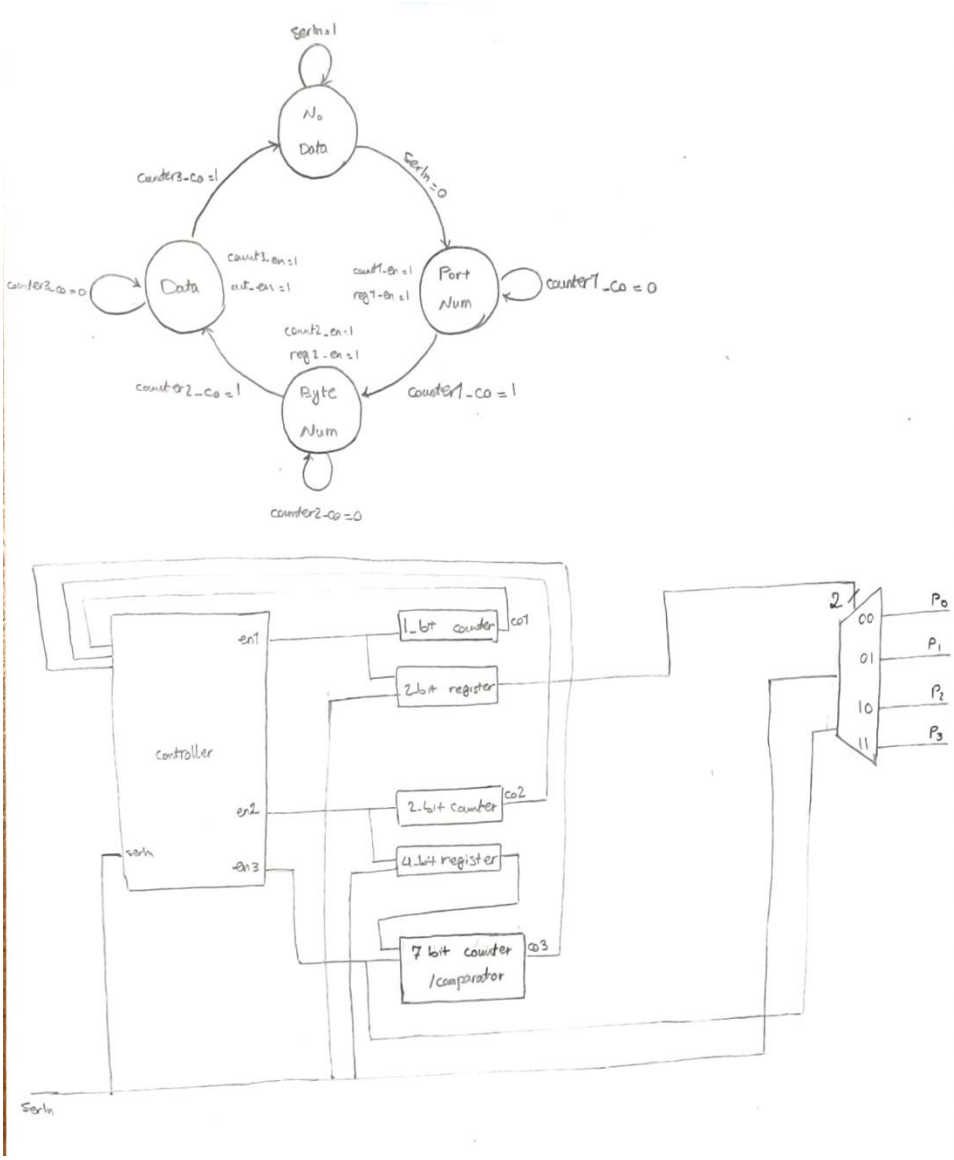
Digital Systems

Computer Assignment 4-5

State Machines and Basic RTL

Mana Mohammadi
810100207



روش کار مدار به این صورت است که ابتدا در state : NoData منتظر 0 شدن serIn می مانیم؛ هنگامی که ورودی از 1 به 0 تغییر کرد وارد state : PortNum می شویم، در این استیت یک کانتر 1 بیتی و یک رجیستر 2 بیتی را enable میکنیم، هنگامی که کانتر 2 کلاک را بشمارد، carry out آن 1 میشود، در این حالت کنترلر استیت را به state : ByteNum تغییر میدهد. در این استیت هم شبیه به استیت قبل یک کانتر 2 بیتی و یک رجیستر 4 بیتی را enable میکنیم و کانتر بعد از شمردن 4 کلاک، carry out = 1 پیدا میکند که به واسطه آن، کنترلر استیت را به Data تغییر میدهد. در استیت Data، یک demultiplexer داریم که serIn ما را به پورت خروجی مورد نظر ما انتقال میدهد. همینطور یک counter / comparator را enable میکنیم که هر گاه شمارش کلاک آن برابر با تعداد بیت مورد نظر ما شد، co یک تحویل دهد (به عنوان ورودی به این کامپونت، تعداد بایت های ذخیره شده در رجیستر 4 بیتی ضربدر 8 تحویل میدهیم؛ یعنی 3 بار به چپ شیفت میدهیم.) بعد از 1 شدن carry out این کانتر، به state : NoData می رویم.

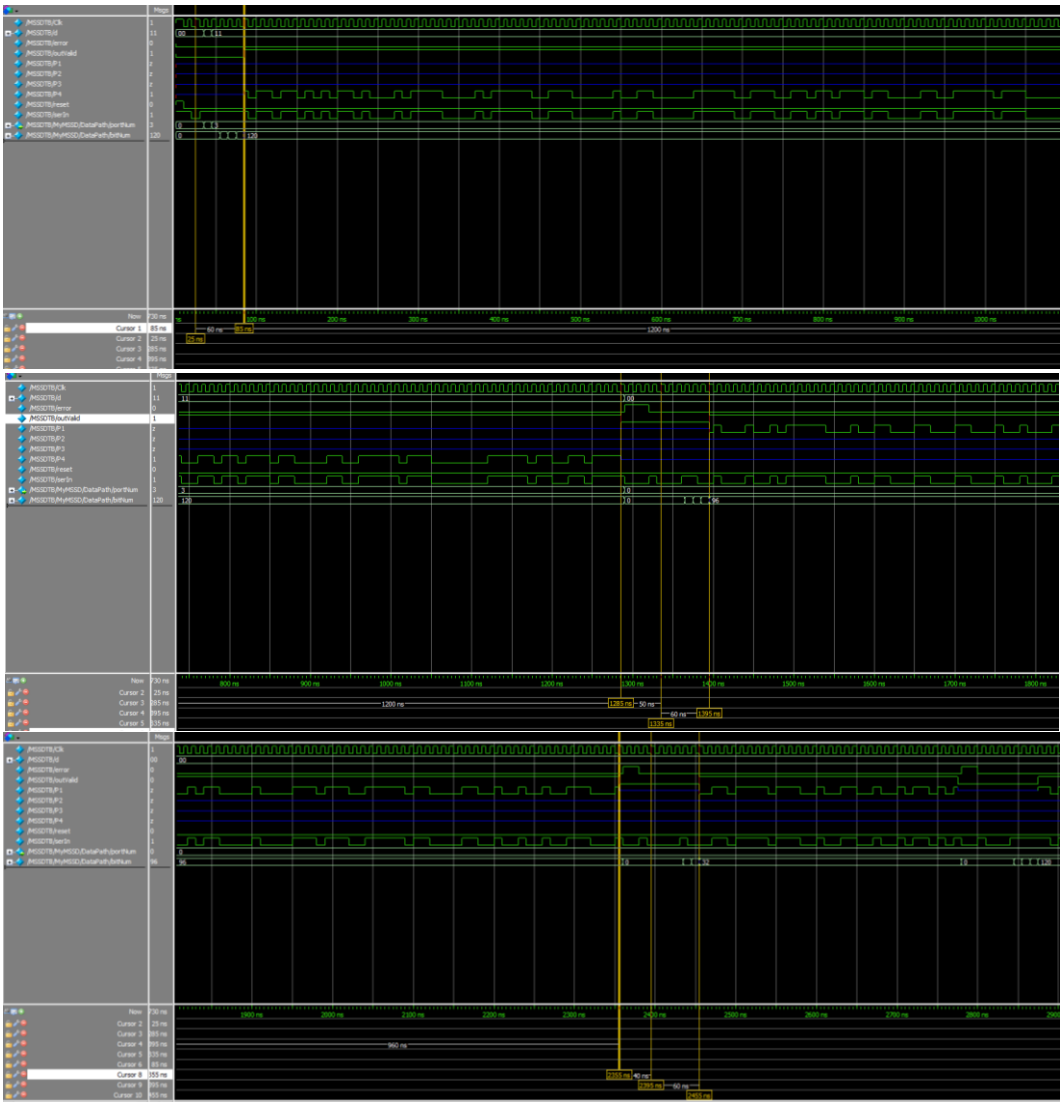
```
always @(posedge Clk, ps, onecountCo, TwocountCo, downcountCo)begin
        case(ps)
                NoData : ns <= (serIn | error) ? NoData : PortNum;
                PortNum : ns <= onecountCo ? ByteNum : PortNum;
                ByteNum : ns <= TwocountCo ? Data : ByteNum;
                Data : ns <= downcountCo ? NoData : Data;
                default : ns <= NoData;
        endcase
end
```

```
always @(ps, ns, onecountCo, TwocountCo, downcountCo)begin
        onecountEn = 1'b0; tworegEn = 1'b0; twocountEn = 1'b0; fourregEn = 1'b0; downcountEn = 1'b0; outEn = 1'b0;
        case(ns)
                NoData :        begin
                                downcountEn = 0;
                                outEn = 0;
                                onecountEn = 0;
                                tworegEn = 0;
                                twocountEn = 0;
                                fourregEn = 0;
                                outValid = 0;
                                end
                PortNum :       begin
                                onecountEn = 1;
                                tworegEn = 1;
                                end
                ByteNum :       begin
                                twocountEn = 1;
                                fourregEn = 1;
                                onecountEn = 0;
                                tworegEn = 0;
                                end
                Data :          begin
                                downcountEn = 1;
                                outEn = 1;
                                twocountEn = 0;
                                fourregEn = 0;
                                outValid = 1;
                                end
        endcase
end
```

