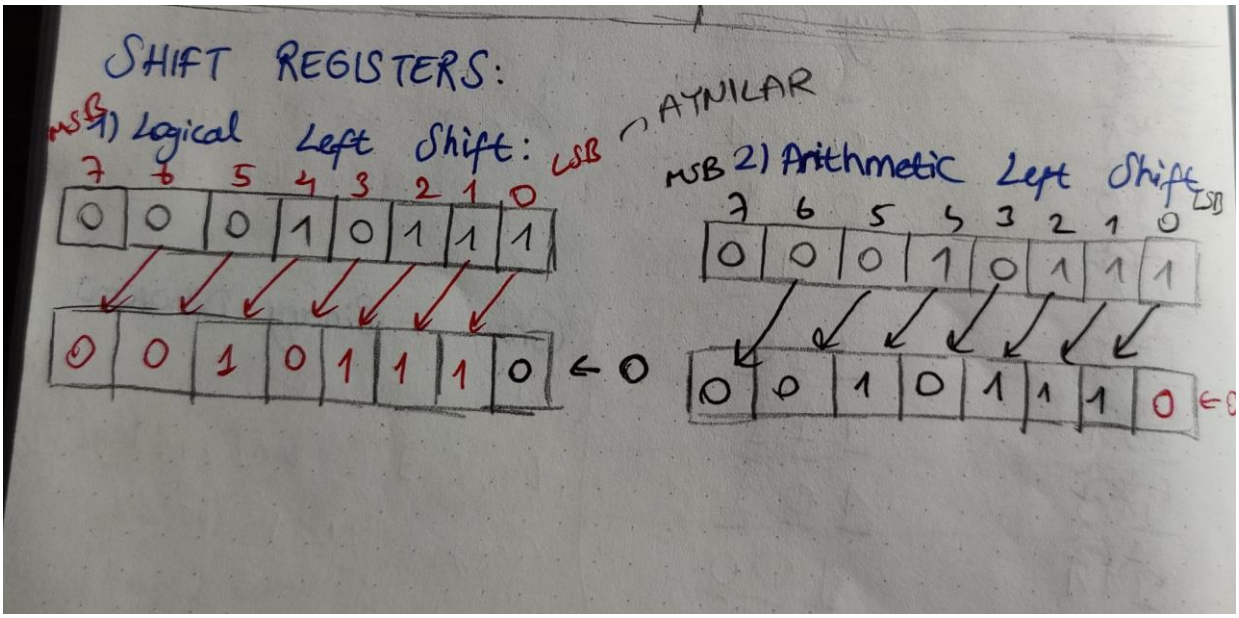
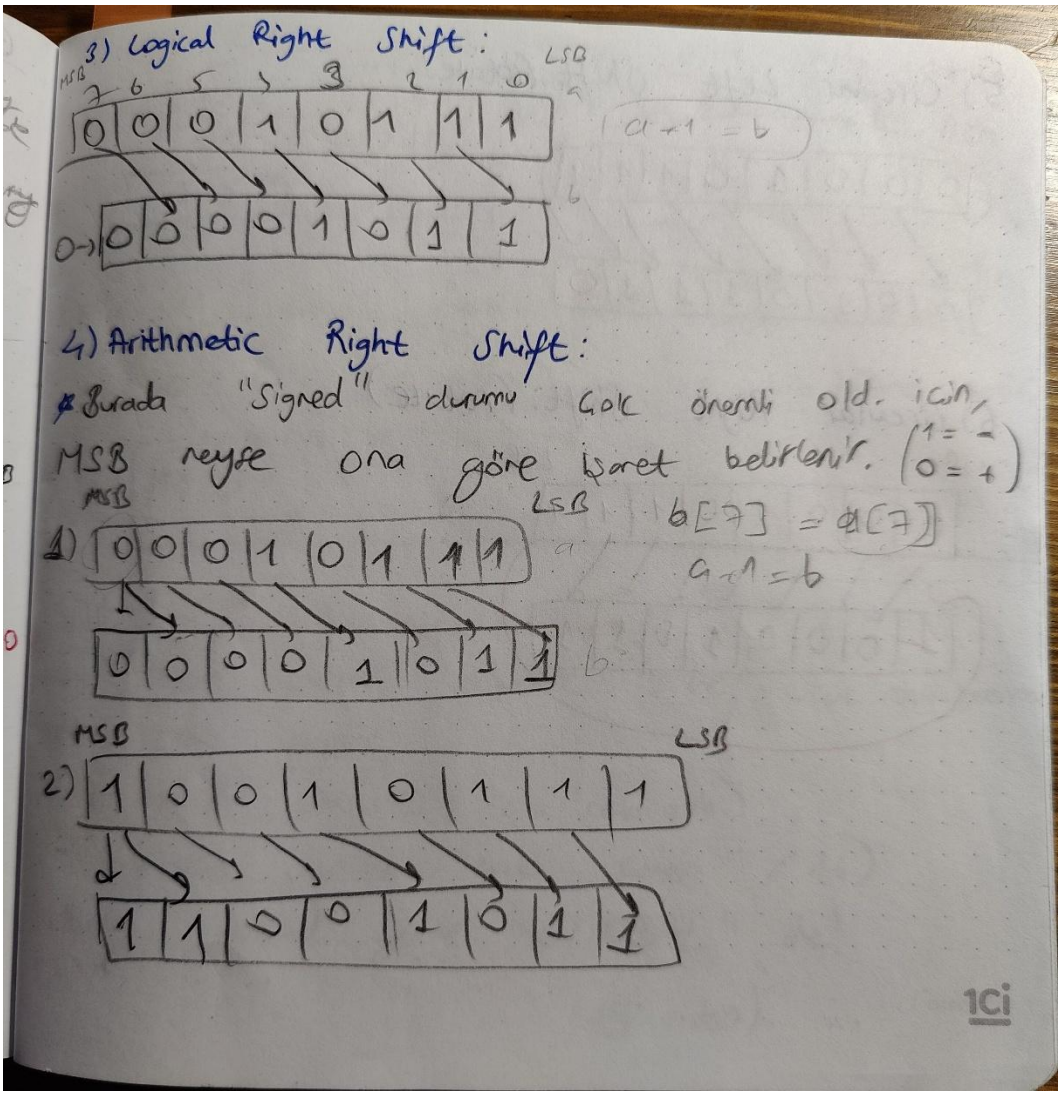
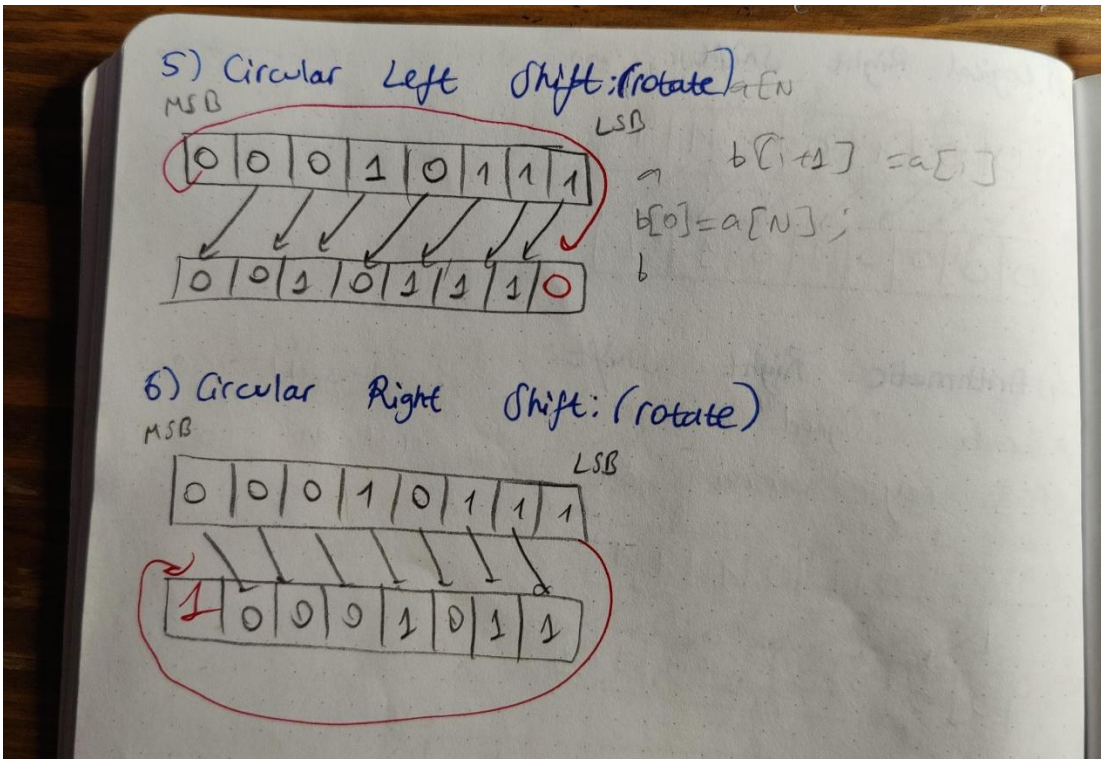
SORU: Verilen “10011101” sayısını aşağıdaki tabloda verilen sıraya göre kaydırma işlemlerini gerçekleştiriniz.

