AvionCPU

**Amaç**

Bu projede, teknik dokümanda sunulan komut seti, durum makinesi (state machine), saklayıcı yapıları ve test kodları doğrultusunda eksik bırakılan top.v dosyası tamamlanmıştır. Tamamlanan tasarım, **AvionChip simülatörü** kullanılarak test edilmiş, **dalgaformu (waveform)** çıktıları üzerinden doğrulama işlemleri gerçekleştirilmiştir. Bu süreçte hedeflenen, işlemci mimarilerini anlamak, bu tür bir mimarinin **Verilog HDL** diliyle nasıl yazıldığını öğrenmek ve simülasyon yoluyla tasarım doğrulama pratiği kazanmaktır.

**Sistem Yapısı**

AvionCPU, temel bir **RAM** ve **CPU** yapısından oluşan sade bir sistem mimarisi sunmaktadır. Tasarım, yalnızca bir adet top.v dosyasıyla temsil edilmekte ve **Verilog** dilinde yazılmıştır. Bu işlemci, yazılan kodun simüle edilebilmesini ve dalgaform çıktılarının analiz edilmesini sağlayan **AvionChip** simülatörü ile çalışacak şekilde yapılandırılmıştır. Geliştirilen işlemci mimarisinde, **Von Neumann mimarisi** esas alınmıştır. Bu mimaride, **veriler ve komutlar aynı bellek alanında** depolanmaktadır. Sistem, bu verilerin ve komutların tutulduğu **1 adet RAM** ve işlemleri gerçekleştiren **1 adet CPU** olmak üzere iki temel bileşenden oluşmaktadır.

CPU, toplamda **9 farklı komutu** yerine getirebilecek kapasitededir. Bu komutlar aşağıdaki gibidir:

* **Veri yükleme (LOAD)**
* **Veri saklama (STORE)**
* **Toplama (ADD)**
* **Çıkarma (SUB)**
* **Çarpma (MUL)**
* **Şartsız atlama (JMP)**
* **Koşullu atlama (JZ - Zero kontrolüyle)**
* **Programı durdurma (HALT)**
* **Boşta bekleme (NOP)**

Bu komutlar, **4 bitlik bir işlem kodu (opcode)** ile temsil edilmekte olup, her komutun devamında **6 bitlik adres veya sabit değer alanı** yer almaktadır. Bu sayede, her komut toplamda **10 bitlik bir formata** sahiptir.

Sistem mimarisi içerisinde, toplamda **8 adet saklayıcı (register)** bulunmaktadır. Bunlar iki ana grupta toplanmıştır:

**Durum Saklayıcıları:**

* state → Mevcut durum bilgisini tutar (state machine)
* PC → Program Sayacı (Program Counter)
* IR → Komut Saklayıcı (Instruction Register)
* ACC → Akümülatör (Accumulator)

**Sinyal Saklayıcıları:**

* MAR → Bellek Adres Saklayıcısı (Memory Address Register)
* MDRIn → Belleğe Yazılacak Veri (Memory Data Register In)
* RAMWr → Belleğe yazma sinyali (Write enable)
* MDROut → Bellekten okunan veri (Memory Data Register Out)

Bu saklayıcılar sistemin şablon kodu içinde tanımlanmıştır ve projeye **ek saklayıcı** eklenmemiştir.

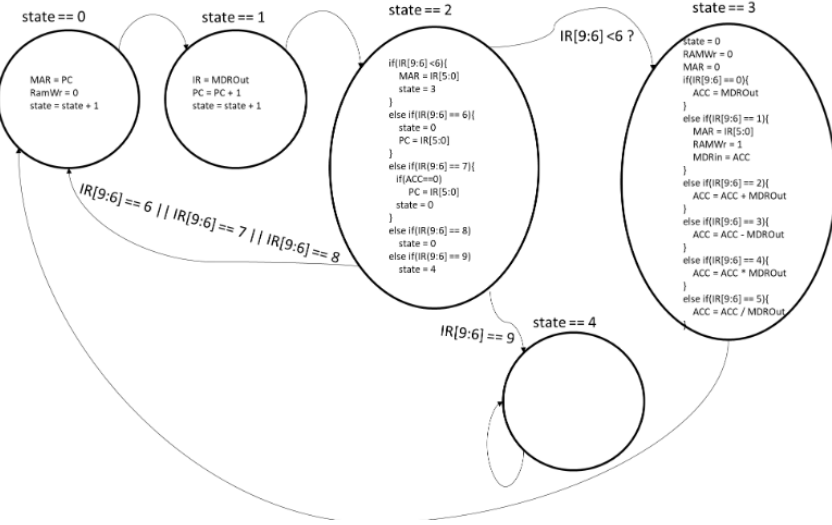
Şablon kod üç ana modülden oluşmaktadır:

* **blram modülü** → Bellek (RAM) davranışını simüle eder.
* **avion\_cpu modülü** → CPU'nun kontrol birimini ve komut yürütme mantığını içerir.
* **tb\_avion\_cpu testbench modülü** → Sistemin doğru çalışıp çalışmadığını kontrol etmek için hazırlanmış test ortamıdır.

avion\_cpu modülünde yer alan **case yapısı**, teknik dokümanda **Şekil 1** olarak verilen durum makinasına (state machine) göre yapılandırılmıştır. Bu durum makinası; komutun alınması, ayrıştırılması, yürütülmesi ve sonuçların yazılması gibi adımları belirli bir sırada yönetmektedir.

**SİMÜLASYON VE TEST**

Tamamlanan top.v dosyası, **AvionChip FPGA simülatörü** aracılığıyla test edilmiştir. Simülasyon sırasında kullanılan **dalgaformları (waveform)** sayesinde işlemcinin çalışma doğruluğu analiz edilmiştir.



**Şekil 1**. AvionCPU durum makinesi

**TESTLER**

Tamamlanan avion\_cpu modülünün doğrulaması, teknik dokümanda belirtilen **üç adet test senaryosu** kullanılarak gerçekleştirilmiştir. Bu testler, simülasyon ortamında tb\_avion\_cpu ve blram modülleri aracılığıyla çalıştırılmıştır. Test senaryolarının her biri, işlemcinin farklı işlem kombinasyonlarını doğru şekilde yerine getirip getirmediğini kontrol etmeye yöneliktir.

**Test Senaryosu-1 Toplama ve Belleğe Yazma**

Bu senaryoda aşağıdaki adımlar gerçekleştirilmiştir:

1. **LOD (Load):** Bellekten bir sayı akümülatöre (ACC) yüklenmiştir.
2. **ADD (Add):** ACC içeriği, başka bir sayı ile toplanmıştır.
3. **STO (Store):** Toplam sonucu tekrar belleğe kaydedilmiştir.
4. **HLT (Halt):** Program sonlandırılmış ve CPU boş duruma geçirilmiştir.

Bu test ile temel aritmetik işlemler ve bellek etkileşimi doğrulanmıştır.

**Test Senaryosu –2 Çarpma ve Belleğe Yazma**

İkinci senaryo birinciye benzer şekilde ilerler, ancak toplama yerine çarpma işlemi gerçekleştirilir:

1. **LOD (Load):** Bellekten bir sayı akümülatöre alınır.
2. **MUL (Multiply):** ACC'deki sayı, başka bir sayı ile çarpılır.
3. **STO (Store):** Çarpım sonucu belleğe kaydedilir.
4. **HLT (Halt):** Program sonlandırılır.

Bu test ile işlemcinin çarpma işlemini doğru şekilde uygulayıp uygulamadığı kontrol edilmiştir.

**Test Senaryosu – 3 Döngü ile Toplama**

Üçüncü test senaryosu, işlemcinin kontrol akışı komutlarını (özellikle **JMP**) ve döngü yapısını doğru işleyip işlemediğini ölçmektedir:

1. **JMP (Jump):** Kodun belirli bir adresine atlayarak döngü oluşturulmuştur.
2. Döngü içinde toplam işlemi yapılmıştır.
3. Döngü, belirli bir koşul sağlanana kadar **10 kez** çalışacak şekilde yapılandırılmıştır.

Bu senaryo ile işlemcinin **dallanma ve tekrarlama** kontrolü test edilmiştir.