CS223 Digital Design Section 2

Lab 5 – Preliminary Report Seçkin Alp

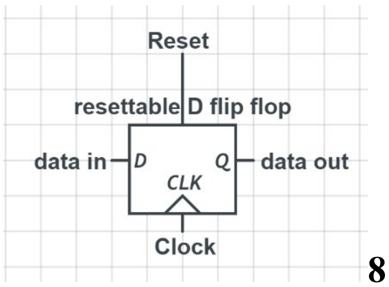
Kargı

22001942

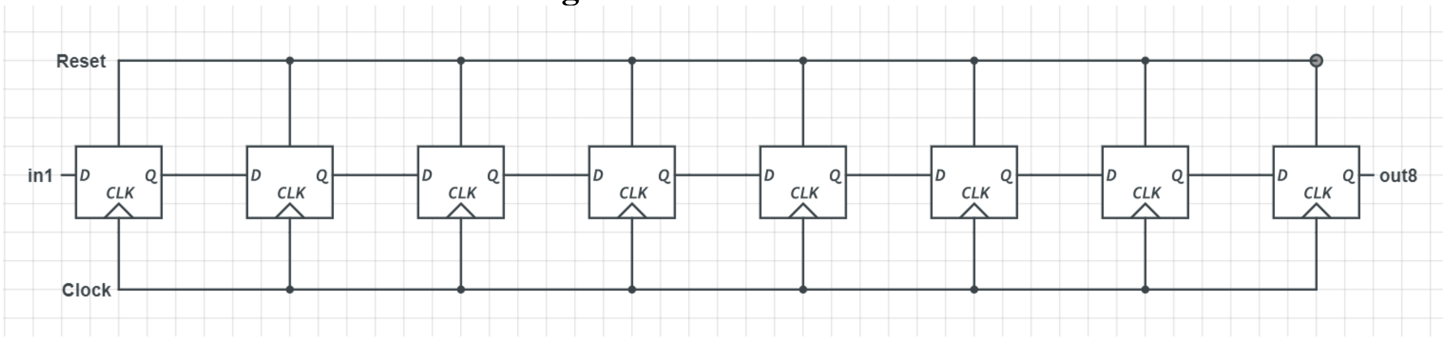
CS

21.05.2023

Circuit schematic for your shift register design using D flip-flops

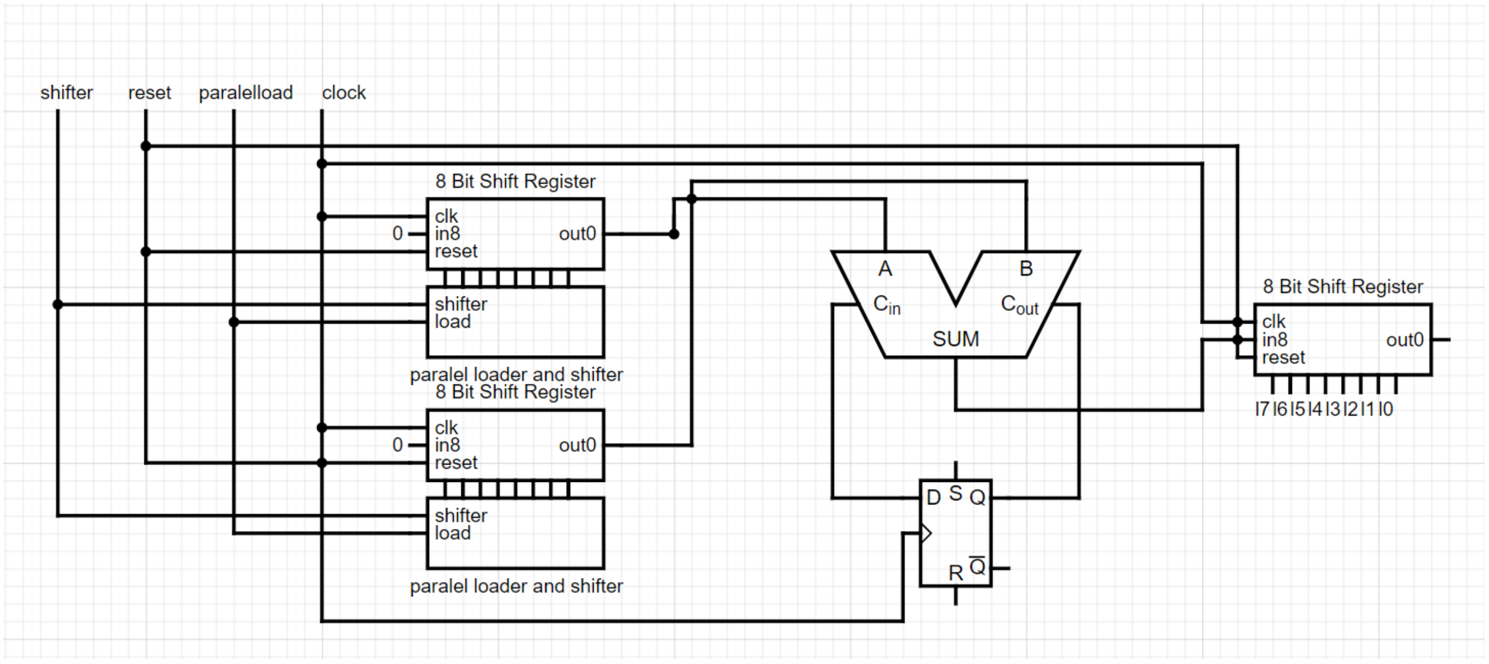


8 bit shift register



I used eight resettable d flip- flops.

Circuit schematic for your serial adder using the shift registers, full adder, and D flip-flop.



SystemVerilog module for synchronously resettable D flip-flop.

```
`timescale 1ns / 1ps
module dflipflop
(
      input logic clock,
      input logic reset,
      input logic datain,
      output logic dataout
 );
always @(posedge clock)
begin
    if( reset)
    begin
      dataout <=  0;
    end
    else
    begin
      dataout <= datain;
    end
end
endmodule
```

TestBench for D flipflop

```
`timescale 1ns / 1ps
module dflipflop_test;
  logic clock;
  logic reset;
  logic dataout;
  logic datain;
  dflipflop x
  (
    .clock(clock),
    .reset(reset),
    .datain(datain),
    .dataout(dataout)
  );
   always #5  clock   = ~clock;
   initial begin
   clock = 0;
    reset = 1;#10;
    datain = 0; #10;
    datain = 1; #10;
    datain = 0; #10;
    datain = 0; #10;
    datain = 1;#10;
    reset = 0;#10;
    datain = 0; #10;
    datain = 1; #10;
    datain = 0; #10;
    datain = 0; #10;
    datain = 1;#10;
    datain = 0; #10;
    datain = 1; #10;
    datain = 0; #10;
    datain = 1; #10;
  end endmodule
```

Structural SystemVerilog module for your shift register using the D flip-flop module along with the testbench.

