CSE 331 - Computer Organization Project 2's Report

FÜRKAN YILDIZ

141044031

Ekstra modül oluşturulmadı, verilen modüller kullanıldı.

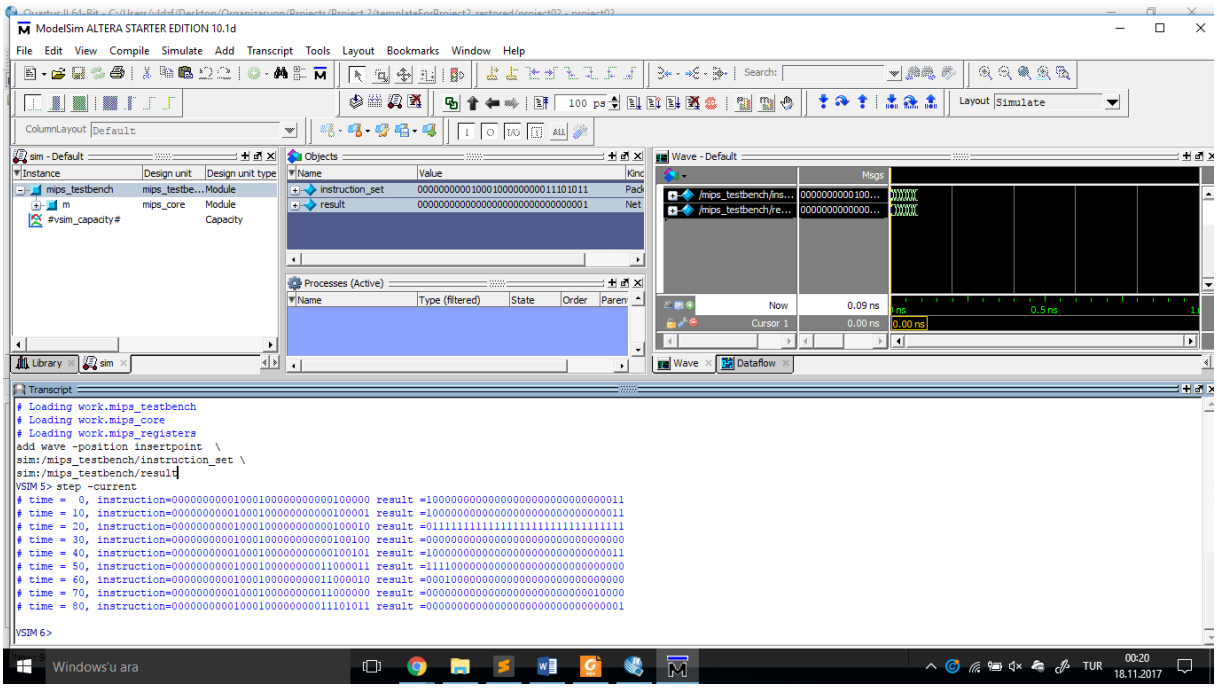
Çalışabilmesi için, similation/modelsim klasörünün altında “registers.mem” adında register contentlerini içeren dosya bulunmalı.

