**Laboratuvar Çalışması 0x6**

Komut Parçalama

Bu labın amacı

* Komut parçakalama devresi tasarlama
* Tasarlanan devreleri gerçekleyip test edebilmek.

### Problem 1 - Komut ayırıcı

Bu problemde 32-bit gönderilen bir komutu, aşağıda verilen isterlere uygun olarak parçalayıp, gerekli çıkışları üreteceksiniz. Bütün devre **kombinasyonel lojik** olarak çalışacaktır. (clk ve reset pinleri kullanılmayacaktır). Gelen komutlar dört farklı tipte olabilir.

Sizin yapmanız gereken, öncelikle **opcode** bölümünü kontrol edeceksiniz. Tablo 6 te verilen değerlerden bir tanesi gelmiş ise ona göre Tablo 1,2,3 veya 4 e göre geri kalan bitleri ayıracaksınız. Eğer Tablo 6 te verilen değerlerden farklı bir değer gelirse **hata** biti 1 olacak (default olarak 0)

* Eğer **R** tipi bir komut geldiyse **rs1**, **rs2** ve **rd** değerlerine direkt atama yapacaksınız, **aluop** çıkışına da fonksiyonda gösterilen formüle göre oluşturup atama yapacaksınız. (bit30, bit14, bit13, bit12). **imm** çıkışı 0 olacak.
* Eğer **I** tipi bir komut geldiyse **rs1** ve **rd** değerlerine direkt atama yapacaksınız, **aluop** çıkışına da fonksiyon bitlerini soldan 0 ekleyerek atama yapacaksınız. (0, bit14, bit13, bit12). **imm** çıkışı **imm12** olacak. diğer çıkışlar 0 olacak.
* Eğer **U** tipi bir komut geldiyse **rs1** ve **rd** değerlerine direkt atama yapacaksınız. **imm** çıkışı **imm20** olacak. diğer çıkışlar 0 olacak.
* Eğer **B** tipi bir komut geldiyse **rs1**, **rs2** değerlerine direkt atama yapacaksınız, **aluop** çıkışına da fonksiyonda gösterilen formüle göre oluşturup atama yapacaksınız. (0, bit14, bit13, bit12). **imm** çıkışı LSB si 0 olacak şekilde 13 bitlik **imm13** olacak.

**Not:** rs1\_data ve rs2\_data değerlerine her zaman 0 atayın.

A. Bu problemin testbench inde, her bir komut tipi için birkaç bit vektörü oluşturup test edin.

B. Kapsamlı bir test vektörü oluşturun ve bu testvektörünü kullanarak test edin

|  |  |
| --- | --- |
| bit 31 - 25 (30 hariç) | 30 hariç kullanılmıyor. Hepsi 0 olması lazım |
| bit 24 - 20 | source register 2 numarası |
| bit 19 - 15 | source register 1 numarası |
| bit30 + bit 14 - 12 | fonksiyon |
| bit 11 - bit 7 | destination register numarası |
| bit 6 - 0 | Tip R operasyon kodu (opcode) |

**Tablo 1 - R Tipi bit değerleri**

|  |  |
| --- | --- |
| bit 31 - 20 | 12-bitlik immediate sayı (imm12) |
| bit 19 - 15 | source register 1 numarası |
| bit 14 - 12 | fonksiyon |
| bit 11 - bit 7 | destination register numarası |
| bit 6 - 0 | Tip I operasyon kodu (opcode) |

**Tablo 2 - I Tipi bit değerleri**

|  |  |
| --- | --- |
| bit 31 - 12 | 20-bitlik immediate sayı (imm20) |
| bit 11 - bit 7 | destination register numarası |
| bit 6 - 0 | Tip U operasyon kodu (opcode) |

**Tablo 3 - U Tipi bit değerleri**

|  |  |
| --- | --- |
| bit 31 - 25 | 13-bitlik immediate ın üst bitleri (imm13[12:6]) |
| bit 24 - 20 | source register 2 numarası |
| bit 19 - 15 | source register 1 numarası |
| bit 14 - 12 | fonksiyon |
| bit 11 - bit 7 | 13-bitlik immediate ın alt bitleri (imm13[5:1]) |
| bit 6 - 0 | Tip B operasyon kodu (opcode) |

**Tablo 4 - B Tipi bit değerleri**

Yukarıdaki tabloların bir başka gösterimi de aşağıdaki şekilde verilmiştir.

31 25 24 20 19 15 14 12 11 7 6 0

+--------------+---------+---------+--------+-------------+----------+

| funct[4] | rs2 | rs1 | funct | rd | opcode | R tipi

+------------------------+---------+--------+-------------+----------+

| imm[11:0] | rs1 | funct | rd | opcode | I tipi

+-------------------------------------------+-------------+----------+

| imm[19:0] | rd | opcode | U tipi

+-------------------------------------------+-------------+----------+

| imm[12:6] | rs2 | rs1 | funct | imm[5:1] | opcode | B tipi

+-------------------------------------------+-------------+----------+

**Tablo 5 - Komut tipleri bit değerleri genel tablo**

Operasyon Kodları

|  |  |
| --- | --- |
| **Operasyon tipi** | **Operasyon kodu (opcode)** |
| R Tipi | 0000001 |
| I Tipi | 0000011 |
| U Tipi | 0000111 |
| B Tipi | 0001111 |

**Tablo 6 - Operasyon kodları**

module p1 (

input logic clk, reset, // problem2 icin gerekli olacaklar

input logic [31:0] komut,

output logic [6:0] opcode,

output logic [3:0] aluop,

output logic [4:0] rs1,

output logic [4:0] rs2,

output logic [31:0] rs1\_data,

output logic [31:0] rs2\_data,

output logic [4:0] rd,

output logic [31:0] imm,

output logic hata

);