

FB-CPU SystemVerilog Testbench

İrem Kalkanlı, Özlem Çalı, Deniz Uzun, Ceyda Uymaz, İrem Bozkurt

Fenerbahçe Üniversitesi
Bilgisayar Mühendisliği
İstanbul, Türkiye

e-mail: {irem.kalkanli , ozlem.cali, deniz.uzun, ceyda.uymaz,irem.bozkurt}@stu.fbu.edu.tr,

Özetçe— FB-CPU işlemcisinin SystemVerilog dili ile otonom kontrolünü yapan bir doğrulama ortamı geliştirdik.

Anahtar Kelimeler — *FPGA, CPU*

Abstract— We have developed a verification environment that performs autonomous control of the FB-CPU processor with the SystemVerilog language.

Keywords — *FPGA, CPU*.

I. Giriş

Bu proje kapsamında dijital tasarım dersinde tamamlanan FB-CPU işlemcisinin SystemVerilog dili ile otonom kontrolünü yapan bir doğrulama ortamı geliştirilecektir.

II. SİSTEM MİMARİSİ

Proje kapsamında 1 araç kullanılacaktır.

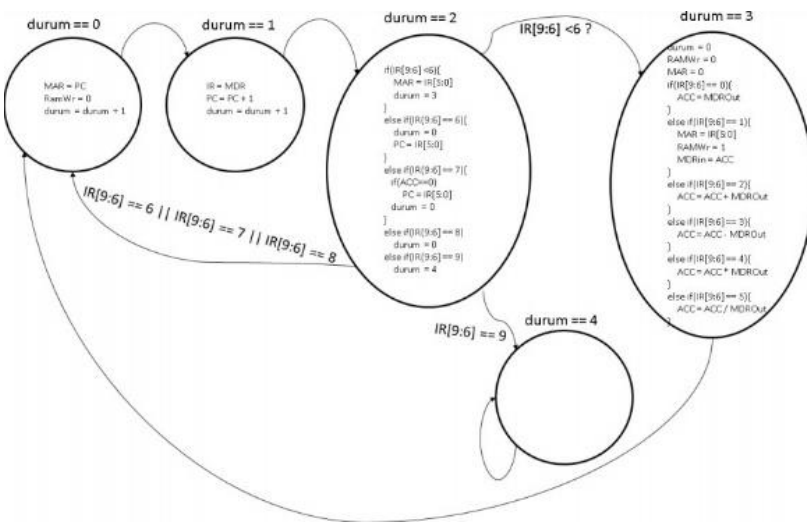
1)Xilinx Vivado Design Suite

Xilinx Vivado Design Suite, FPGA geliştirme kartları üzerinde çalışmalar yapmak için gerekli olan tasarımı oluşturmak için kullanılmaktadır. Verilog, VHDL vb.. donanım tasarım dillerini alarak, FPGA'e konfigüre edilebilecek (Xilinx firması FPGA'leri için .bit uzantılı dosyalar) tasarım dosyasını oluşturur. Vivado Tasarım Aracı, Xilinx'in 7 ve daha yeni jenerasyon FPGA'leri için kullanılabilen bir geliştirme ortamıdır. Bu ortam Xilinx'in sunduğu çeşitli geliştirme ve doğrulama araçlarını barındırır.

Vivado:

- Verilog
- System Verilog
- VHDL Dillerini desteklemektedir. Projede Verilog dili ile tasarımlar yapılacaktır.

III. KULLANILAN YAZILIM

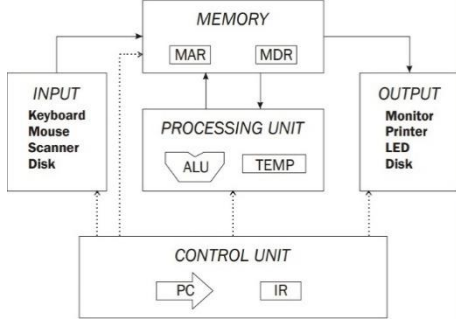


FB-CPU isimli yaptığımız projenin tasarımı şekilde verilmiştir. İstenilen durumlara göre komutların yerine getirilmesi amaçlanmıştır. FB-CPU tasarımı 10 adet komutu yapabilecek şekilde tasarlanmıştır. İşlemci belirtilen komutları yerine getirmek için gerekli durum değerlerini sağlanmalıdır.

