. Contoh : UART, RS-232, USB, dll.
c. ***Full duplex,*** adalah jenis komunikasi serial yang menyatakan hubungan antara dua perangkat keras. Misalnya ada sebuah perangkat A dan B, jika A sedang melakukan pengiriman data, pada saat yang sama A dapa menerima data dari B, begitupun sebaliknya. Kondisi ini dinamakan full duplex atau komunikasi dua arah, contohnya adalah telepon.
d. ***Half duplex,*** merupakan kondisi ketika proses pengiriman dan penerimaan data tidak dapat dilakukan secara bersamaan, namun secara bergantian. Contohnya adalah walkie talkie.
e. ***Simplex,*** merupakan jenis komunikasi dimana pengiriman hanya terjadi satu arah saja kepada penerima. Contohnya seperti membroadcast sebuah pesan atau seperti tv.



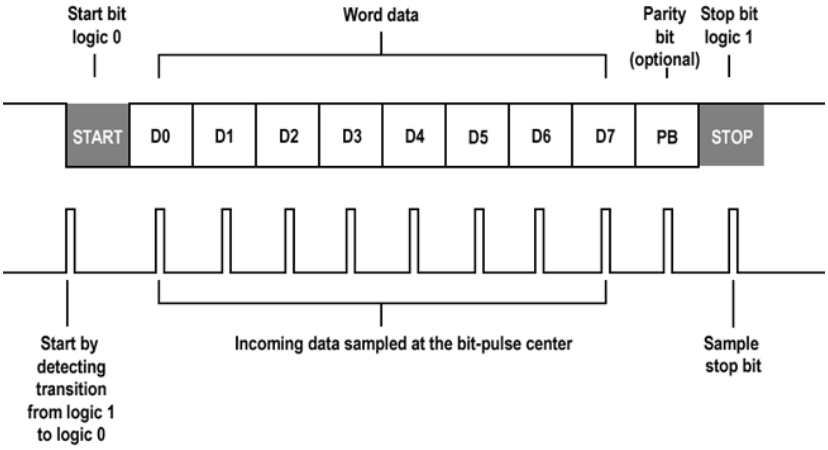
Gambar 7.2 Jenis komunikasi.



Gambar 7.3 Port UART.

7.1.2 UART

UART atau Universal Asynchronous Receiver Transmitter adalah protocol komunikasi serial yang umum digunakan dalam pengiriman data serial antara device satu dengan yang lainnya. Sebagai contoh, komunikasi antara sesama mikrokontroler atau mikrokontroler dengan PC.



Gambar 7.4 Format data komunikasi serial.

Dalam pengiriman data menggunakan UART, baud rate antara pengirim dan penerima harus sama karena paket data dikirim tiap bit mengandalkan clock tersebut. Inilah salah satu keuntungan menggunakan model asynchronous dalam pengiriman data karena dengan hanya satu kabel transmisi maka data dapat dikirimkan. Berbeda dengan model synchronous yang terdapat pada protocol SPI dan I2C karena protocol tersebut membutuhkan minimal dua kabel dalam transmisi data, yaitu transmisi clock dan data. Namun kelemahan model asynchronous adalah dalam hal kecepatannya dan jarak transmisi. Semakin cepat dan jauhnya jarak transmisi membuat paket-paket bit data menjadi terdistorsi sehingga data yang dikirim atau diterima bisa mengalami error.

7.2 Cara Kerja UART

Sama seperti komunikasi serial lainnya, digunakan bit Start yang bernilai logic 0 sebagai tanda kepada penerima bahwa ada data yang akan dikirimkan. Bit individu dari data yang dikirim dimulai dari yang terkecil pertama (LSB). Setiap bit dalam transmisi ditransmisikan serupa dengan jumlah bit lainnya, dan penerima mendeteksi jalur disekitar pertengahan periode setiap bit untuk menentukan apakah bit adalah 1 atau 0. Misalnya, jika dibutuhkan dua detik untuk mengirim setiap bit, penerima akan memeriksa sinyal untuk menentukan apakah itu adalah 1 atau 0 setelah satu detik telah berlalu.