```
`timescale 1ns / 1ps
module shiftregister
(
    input logic datain1,
    input  logic datain2,
    input logic datain3,
    input logic datain4,
    input logic  datain5,
    input logic datain6,
    input logic datain7,
    input logic datain8,
    output logic dataout1,
    output logic dataout2,
    output logic dataout3,
    output logic dataout4,
    output logic dataout5,
    output logic dataout6,
    output logic dataout7,
    output logic dataout8,
    input logic clock,
    input logic reset
);
Assign datain2 = dataout1;
Assign datain3 = dataout2;
Assign datain4 = dataout3;
Assign datain5 = dataout4;
Assign datain6 = dataout5;
Assign datain7 = dataout6;
Assign datain8 = dataout7;
  dflipflop x1
  (
    .clock(clock),
    .reset(reset),
    .datain(datain1),
    .dataout(dataout1)
  );
    dflipflop x2
  (
    .clock(clock),
    .reset(reset),
    .datain(datain2),
    .dataout(dataout2)

  );
    dflipflop x3
  (
    .clock(clock),
    .reset(reset),
    .datain(datain3),
    .dataout(dataout3)

  );
    dflipflop x4
  (
    .clock(clock),
    .reset(reset),
    .datain(datain4),
    .dataout(dataout4)

  );
    dflipflop x5
  (
    .clock(clock),
    .reset(reset),
    .datain(datain5),
    .dataout(dataout5)

  );
    dflipflop x6
  (
    .clock(clock),
    .reset(reset),
    .datain(datain6),
    .dataout(dataout6)

  );
    dflipflop x7
  (
    .clock(clock),
    .reset(reset),
    .datain(datain7),
    .dataout(dataout7)

  );
    dflipflop x8
  (
    .clock(clock),
    .reset(reset),
    .datain(datain8),
    .dataout(dataout8)

  );
Endmodule
```

