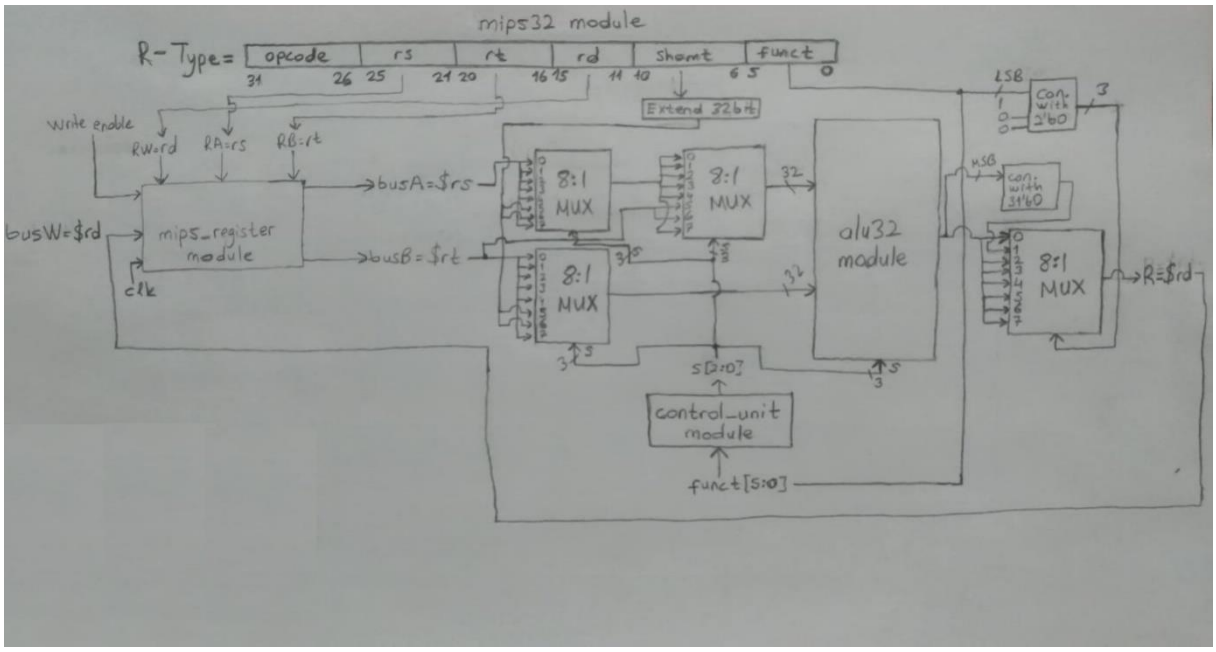
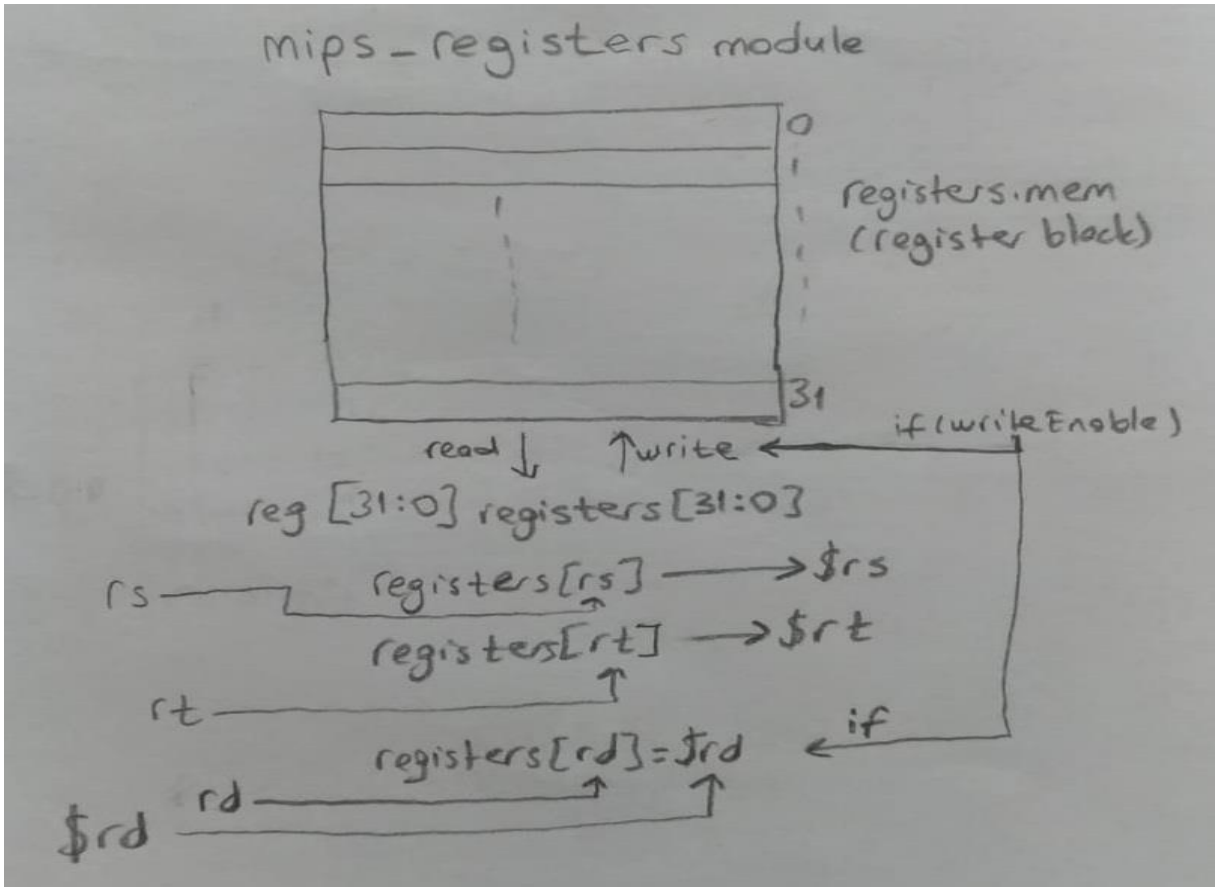
## Schematic Designs For All Modules