ACC yapılan işlem sonucunu tutar.

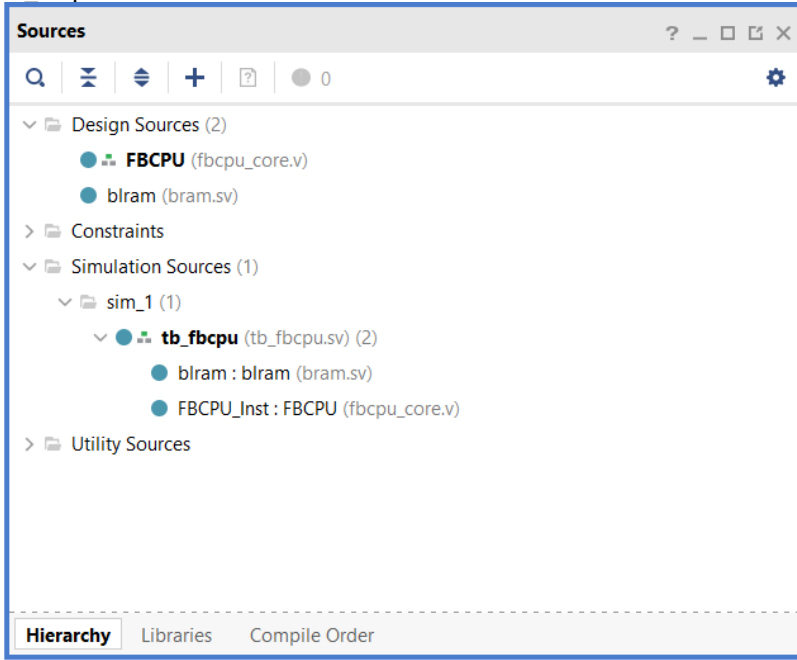
PC şu anki komutu tutar.

IR bir sonraki komutu tutar.



Durumdan gelen değerlere göre gerekli yapı seçilerek işlemler gerçekleştirilmiş olur.

tb_fbcpu.v FBCPU modülünü test etmektedir. FBCPU modülümüz ile birlikte proje dosyalarını Vivado aracında ekledik.

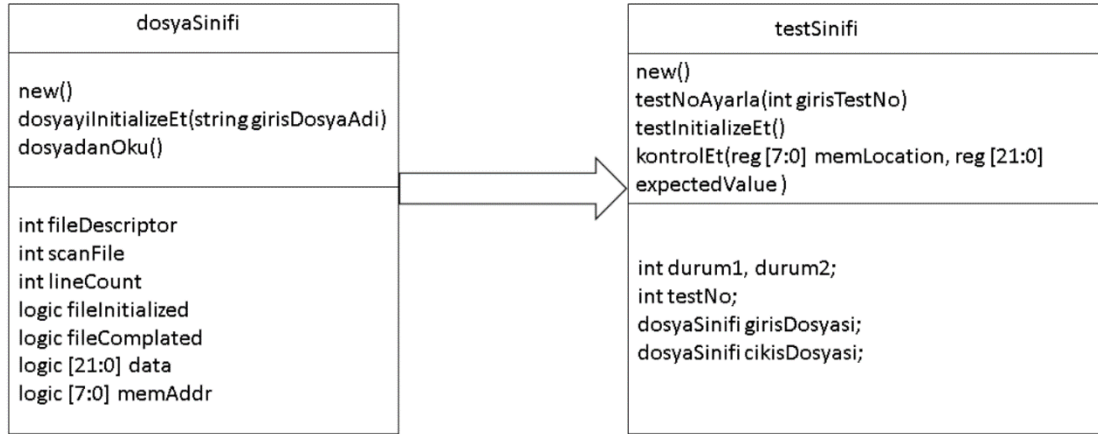


FBCPU ve blram'i design sources tb_fbcpu 'u simulation sources olarak ekledik

tb_fbcpu.v dosyasında iki adet sınıf tanımı yapılmıştır.