TestBench for Shift Register

```
`timescale 1ns / 1ps
module shiftregister_test;
logic datain1;
logic datain2;
logic datain3;
logic datain4;
logic  datain5;
logic datain6;
logic datain7;
logic datain8;
logic dataout1;
logic dataout2;
logic dataout3;
logic dataout4;
logic dataout5;
logic dataout6;
logic dataout7;
logic dataout8;
logic clock;
logic reset;
shiftregister y
(
.reset(reset),
    .datain1(datain1),
    .datain2(datain2),
    .datain3(datain3),
    .datain4(datain4),
    .datain5(datain5),
    .datain6(datain6),
    .datain7(datain7),
    .datain8(datain8),
    .dataout1(dataout1),
    .dataout2(dataout2),
    .dataout3(dataout3),
    .dataout4(dataout4),
    .dataout5(dataout5),
    .dataout6(dataout6),
    .dataout7(dataout7),
    .dataout8(dataout8),
    .clock(clock)
    );
always #2.5 clock = ~clock;
  initial begin
  clock = 0;
    for(int i = 0; i < 512; i = i + 1)
       begin
        {reset,datain1,datain2,datain3,datain4,datain5,datain6,datain7,datain8} = i;
        #2.5;
    end    end endmodule
```

Full Adder

```
`timescale 1ns / 1ps

module fulladder
(
    input logic carryin,
    input logic datain1,
    input logic datain2,
    output logic dataout,
    output logic carryout

);

assign carryout  = (datain2 & carryin) | ( datain1 &  carryin) |
(datain1 & datain2)  ;
assign dataout   = carryin ^datain1 ^ datain2;

endmodule
```

Test Bench Full Adder

```
`timescale 1ns / 1ps
module fulladder_test;
logic datain1;
logic datain2;
logic carryin;
logic dataout;
logic carryout;

fulladder z
(
   .datain1(datain1),
   .datain2(datain2),
   .carryin(carryin),
   .dataout(dataout),
   .carryout(carryout)
);
initial
begin
datain1 = 0; datain2 = 0; carryin = 0; #10;
datain1 = 0; datain2 = 0; carryin = 1; #10;
datain1 = 0; datain2 = 1; carryin = 0; #10;
datain1 = 0; datain2 = 1; carryin = 1; #10;
datain1 = 1; datain2 = 0; carryin = 0; #10;
datain1 = 1; datain2 = 0; carryin = 1; #10;
datain1 = 1; datain2 = 1; carryin = 0; #10;
datain1 = 1; datain2 = 1; carryin = 1; #10;
end
endmodule
```