UART memiliki tugas mengubah data yang diterima dari computer melewati sirkuit parallel menjadi bit stream serial untuk dikirimkan ke perangkat keras dan sebaliknya. UART juga berfungsi menambahkan bit parity untuk melindungi data dari kesalahan, menambahkan start bit dan stop pada waktu pengiriman data, serta menangani interrupt dari perangkat keras.

7.3 FTDI Chip

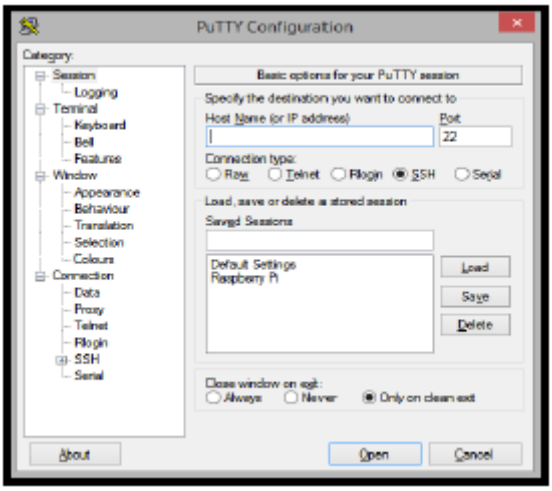
Komunikasi UART tidak bisa diterapkan melalui jalur USB secara langsung tanpa menggunakan sebuah modul FTDI. Modul ini mengelola transaksi data dengan menerapkan protocol USB dari data yang ditransfer melalui serial asinkron. Tanggungjawab perangkat ini adalah untuk memberitahu PC bahwa perangkat yang digunakan dengan menambahkan beberapa informasi identifikasi, sehingga PC dapat memuat driver yang tepat.

Gambar 7.5 Chip FTDI

7.4 PuTTY

PuTTY adalah sebuah program open source yang dapat digunakan untuk melakukan protocol jaringan SSH, Telnet, Serial dan Rlogin. Protocol ini dapat digunakan untuk menjalankan sesi remote pada sebuah computer melalui sebuah jaringan baik itu LAN maupun internet. PuTTY pada awalnya dibuat untuk Microsoft Windows, tetapi telah mendukung berbagai system operasi lainnya.

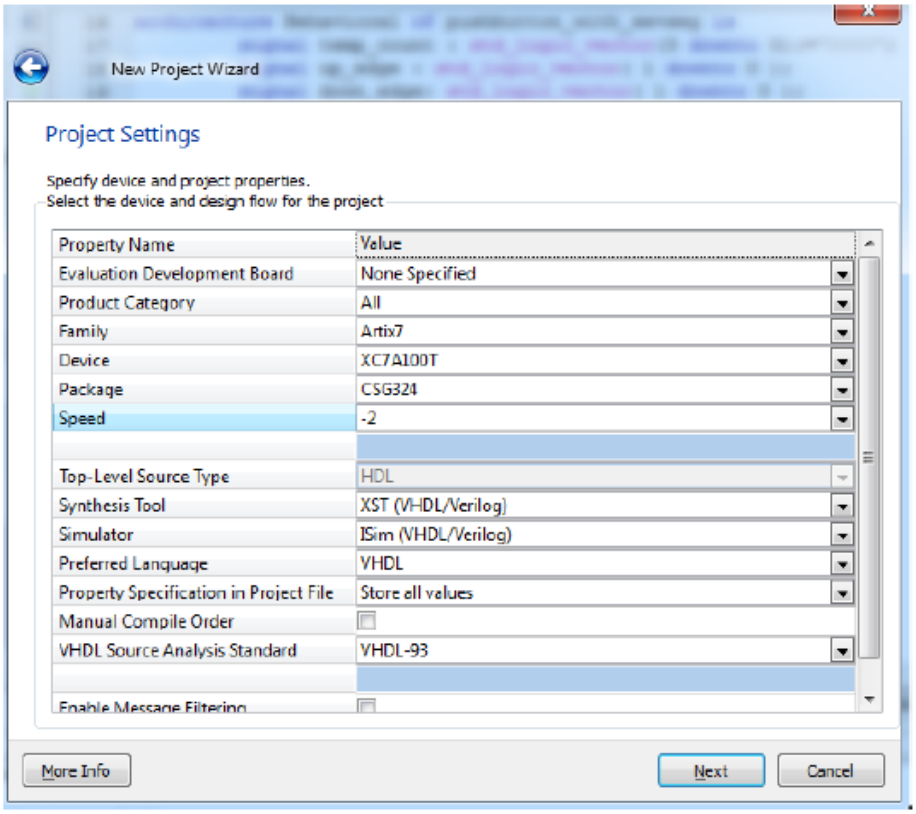
Aplikasi PuTTY digunakan ketika ingin mentransfer sebuah data dari computer ke sebuah perangkat lain. Program ini juga sering digunakan untuk menyambungkan, mensimulasi atau mencoba berbagai hal yang terkait dengan jaringan.