Bunlar; • dosyaSinifi • testSinifi

Bu sınıflardan objeler türetilerek initial begin bloğu içerisinde sınıfın çeşitli fonksiyon ve değişkenleri kullanılarak FBCPU test edilmektedir. Proje kapsamında dosyaSinifi ve testSinifi isimli sınıfların boş bırakılan sınıf içerikleri doldurulacaktır. Aşağıda dosyaSinifi ve testSinifi isimli sınıfların içerdiği değişken ve fonksiyonlar verilmektedir. testSinifi isimli sınıf dosyaSinifi sınıfından kalıtım yapılmıştır.



Aşağıda dosyaSinifi isimli sınıfın fonksiyonları açıklanmıştır.

```

class dosyaSinifi;

    integer fileDescriptor    ;
    integer scanFile        ;

    int lineCount;
    logic fileInitialized ;
    logic fileComplated ;
    logic [7:0]memAddr;
    logic [21:0]data;

function new();
    lineCount = 0 ;
    fileInitialized = 0 ;
    fileComplated = 0 ;
    memAddr = 0 ;
    data = 0 ;
endfunction

function int dosyayiInitializeEt( string girisDosyaAdi);
    fileDescriptor = $fopen(girisDosyaAdi, "r");
    if (fileDescriptor ==0) begin
        $display("Dosya Bulunamadı!");
        fileInitialized = 0;
        return 0;
    end
    else begin
        $display("Dosya Bulundu!: %d \n" , girisDosyaAdi);
        fileInitialized = 1;
        return 1;
    end
endfunction

```

Integer olarak tanımladığımız fileDescriptor ve scanFile'ı dosyayı initialize ederken ve okurken kullanıcaz.

lineCount ise dosyadan okuyacağımız başarılı sonuçların sayısını tutacağımız değişkenimiz.

fileInitialized initalize edildiğinde

fileComplated dosya okuma tamamlandığında değiştirdiğimiz değişkenler.

Okumak istediğimiz değer memAddr ye

Okunan adresteki veriyi dataya yazıyoruz.

- new():Constructor'dır. Tüm değişkenleri 0'a atamaktadır.

- dosyayıInitializeEt(string girisDosyaAdi):Kendisine verilen string argümandaki dosya adı ile dosyayı açmaya çalışır. Dosya açıldığında geriye dönen file descriptor'u fileDescriptor değişkenine atar. Dosya başarılı olarak açılmazsa yani fileDescriptor 0'sa fileInitialized değişkeni 0 olur fonksiyon geriye 0 döndürür, diğer durumda yani dosya başarılı olarak açılırsa fileInitialized değişkeni 1 olur fonksiyon geriye 1 döndürür

```
function int dosyadanOku( );
    if (fileInitialized == 1) begin
        if(fileComplated==0) begin
            scanFile = $fscanf(fileDescriptor, "%x %x\n", memAddr, data);
            lineCount++;
            $display("Okunan Satır:%d Okunan memAddr: %d \n",lineCount,memAddr );
            $display("Okunan Satır:%d Okunan data: %d \n",lineCount, data );

            if ($feof(fileDescriptor)) begin
                fileComplated=1;
            end
            return 1;
        end
    else begin
        fileComplated=0;
        lineCount = 0;
        return 0 ;
    end
end
else begin
    $display("Dosya Initialize Edilemedi!" );
end
endfunction
endclass
```

- dosyadanOku(): fileInitialize değişkeni 1 ise yani dosya başarılı bir şekilde açıldıysa , fscanf ile dosyadan 1 satır okuyup bunları memAddr ve data değişkenlerine yazar. Dosyadan her başarılı okunan satır için lineCount değişkenini bir arttırır ve memAddr ve data yazdırılır. Dosyanın sonuna erişildiğinde ise fileComplated değişkenini 1yapar. Başarılı okunmalarda fonksiyon geriye 1 döndürür , diğer durumda ise fileComplated ve lineCount'u 0 yapar geriye 0 döndürür.