Structural SystemVerilog module for your serial adder using the shift register, full adder, and D flip-flop modules along with the testbench.

```
`timescale 1ns / 1ps

module serialadder
(
    input logic shiftsignal,
    input logic paralelloadsignal,
    input logic reset,
    input logic clock,
    input logic adatain1,
    input logic adatain2,
    input logic adatain3,
    input logic adatain4,
    input logic adatain5,
    input logic adatain6,
    input logic adatain7,
    input logic adatain8,
    input logic bdatain1,
    input logic bdatain2,
    input logic bdatain3,
    input logic bdatain4,
    input logic bdatain5,
    input logic bdatain6,
    input logic bdatain7,
    input logic bdatain8,
    output logic dataout1,
    output logic dataout2,
    output logic dataout3,
    output logic dataout4,
    output logic dataout5,
    output logic dataout6,
    output logic dataout7,
    output logic dataout8
```

```
);
logic aout1,aout2,aout3,aout4,aout5,aout6,aout7,aout8;
logic bout1,bout2,bout3,bout4,bout5,bout6,bout7,bout8;
logic ain1 = 0,ain2 = 0,ain3 = 0,ain4 = 0,ain5 = 0,ain6 = 0,ain7 = 0,ain8
= 0;
logic bin1 = 0,bin2 = 0,bin3 = 0,bin4 = 0,bin5 = 0,bin6 = 0 ,bin7 =
0,bin8 = 0;
logic oin1 = 0,oin2 = 0,oin3 = 0,oin4 = 0,oin5 = 0,oin6 = 0 ,oin7 = 0,oin8
= 0;
logic dclock ;
logic dout ;
logic fcarry , fsum ;
logic check = 1'b1;
logic check2 = 1'b1;
logic [2:0] x = 3'b0;
assign dclock = shiftsignal & clock;
shiftregister a
(
    .reset(reset),
    .datain1(ain1),
    .datain2(ain2),
    .datain3(ain3),
    .datain4(ain4),
    .datain5(ain5),
    .datain6(ain6),
    .datain7(ain7),
    .datain8(ain8),
    .dataout1(aout1),
    .dataout2(aout2),
    .dataout3(aout3),
    .dataout4(aout4),
    .dataout5(aout5),
    .dataout6(aout6),
    .dataout7(aout7),
    .dataout8(aout8),
    .clock(clock)
);
shiftregister b
(
    .reset(reset),
    .datain1(bin1),
    .datain2(bin2),
    .datain3(bin3),
    .datain4(bin4),
    .datain5(bin5),
    .datain6(bin6),
    .datain7(bin7),
    .datain8(bin8),
    .dataout1(bout1),
    .dataout2(bout2),
    .dataout3(bout3),
    .dataout4(bout4),
    .dataout5(bout5),
    .dataout6(bout6),
    .dataout7(bout7),
    .dataout8(bout8),
    .clock(clock)
);

dflipflop d (  .clock(clock),   .reset(0),  .datain(fcarry),  .dataout(dout));
fulladder f ( .datain1(aout8), .datain2(bout8), .carryin(dout),
.carryout(fcarry),  .dataout(fsum));


shiftregister out
(
    .reset(reset),
    .datain1(oin1),
    .datain2(oin2),
    .datain3(oin3),
    .datain4(oin4),
    .datain5(oin5),
    .datain6(oin6),
    .datain7(oin7),
    .datain8(oin8),
    .dataout1(dataout1),
    .dataout2(dataout2),
    .dataout3(dataout3),
    .dataout4(dataout4),
    .dataout5(dataout5),
    .dataout6(dataout6),
    .dataout7(dataout7),
    .dataout8(dataout8),
    .clock(clock)
);
always @(posedge clock)
begin
    if(paralelloadsignal)
    begin

     ain1 <= adatain1;
     ain2 <= adatain2;
     ain3 <= adatain3;
     ain4 <= adatain4;
     ain5 <= adatain5;
     ain6 <= adatain6;
     ain7 <= adatain7;
     ain8 <= adatain8;
     bin1 <= bdatain1;
     bin2 <= bdatain2;
     bin3 <= bdatain3;
     bin4 <= bdatain4;
```

```
 bin5 <= bdatain5;
 bin6 <= bdatain6;
 bin7 <= bdatain7;
 bin8 <= bdatain8;
end

else if(shiftsignal & check2 == 1'b1 &check == 1'b1)
begin


oin8 <= oin7;
oin7 <= oin6;
oin6 <= oin5;
oin5 <= oin4;
oin4 <= oin3;
oin3 <= oin2;
oin2 <= oin1;
oin1 <= fsum;

ain8 <= ain7;
ain7 <= ain6;
ain6 <= ain5;
ain5 <= ain4;
ain4 <= ain3;
ain3 <= ain2;
ain2 <= ain1;
ain1 <= 0;

bin8 <= bin7;
bin7 <= bin6;
bin6 <= bin5;
bin5 <= bin4;
bin4 <= bin3;
bin3 <= bin2;
```

```
    bin2 <= bin1;
    bin1 <= 0;

    x <= x +1;
    if(x == 3'b111)
    begin
    check2 <= 1'b0;
    end
    check <= 1'b0;
  end
else if(reset)
  begin
  ain1 = 0;ain2 = 0;ain3 = 0;ain4 = 0;ain5 = 0;ain6 = 0;ain7 = 0;ain8 =
0;
 bin1 = 0;bin2 = 0;bin3 = 0;bin4 = 0;bin5 = 0;bin6 = 0 ;bin7 = 0;bin8 =
0;
 oin1 = 0;oin2 = 0;oin3 = 0;oin4 = 0;oin5 = 0;oin6 = 0 ;oin7 = 0;oin8 =
0;
 check <= 1'b1;
 check2 <= 1'b1;
 x <= 3'b0;
  end
  else if(~shiftsignal)
  begin
  check <= 1'b1;
  end
end

endmodule
```