Gambar 7.6 Tampilan awal PuTTY

7.5 Membuat Program Transmitter

1. Buka aplikasi Xilinx ISE Design Suite 14.5.
2. Klik File → New Project.
3. Berilah nama pada project tersebut, lalu klik "**Next**".
4. Atur device properties seperti dibawah ini :



Gambar 7.7 Project setting.

5. Klik kanan pada tab hierarchy, pilih "**New Source**" → ketika nama file menjadi "**vhdl_transmitter**" → pilih "**VHDL Module**" → klik "**Next**".

6. Setelah muncul tampilan project yang akan dibuat, isikan program seperti dibawah ini :

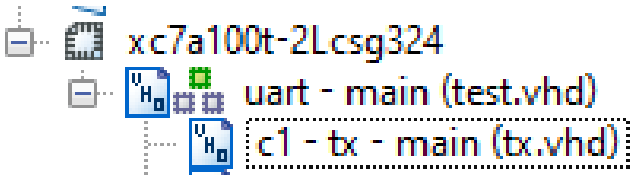
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity uart is PORT(
6     clk: in std_logic;
7     sw: in std_logic_vector(7 downto 0);
8     send: in std_logic;
9     uart_tx: out std_logic; --N18
10    uart_rx: in std_logic --N17
11 );
12 end uart;
13 ----------------------------------------------------------------------
14 architecture main of uart is
15    signal tx_data: std_logic_vector(7 downto 0);
16    signal tx_start: std_logic:='0';
17    signal tx_busy: std_logic;
18    signal send_edge: std_logic_vector(1 downto 0):="00";
19 ----------------------------------------------------------------------
20 component tx port(
21    clk:in std_logic;
22    start: in std_logic;
23    busy: out std_logic;
24    data : in std_logic_vector(7 downto 0);
25    tx_line: out std_logic
26 );
27 end component tx;
```