Aşağıda testSinifi isimli sınıfın fonksiyonları açıklanmıştır.

```

class testSinifi extends dosyaSinifi;
    int durum1;
    int durum2;
    int testNo;
    dosyaSinifi girisDosyasi;
    dosyaSinifi cikisDosyasi;
    function new();
        super.new();
        testNo = 0;
        girisDosyasi =new;
        cikisDosyasi =new;
    endfunction
    function int testNoAyarla( int girisTestNo );
        testNo = girisTestNo;
        return 0;
    endfunction
    function int testInitializeEt( );
        if(testNo == 0)begin
            girisDosyasi.dosyayiInitializeEt("input1.txt");
            cikisDosyasi.dosyayiInitializeEt("output1.txt");
        end
        else if ( testNo == 1 )begin
            girisDosyasi.dosyayiInitializeEt("input2.txt");
            cikisDosyasi.dosyayiInitializeEt("output2.txt");
        end
        else if ( testNo == 2 ) begin
            girisDosyasi.dosyayiInitializeEt("input3.txt");
            cikisDosyasi.dosyayiInitializeEt("output3.txt");
        end
    endfunction
    function int kontrolEt( reg [7:0] memLocation, reg [21:0] expectedValue );
        durum1 = b1ram.memory[memLocation];
        durum2 = expectedValue;
        if(durum1 == durum2)begin
            $display("FBCPU'nun Ürettiği Sonuç:%d \n Beklenen Değer:%d \n SONUC DOGRU! \n",durum1 , durum2 );
        end
        else begin
            $display("FBCPU'nun Ürettiği Sonuç:%d \n Beklenen Değer:%d \n SONUC YANLIŞ! \n",durum1 , durum2 );
        end
    endfunction
endclass

```

Test sınıfı dosya sınıfından kalıtım yapılarak oluşturulmuştur.

Integer olarak durum1, durum2 testNo tanımladık. Dosya sınıfından girisDosyasi ve cikisdosyasi olmak üzere 2obje oluşturduk.

- new(): fonksiyonu Constructor'dır. testNo değişkenini 0'a atar ve girisDosyasi, cikisDosyasi değişkenlerini new ile initialize eder.
- testNoAyarla(int girisTestNo) fonksiyonu integer Olarak girisTestNo'yu alır testNo değişkenine girisTestNo argümanını yazar.
- testInitializeEt(): Sınıfın içinde bulunan testNo değişkenin değerine göre girisDosyasi.dosyayiInitializeEt fonksiyonu ile input1, input2, input3 txt dosyalarından birini açar (örn girisDosyasi.dosyayiInitializeEt("input1.txt")) ve çıkış dosyalarından output1, output2, output3 dosyalarından birini açar. Dosyaların açılmasında sorun olursa \$finish; komutu ile simülasyonu durdurur.
- kontrolEt(reg [7:0] memLocation, reg [21:0] expectedValue):Kendisine argüman olarak verilen memLocation bilgisini kullanarak, BRAM'deki adres'e bakar. O adresteki içeriğin değeri ile expectedValue değerini karşılaştırır. Aynı ise simülasyon başarılı olarak çıktı verir, değil ise simülasyon hatalı olarak çıktı verir.

```

155     testSınıfı test = new;
156     initial begin
157
158
159         clk = 0;
160         rst = 0;
161
162         for (int i = 0; i<3 ;i = i+1) begin
163             $display("Su anki Test no:  %d\n",i );
164
165             test.testNoAyarla(i);
166             test.testInitializeEt();
167
168
169             while(test.girisDosyasi.dosyadanOku() == 1) begin
170                 blram.memory[test.girisDosyasi.memAddr] = test.girisDosyasi.data;
171                 //@(posedge clk );
172             end
173
174             rst <= #1 1;
175             repeat(10) @(posedge clk );
176             rst <= #1 0;
177             repeat(10000) @(posedge clk );
178
179
180             while (test.cikisDosyasi.dosyadanOku() == 1) begin
181                 $display("TEST SONUCU:" );
182                 test.kontrolEt(test.cikisDosyasi.memAddr, test.cikisDosyasi.data);
183
184
185             end
186             $display("Bitirilen Test->%d\n",i );
187
188         end
189         $display("Simulasyon Tamamlandı!");
190         $finish;
191
192     end
193 endmodule

```

test kısmında test sınıfından test isminde bir obje oluşturulur daha sonra initial begin kod bloğunda for döngüsü içinde tüm testlerde dolaşarak önce test no ayarlaması yapılır. sonra bu testin input ve outputlarını initial edilir. Dosyadan okuma bitince while döğüsü içinde dosyadan okunan memory adres ve datayı blramin içindeki istenilen memAddr istenilen datayı atar. Bir sonraki while döngüsü çıkış dosyası okunduysa test sonucu display edilir. teskontrolet fonksiyonu ile ilk önce blockramin içine atılacak adres verilir. ikincisinde ise beklenen değer veriyor.

daha önce kontrolet dosyasının içinde durum1e girilen memory lokasyonunu blockmemorye yazar. Durum 2de ise beklenen değeri karşılaştırılır durum 1 ve 2 eşitse doğru çalıştığı değilse yanlış çalıştığı anlamına gelir. daha sonra bitirilen testin değeri yazılır. Simülasyon 2. testno'ya kadar devam eder ve tamamlanır.