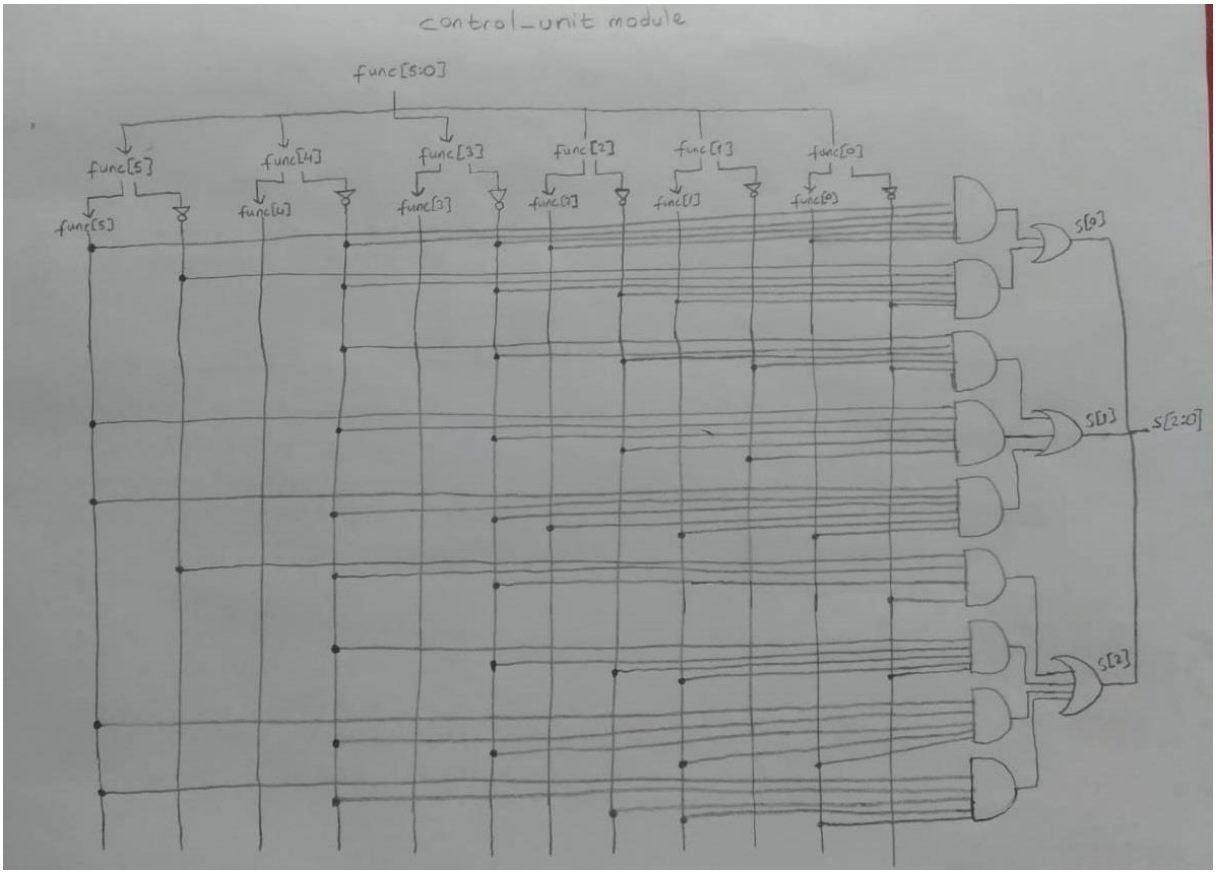
### mips32 schematic design




### mips_registers schematic design



## control unit schematic design




## Verilog Modules and Their Descriptions mips32

Top-level entitydir. Input olarak 32 bitlik instruction alır. Bu instructionu çalıştırır ve ouput olarak rd registerinin contentini çıkarır.

Input olarak gelen instructionu rs,rt,rd,shamt ve funct olarak parçalara ayırır. Bu parçalardan rs,rt ve rt adreslerini **mips_registers** modülüne input olarak yollar. funct parçasını ise **control_unit** modülüne input olarak yollar. **mips_registers** modülünden de rs ve rt registerlerinin contentlerini alır.

Ardından rs contenitini 8 e 1 lik bir multiplexer a input olarak gönderir.Muxtan gelen çıktı ya rs in contenti yada 32bitlik extend edilmiş shmt nin contentidir. Eğer instructionu umuz sll yada srl ise gelen çıktı 32bitlik extend edilmiş shmt dir değilse rs contentidir.Sonra tekrar bir mux daha kullanıldı. Bu muxada bir öceki mux un outputu ve rt contenti gönderildi. Eğer instruction sll yada srl ise mux çıktısı rt contentidir.Değilse rs contentidir ve bu çıktı alunın ilk girişine yollandı.

**mips_register** modülünden okunan rt contenti ve extend edilmiş shamt muxa yollanır ve eğer gelen instruction sll yada srl ise çıktı 32 bitlik extend edilmiş shamt dir ve bu çıktıda **alunun** ikinci girişine yollandı.