	in	in	in	out
operation	**sel**	**din**	**shift_count [2:0]**	**dout**
noshift	000	10011101	2	10011101
logical left shift	001	10011101	2	01110100
arithmetic left shift	010	10011101	2	01110100
logical right shift	011	10011101	2	00100111
arithmetic right shift	100	10011101	2	11100111
rotate left	101	10011101	2	01110110
rotate right	110	10011101	2	01100111
noshift	111	10011101	2	10011101

Kodu yazmaya geçmeden önce kaydırma operasyonları nedir, bunları anlayalım:




3) Logical Right Shift:
MSB
LSB
7 6 5 4 3 2 1 0
0 0 0 1 0 1 1 1
0→ 0 0 0 0 1 0 1 1
a>>1 = b
4) Arithmetic Right Shift:
* Burada "Signed" durumu çok önemli old. için, MSB neyse ona göre işaret belirlenir. (1 = -, 0 = +)
MSB
LSB
1) 0 0 0 1 0 1 1 1
0 0 0 0 1 0 1 1
b[7] = a[7]
a>>1 = b
MSB
LSB
2) 1 0 0 1 0 1 1 1
1 1 0 0 1 0 1 1

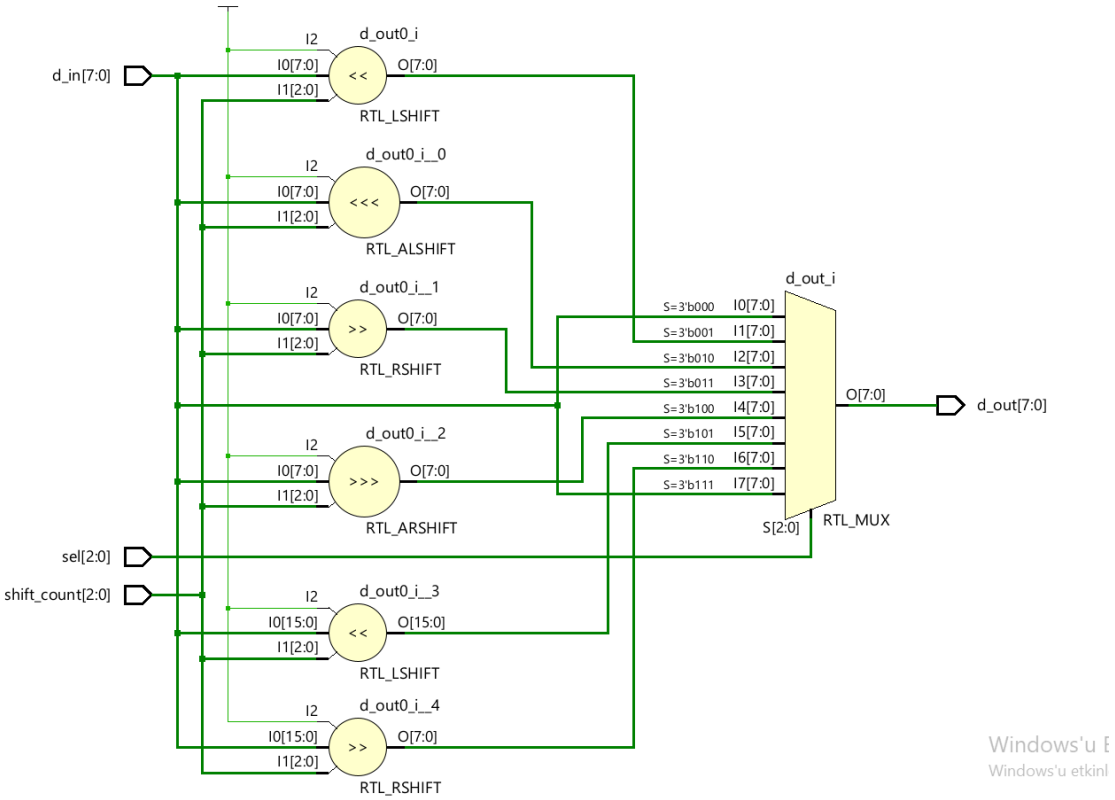

5) Circular Left Shift: (rotate)
MSB
LSB
0 0 0 1 0 1 1 1
0 0 1 0 1 1 1 0
b[i+1] = a[i]
b[0] = a[N];
6) Circular Right Shift: (rotate)
MSB
LSB
0 0 0 1 0 1 1 1
1 0 0 0 1 0 1 1