Sonuca göre doğru çalışan bir FBCPU'dan alınan sonuçlar da doğrudur. Eğer hatalı bir FBCPU konulursa testbench hatalı olduğunu teyit edecektir. hatalı bir CPU ile alınan sonuçların da hatalı olduğu görülür.

Doğru çalışan bir fbcpu'yla test benchimizi çalıştırıyoruz “Sonuc Doğru” yazısı ekrana basılıyor

Su anki Test no: 0

Dosya Bulundu!: input1.txt

Dosya Bulundu!: output1.txt

Okunan Satır: 1 Okunan memAddr: 0
Okunan Satır: 1 Okunan data: 50
Okunan Satır: 2 Okunan memAddr: 1
Okunan Satır: 2 Okunan data: 179
Okunan Satır: 3 Okunan memAddr: 2
Okunan Satır: 3 Okunan data: 116
Okunan Satır: 4 Okunan memAddr: 3
Okunan Satır: 4 Okunan data: 576
Okunan Satır: 5 Okunan memAddr: 50
Okunan Satır: 5 Okunan data: 5
Okunan Satır: 6 Okunan memAddr: 51
Okunan Satır: 6 Okunan data: 10

INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_fbcpu_behav' loaded.

INFO: [USF-XSim-97] XSim simulation ran for 1000ns

launch_simulation: Time (s): cpu = 00:00:14 ; elapsed = 00:00:18 . Memory (MB): peak = 1252.727 ; gain = 0.000

run all

Okunan Satır: 1 Okunan memAddr: 52
Okunan Satır: 1 Okunan data: 15

TEST SONUCU:

FBCPU'nun Ürettiği Sonuç: 15

Beklenen Değer: 15

SONUC DOGRU!

Bitirilen Test-> 0

```
Su anki Test no:          1

Dosya Bulundu!: input2.txt

Dosya Bulundu!: output2.txt

Okunan Satır:          1 Okunan memAddr:    0

Okunan Satır:          1 Okunan data:       50

Okunan Satır:          2 Okunan memAddr:    1

Okunan Satır:          2 Okunan data:       307

Okunan Satır:          3 Okunan memAddr:    2

Okunan Satır:          3 Okunan data:       116

Okunan Satır:          4 Okunan memAddr:    3

Okunan Satır:          4 Okunan data:       576

Okunan Satır:          5 Okunan memAddr:    50

Okunan Satır:          5 Okunan data:        5

Okunan Satır:          6 Okunan memAddr:    51

Okunan Satır:          6 Okunan data:        10

Okunan Satır:          1 Okunan memAddr:    52

Okunan Satır:          1 Okunan data:        50

TEST SONUCU:
FBCPU'nun Ürettiği Sonuç:          50
Beklenen Değer:          50
SONUC DOGRU!

Bitirilen Test->          1
```