**Alu** çıktısına mux koyuldu bir girişi **alu** çıktısı diğeri ise **alu** çıktısının most siginificant biti ile 31 tane 0 ın birleştirilmiş halidir. Eğer instruction sltu ise çıktımız alu çıktısının most siginificant biti ile 31 tane 0 ın birleştirilmiş halidir. Değilse direkt olarak alu çıktısının kendisidir.Bu çıktıda **mips_register** modulune rd registerinin contentine yazılmak üzere yollanır.Write enable 1 iken rd ye yazılır.

## mips_registers

İki boyutlu array a **registers.mem** dosyasından okuma yapılır. Array doldurulur. Input olarak gelen rs ve rt adresleri kullanılarak, arrayın içinde rs ve rt adreslerine göre erişim yapılır ve contentleri okunur ve bu contentler output olarak verilir.Eğer write signal 1 ise gelen rd adresine registers arrayde rd contenti atılır ve register arrayinin registers.mem dosyasına yazımı gerçekleştirilir. 0. Registere yazmama da kontrol edilmiştir.

## control unit

Gelen 6 bitlik function code unu kullarak 3 bitlik alu select biti üretir. And or ve not kullanılmıştır. Gerekli sadeleştirmeler truth tabledan hesaplanarak yapılmıştır.

## _32bit_shift_right_logical

Gelen 32 bitlik A ve B sayısını kullanarak A yı B kadar sağa shift eder ve her shift ettiğinde most significant bit e 0 verilir. Bir önceki assigment ta 0 yerine msb yi veriyorduk şimdi 0 veriyoruz. Tek değişiklik budur.