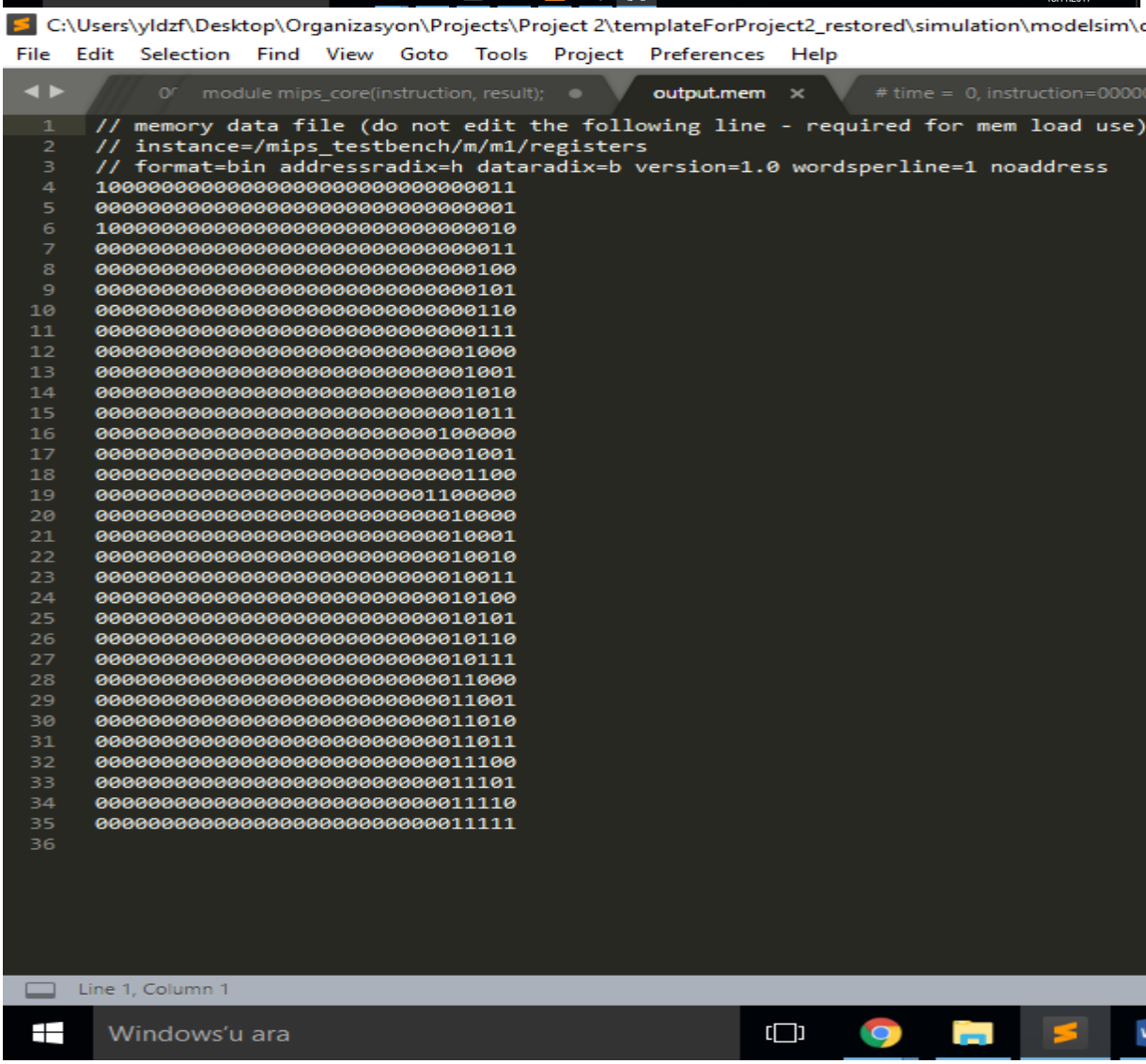
mips_core modülü:

İnput olarak 32 bitlik instruction alan ve 32 bitlik result return eden bir top modül istendiğinden, verilen “mips_core” modülünü kullandım. Output olan result’a atama yapabilmek için (always blok’ta) result’un tipini “output reg” olarak tanımladım. Daha sonra gelen instruction’ı parçalama aşamasına geçtim, function field,shamt,rd,rs,rt adress ve opcode olarak gerekli şekilde ayırdım(bit sıralarına bakarak, son 6 bit function code, … , ilk 6 bit opcode şeklinde). Bu ayırmada assigmenti kullanabilmek için instructiondan çektiğim function field,shamt,rd,rs,rt adress ve opcode’yi wire tipinde tanımladım. ALU sadece R type sistemlerde çalışacağından opcode 000000 olacak her zaman, funciton field ise, yapılacak operasyonu belirtecek. Bu ödevde r-typenin 9 tipini çalıştırabilmemiz isteniyor, bu 9 tipin funciton fieldlerini wire ye kaydettim (add = 100000 gibi). Böylece gelen instroductionun function fieldi hangisi ise, yapılacak operasyonda o olacaktır. Ardından ise register modülünü tanımladım ve always bloğu oluşturdum çünkü, register’in parametrelerinin değiştiği zaman o modülün çağırılmasını istedim. Registerin parametrelerinin değişince, register modülünün çağırılmasını istememin nedeni ise, register bloğunu birden fazla çağırmaya ihtiyaç duymamızdan kaynaklanmaktadır. Birinci çağırımda, register modülü rs ve rt nin adreslerini kullanarak contentlerine erişecek ve output olarak bu contentleri verilecek, mipc_core modülü ise bu contentleri kaydedecek ve funciton fielde bakarak (funciton field ile daha önce operasyonların kodlarını kaydettiğimiz değerleri karşılaştırarak if else yapısı sayesinde yapılacak operasyona karar veririz) gerçekleştirilmesi gereken operasyonu gerçekleştiririz. (İlk çağırımda, rt registerine yazma olmadığı için write_sig ve clock sinyallerini 0 verdik.) Operasyon gerçekleştikten sonra operasyonun sonucunun alınmasının ardından ise bu sonucu rd registerine yazmak için register modülünü tekrar çağırıyoruz (write ve clock sinyallerine 1 vererek) ve hesaplanan sonucu rd registerine yazdırıyoruz.

Register modülü:

Register modülünde ise, 32 bitlik register arrayi oluşturuyoruz (32 adet register bloğu ve her blok 32 bit yani [32][32] lik double array). Daha sonra oluşturduğumuz bu register arrayini binary dosya okuyarak dolduruyoruz. İnput olarak gelen Rs ve Rt nin adreslerinden, rs ve rt nin contentini output olarak veriyoruz. (registers[10001] şeklinde vererek bunu gerçekleyebiliyoruz decimal sayıya çevirmemize gerek kalmıyor). Son olarak ise bir always bloğu tanımlıyoruz ve bu blok, clok sinyali 1 olduğu sürece çalışacak şekilde ayarlıyoruz (posedge ile) daha sonra bu blok içerisinde write sinyalinin kontorlunu yapıyoruz eğer bu sinyal 1 ise, input olarak gelen “write_data” yı Rd adresine yazıyoruz. Ve son işlem olarak tüm registerleri binary olarak register dosyasına kaydediyoruz.


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Transcript
Loading work.mips_testbench
Loading work.mips_core
Loading work.mips_registers
add wave -position insertpoint \
sim:/mips_testbench/instruction_set \
sim:/mips_testbench/result
VSIM 5> step -current
time = 0, instruction=00000000001000100000000000100000 result =10000000000000000000000000000011
time = 10, instruction=00000000001000100000000000100001 result =00000000000000000000000000000001
time = 20, instruction=00000000001000100000000000100010 result =10000000000000000000000000000010
time = 30, instruction=00000000001000100000000000100100 result =00000000000000000000000000000011
time = 40, instruction=00000000001000100000000000100101 result =00000000000000000000000000000100
time = 50, instruction=00000000001000100000000011000011 result =00000000000000000000000000000101
time = 60, instruction=00000000001000100000000011000010 result =00000000000000000000000000000110
time = 70, instruction=00000000001000100000000011000000 result =00000000000000000000000000000111
time = 80, instruction=00000000001000100000000011101011 result =00000000000000000000000000001000
VSIM 6>
Windows'u ara


C:\Users\yldzf\Desktop\Organizasyon\Projects\Project 2\templateForProject2_restored\simulation\modelsim\output.mem - Sub
File Edit Selection Find View Goto Tools Project Preferences Help
module mips_core(instruction, result);
output.mem
time = 0, instruction=00000000001000100000
// memory data file (do not edit the following line - required for mem load use)
// instance=/mips_testbench/m/m1/registers
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
10000000000000000000000000000011
00000000000000000000000000000001
10000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000100000
00000000000000000000000000001001
00000000000000000000000000001100
00000000000000000000000001100000
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
Line 1, Column 1
Windows'u ara