Testbenchimizin doğruluğunu anlamak için fbcpu umuzu bozuyoruz. == olan bazı kısımları !=yapıyoruz böylelikle fbcpu bozulduğundan hatalı sonuç almayı bekliyoruz.


```
Okunan Satır: 16 Okunan data: 0
Okunan Satır: 17 Okunan memAddr: 50
Okunan Satır: 17 Okunan data: 5
Okunan Satır: 18 Okunan memAddr: 51
Okunan Satır: 18 Okunan data: 10
Okunan Satır: 1 Okunan memAddr: 52
Okunan Satır: 1 Okunan data: 50

TEST SONUCU:
FBCPU'nun Ürettiği Sonuç: 974
Beklenen Değer: 50
SONUC YANLIŞ!

Bitirilen Test-> 2

Simulasyon Tamamlandı!
$finish called at time : 300295 ns : File "C:/Users/recep/fbcpu_testbench

Okunan Satır: 4 Okunan data: 576
Okunan Satır: 5 Okunan memAddr: 50
Okunan Satır: 5 Okunan data: 5
Okunan Satır: 6 Okunan memAddr: 51
Okunan Satır: 6 Okunan data: 10

INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_fbcpu_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
run all
Okunan Satır: 1 Okunan memAddr: 52
Okunan Satır: 1 Okunan data: 15

TEST SONUCU:
FBCPU'nun Ürettiği Sonuç: 1019
Beklenen Değer: 15
SONUC YANLIŞ!

Bitirilen Test-> 0

Su anki Test no: 1

Dosya Bulundu! input2.txt

3: begin//3.durum
durumNext=0;
RAMWr=0;
MAR=0;
if (IR[9:6]==0)begin
ACCNext=MDROut;
end else if (IR[9:6]==1)begin
MAR=IR[5:0];
RAMWr=1;
MDRIn=ACC;
end else if (IR[9:6]==2)begin //yanlış kod
ACCNext=ACC+MDROut;
end else if (IR[9:6]==3)begin //yanlış kod
ACCNext=ACC-MDROut;
end else if (IR[9:6]==4)begin //yanlış kod
ACCNext=ACC*MDROut;
end else if (IR[9:6]==5)begin //yanlış kod
ACCNext=ACC/MDROut;
end
end

4: begin//4.durum
durumNext=4;
end
endcase
end
```

Sol taraflarda da gördüğünüz gibi hatalı bi fbcpu da test benchimiz sonucun hatalı olduğunu ekrana yazdırıyor.

IV. SONUÇLAR

Bu proje kapsamında dijital tasarım dersinde tamamlanan FB-CPU işlemcisinin SystemVerilog dili ile otonom kontrolünü yapan bir doğrulama ortamı geliştirdik. Sonuca göre doğru çalışan bir FBCPU'dan alınan sonuçlar da doğrudur. Eğer hatalı bir FBCPU konulursa testbench hatalı olduğunu teyit etmektedir.

PROJE EKİBİ

İREM KALKANLI (PROJE EKİP SORUMLUSU):

OKUL NUMARASI:190301007

DOĞUM TARİHİ:15.01.2000

DOĞUM YERİ: İSTANBUL

MEZUN OLDUĞU LİSE: ATAŞEHİR 3 DOĞA KOLEJİ

DENİZ UZUN:

OKUL NUMARASI:190301015

DOĞUM TARİHİ:08.04.2001

DOĞUM YERİ: İSTANBUL

MEZUN OLDUĐU LİSE: KAVACIK UĐUR ANADOLU LİSESİ

ÖZLEM ÇALI:

OKUL NUMARASI:190301002

DOĐUM TARİHİ:19.05.2000

DOĐUM YERİ: HATAY

MEZUN OLDUĐU LİSE: NECMİ ASFUROĐLU ANADOLU LİSESİ

İREM BOZKURT:

OKUL NUMARASI:190302010

DOĐUM TARİHİ:04.11.1998

DOĐUM YERİ: ADIYAMAN

MEZUN OLDUĐU LİSE: ÖZEL BİL KOLEJİ FEN LİSESİ

CEYDA UYMAZ:

OKUL NUMARASI:200301503

DOĐUM TARİHİ:26.08.2000

DOĐUM YERİ: İSTANBUL

MEZUN OLDUĐU LİSE: CELAL ARAS ANADOLU LİSESİ

REFERANS DOSYALAR

<https://youtu.be/xZI9dXtoi4g>

KAYNAKLAR

- [1] Levent, Vecdi Emre (2020) “Donanım Doğrulama Metodları (DDM): Sınıflar”, *Electronic Circuits-Ders Notları*.
- [2] Levent, Vecdi Emre (2020) “DDM: Kalıtım”, *Electronic Circuits -Ders Notları*.
- [3] Levent, Vecdi Emre (2020) “FB-CPU System Verilog TB”, *Electronic Circuits-Ders Notları*.
- [4] Levent, Vecdi Emre (2020) “DDM: Arayüzler”, *Electronic Circuits-Ders Notları*.
- [5] Levent, Vecdi Emre (2020) “Hata Ayıklama”, *Electronic Circuits-Ders Notları*.
- [6] Levent, Vecdi Emre (2020) “FB-CPU RTL Tasarım”, *Mantıksal Sistem Tasarımı-Ders Notları*.