```
28 ----------------------------------------------------------------------
29 begin
30
31 c1: tx port map(clk,tx_start,tx_busy, tx_data,uart_tx);
32
33 ----------------------------------------------------------------------
34 process(clk)
35 begin
36    if(clk'event and clk='1') then
37       send_edge<=send_edge(0) & send;
38       if(send_edge="01" and tx_busy='0') then
39          tx_data<=sw(7 downto 0);
40          tx_start<='1';
41       else
42          tx_start<='0';
43       end if;
44    end if;
45 end process;
46 ----------------------------------------------------------------------
47 end main;
```

Gambar 7.8 Program VHDL

Setelah menuliskan program tersebut, lalu simpan.

7. Pada tab hierarchy, pilih "**Add Source".** Lalu masukkan file "**tx**" yang akan diberikan oleh asisten, setelah itu simpan project tersebut.

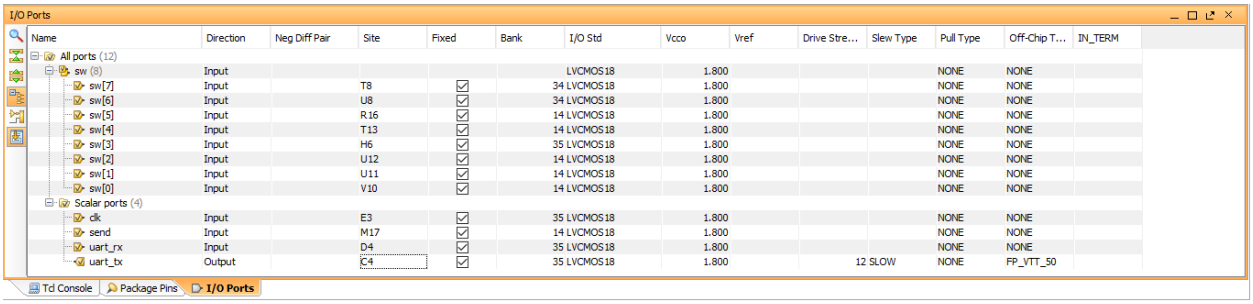


Gambar 7.9 File TX.

Isi dari file "**tx**" tersebut adalah sebuah program untuk transmitter.

8. Klik kanan pada "**Synthesize**" lalu pilih "**Run**".

9. Setelah itu, klik expand pada "**User Constraints**" lalu pilih I/O Planning (PlanAhead) – Pre Synthesis. Jalankan dengan cara klik kanan lalu pilih "**Run**". Isikan seperti berikut, kemudian simpan.



Gambar 7.10 PlanAhead.

Keterangan :

- Sw(0) = V10
- Sw(1) = U11
- Sw(2) = U12
- Sw(3) = H6
- Sw(4) = T13
- Sw(5) = R16
- Sw(6) = U8
- Sw(7) = T8
- Clk = E3
- Send = M17
- Uart_rx = D4
- Uart_tx = C4

Setelah selesai, klik tombol “Save” yang ada pada kiri atas.

10. Kembali ke software Xilinx, lalu klik expand pada menu berikut. Terdapat sebuah file dengan ekstensi .ucf, buka file tersebut dengan cara klik dua kali lalu pilih “Yes”. Pastikan tampilannya sudah seperti berikut ini :

```
1  NET "sw[0]" LOC = V10;
2  NET "sw[1]" LOC = U11;
3  NET "sw[2]" LOC = U12;
4  NET "sw[3]" LOC = H6;
5  NET "sw[4]" LOC = T13;
6  NET "sw[5]" LOC = R16;
7  NET "sw[6]" LOC = U8;
8  NET "sw[7]" LOC = T8;
9
10 NET "clk" LOC = E3;
11 NET "send" LOC = M17;
12 NET "uart_tx" LOC = C4;
13 NET "uart_rx" LOC = D4;
14
15
16
17 NET "clk" IOSTANDARD = LVCMOS18;
18 NET "send" IOSTANDARD = LVCMOS18;
19 NET "uart_tx" IOSTANDARD = LVCMOS18;
20 NET "uart_rx" IOSTANDARD = LVCMOS18;
21
22 NET "sw[7]" IOSTANDARD = LVCMOS18;
23 NET "sw[6]" IOSTANDARD = LVCMOS18;
24 NET "sw[5]" IOSTANDARD = LVCMOS18;
25 NET "sw[4]" IOSTANDARD = LVCMOS18;
26 NET "sw[3]" IOSTANDARD = LVCMOS18;
27 NET "sw[2]" IOSTANDARD = LVCMOS18;
28 NET "sw[1]" IOSTANDARD = LVCMOS18;
29 NET "sw[0]" IOSTANDARD = LVCMOS18;
30
```

Gambar 7.11 File UCF.

NOTE :

- Kirim dua buah file kepada PJ Shift masing-masing yaitu file dengan ekstensi .vhd dan .ucf.
- File tersebut dapat ditemukan pada local disk C:\Xilinx lalu pilih nama file yang sudah dibuat.