TestBench Serial Adder

```
`timescale 1ns / 1ps
module serialadder_test;
logic shiftsignal;
logic paralelloadsignal;
logic reset;
logic clock;
logic adatain1;
logic adatain2;
logic adatain3;
logic adatain4;
logic adatain5;
logic adatain6;
logic adatain7;
logic adatain8;
logic bdatain1;
logic bdatain2;
logic bdatain3;
logic bdatain4;
logic bdatain5;
logic bdatain6;
logic bdatain7;
logic bdatain8;
logic dataout1;
logic dataout2;
logic dataout3;
logic dataout4;
logic dataout5;
logic dataout6;
logic dataout7;
logic dataout8;


serialadder add
```

```
(
    .adatain1(adatain1),
    .adatain2(adatain2),
    .adatain3(adatain3),
    .adatain4(adatain4),
    .adatain5(adatain5),
    .adatain6(adatain6),
    .adatain7(adatain7),
    .adatain8(adatain8),
    .bdatain1(bdatain1),
    .bdatain2(bdatain2),
    .bdatain3(bdatain3),
    .bdatain4(bdatain4),
    .bdatain5(bdatain5),
    .bdatain6(bdatain6),
    .bdatain7(bdatain7),
    .bdatain8(bdatain8),
    .dataout1(dataout1),
    .dataout2(dataout2),
    .dataout3(dataout3),
    .dataout4(dataout4),
    .dataout5(dataout5),
    .dataout6(dataout6),
    .dataout7(dataout7),
    .dataout8(dataout8),
    .clock(clock),
    .paralelloadsignal(paralelloadsignal),
    .reset(reset),
    .shiftsignal(shiftsignal)
);
   always #5  clock   = ~clock;
   initial begin
   clock = 0;
   reset = 0;
```

```
paralelloadsignal = 0;
shiftsignal = 0;
adatain1 = 0;
adatain2 = 0;
adatain3 = 0;
adatain4 = 0;
adatain5 = 0;
adatain6 = 0;
adatain7 = 1;
adatain8 = 0;
bdatain1 = 1;
bdatain2 = 0;
bdatain3 = 0;
bdatain4 = 0;
bdatain5 = 0;
bdatain6 = 0;
bdatain7 = 1;
bdatain8 = 1;
#10
paralelloadsignal = 1; #20
paralelloadsignal = 0; #20
shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
  shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
     shiftsignal = 1; #10;
shiftsignal = 0; #10;
reset = 1;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
   shiftsignal = 1; #10;
shiftsignal = 0; #10;
     shiftsignal = 1; #10;
shiftsignal = 0; #10;
   adatain1 = 0;
adatain2 = 0;
adatain3 = 0;
adatain4 = 1;
adatain5 = 1;
adatain6 = 0;
adatain7 = 1;
adatain8 = 0;
bdatain1 = 1;
bdatain2 = 0;
```

```
   bdatain3 = 0;
   bdatain4 = 0;
   bdatain5 = 0;
   bdatain6 = 0;
   bdatain7 = 1;
   bdatain8 = 1;
     reset = 0;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
     shiftsignal = 1; #10;
   shiftsignal = 0; #10;
      shiftsignal = 1; #10;
   shiftsignal = 0; #10;
   #10
  end
endmodule
```