Design koduna geçecek olursak:

```
module shift_operations(

    input logic signed [7:0] d_in,  // MSB in 1 olursa negatif olmasını
sağlar.
    input logic [2:0] sel,
    input logic [2:0] shift_count,
    output logic signed [7:0] d_out
  );

  logic [15:0] temp;
  logic [15:0] temp_vio;

  always_comb
  begin
    case (sel)
      3'b000:
        d_out = d_in; //no shift
      3'b001:
        d_out = d_in << shift_count; // logical left shift
      3'b010:
        d_out = d_in <<< shift_count; // arithmetic left shift
      3'b011:
        d_out = d_in >> shift_count; // logical right shift
      3'b100:
        d_out = d_in >>> shift_count; // arithmetic right shift
      3'b101: // circular (rotate) left shift
      begin
        temp = {d_in, d_in} << shift_count;
        d_out = temp[15:8];
      end
      3'b110:// circular (rotate) right shift
      begin
        temp = {d_in, d_in} >> shift_count;
        d_out = temp[7:0];
      end
      3'b111:
        d_out = d_in;// reserved
      default:
        d_out = d_in;
    endcase
  end


endmodule

// {} bitleri birleştirmek veya dizileri oluşturmak için kullanılır.
```

```
// temp = {d_in, d_in} demek d_in bitini iki kere arka arkaya koyup 16 bitlik sayı elde etmek demektir.
```

Design çıktımız şu şekilde olacaktır:

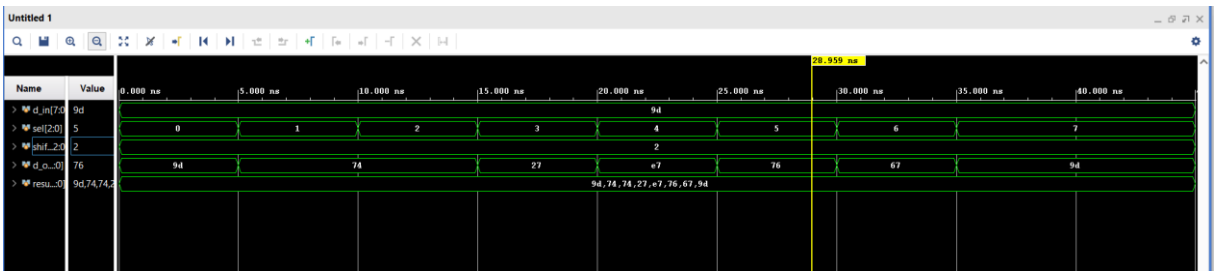


Artık simülasyon kodumuzu yazabiliriz:

```
module tb_shift_operations();

  logic signed [7:0] d_in;
  logic [2:0] sel;
  logic [2:0] shift_count;
  logic signed [7:0] d_out;

  shift_operations shift_operations_Inst
                  (
                    .d_in(d_in),
                    .sel(sel),
                    .shift_count(shift_count),
                    .d_out(d_out)
                  );

  logic [7:0] result_vector [0:7] = // ilk [7:0] dizideki elemanların bit sayısını verir.
        // ikinci [0:7] ise kaç tane eleman olduğunu belirtir.
```

```
        {
          8'b10011101,
          8'b01110100,
          8'b01110100,
          8'b00100111,
          8'b11100111,
          8'b01110110,
          8'b01100111,
          8'b10011101

        };

  initial
  begin
    d_in = 8'b10011101;
    shift_count = 2;

    for(int i = 0; i < 8; i++)
    begin
      sel = i;
      #5;

      assert(d_out == result_vector[i])
            begin
              $display("Test sonucu dogrudur,");
            end

            else
            begin
              $display("Test sonucu yanlistir.");
            end
          end

          #5;
    $stop;
  end

endmodule
```

Simülasyon sonuçlarımız aşağıdaki gibi olacaktır:



```
* /
# run 1000ns
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
Test sonucu dogrudur,
$stop called at time : 45 ns : File "C:/Users/zulal/
INFO: [USF-XSim-96] XSim completed. Design snapshot
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapse
```