- خروجی های دیگر مدار :
  - d : همان PortNum است.
  - outValid : هنگامی که در استیت Data هستیم 1 میشود تا یوزر بداند که در حال انتقال دیتای مورد نظر هستیم.
  - error : هنگامی که ps : Data و ns : NoData باشد، یعنی در واقع وقتی انتقال دیتا تمام شده، اگر serIn=0 باشد، ارور یک میشود و تا وقتی که serIn 1 نشده باشد، همچنان ارور یک میماند.

```
if(((ps==Data)&(ns==NoData)) | (error==1)) begin
        error = ~serIn;
end
ps<=ns;
```

خروجی :



- مشاهده می شود که پس از 0 شدن serIn به اندازه 6 کلاک (60ns) طول میکشد تا مقداری روی پورت خروجی برود، یا outValid برابر 1 شود؛ که به اندازه 2 کلاک برای PortNum و 4 کلاک برای ByteNum است.
- دفعه دوم و سوم مشاهده میشود که serIn پس از مبادله داده و تمام شدن دیتا ها 1 نشده و در نتیجه ارور 1 میشود و 6 کلاک پس از 0 شدن آن (1 شدن serIn) خروجی روی پورت مورد نظر دیده میشود.
- همینطور در عکس ها دیده میشود که تبادل داده ها دفعه اول و دوم به ترتیب 1200ns و 960ns طول کشیده که همان 120 کلاک و 96 کلاک هستند که برابر با BitNum هر تبادلند.