# Modelsim Simulation Results

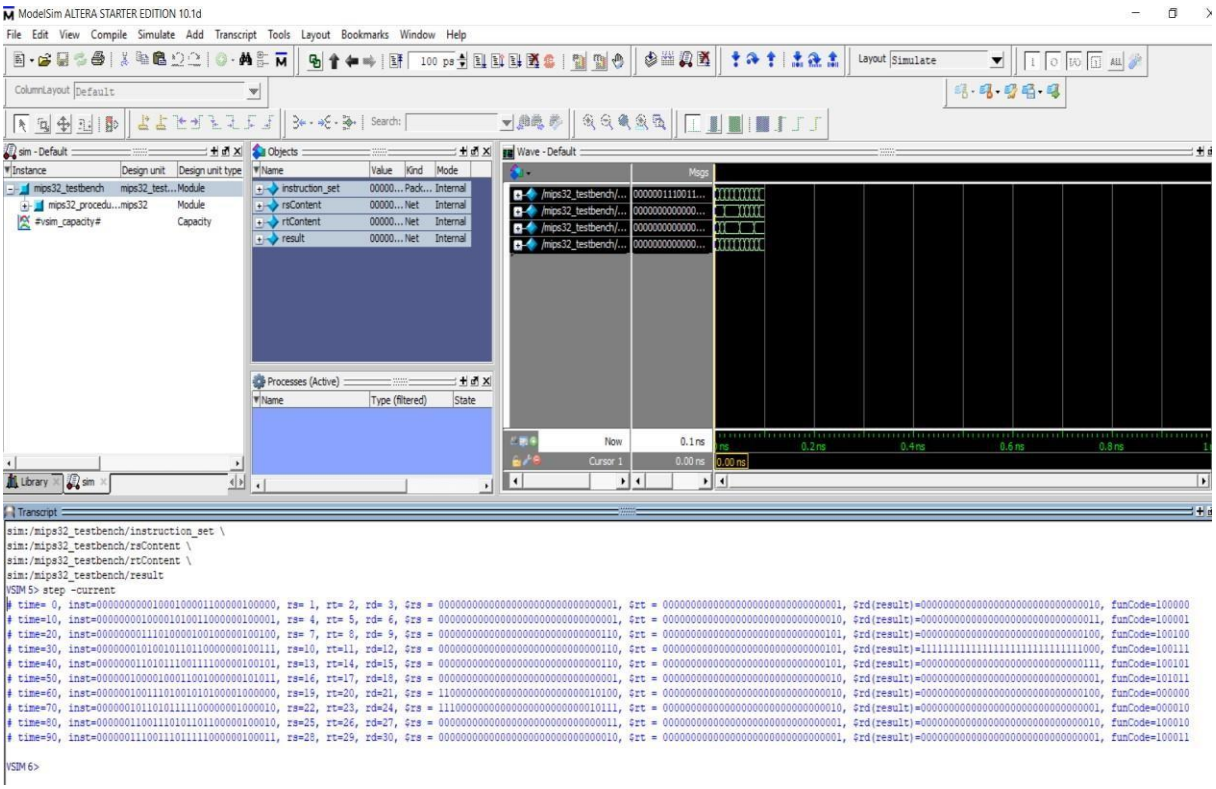
## 1.

registers.mem (Before)

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000000
11000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000000
11100000000000000000000000010111
00000000000000000000000000000010
00000000000000000000000000000000
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000000
```

registers.mem (After)

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000100
00000000000000000000000000000110
00000000000000000000000000000101
11111111111111111111111111111000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000111
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000001
11000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000100
11100000000000000000000000010111
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000000
```



## 2.

register.mem(before)

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000000
00000000000000000000000000000011
00000000000000000000000000000011
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000000
11000000000000000000000000010100
00000000000000000000000000001110
00000000000000000000000000000000
11100000000000000000000000010111
00000000000000000000000011000000
00000000000000000000000000000000
00000000000000000000000000000111
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000111
00000000000000000000000000000011
00000000000000000000000000000000
00000000000000000000000000000000
```

register.mem(after)

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000011
00000000000000000000000000000011
00000000000000000000000000000110
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000100
00000000000000000000000000000110
00000000000000000000000000000101
11111111111111111111111111111000
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000111
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000000
11000000000000000000000000010100
00000000000000000000000000001110
00000000000000000000000001110000
11100000000000000000000000010111
00000000000000000000000011000000
00000000000000000000000000110000
00000000000000000000000000000111
00000000000000000000000000000001
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000000
```