Modules

And4bit(A, B, Output)

4 bitlik A, B inputlarını alır. Her eleman and kapısı yardımıyla and ‘lenir. Sonuç 4 bitlik Outputa yazdılır.

And32bit(A, B, Output)

4 bitlik elemanlar and4bit modülüyle toplanır. Sonuç 32 bitlik Outputa yazdılır.

Or4bit(A, B, Output)

4 bitlik A, B inputlarını alır. Her eleman or kapısı yardımıyla or‘lanır. Sonuç 4 bitlik Outputa yazdılır.

Or32bit(A, B, Output)

4 bitlik elemanlar or4bit modülüyle toplanır. Sonuç 32 bitlik Outputa yazdılır.

Xor32bit(A, B, Output)

Xor gate pek çok kez çalıştırılarak 32 bitlik Output oluşturulur.

Nor32bit(A, B, Output)

Nor gate pek çok kez çalıştırılarak 32 bitlik Output oluşturulur.

Adder1bit(A, B, Cin, S, Cout)

İnputlar xorlanır çıkan sonuçla input girişi(cin) zorlanır, çıktı S(sonuçtur). Cout ise ilk xordan çıkan sonuç ve cin andlenmesiyle oluşan çıktı ve inputların andlenmesiyle oluşan çıktı orlanmasıyla oluşur.

Adder32bit(A, B, Cin, S, Cout)

32 bit full-adder ise 1 bit full adderların peş peşe 32 kez çağırılması ile oluşur. Her oluşan adder çıktısı diğer modülün girişine verilir.

Subt32bit(A, B, Bin, D, Bout)

32 bit full-subtractor 32 bit full-adder yardımıyla yapılır. A inputundan B inputu çıkarmak için A + (-B) + 1 formülü kullanılabilir. Bu yüzden B inputunun not kapısı yardımıyla tersi alınır. Daha sonra full-adder kullanılarak toplanır. Sonuç 32 bitlik S outputudur.

Mux2x1(A, B, X, S)

A ve B inputlar, X seçme girişi ve S çıkıştır. Seçme girişi tersiyle A inputu andlenir. Seçme girişiyle B andlenir. Sonuçlar orlanır.

Mux4x1(out, i0, i1, i2, i3, s1, s0)

i0 i1 girişleri, i2 i3 girişleri 2x1 bir mux'a verilir. Seçme girişleri s1 dir. Çıkan sonuçlar tekrar 2x1 mux'a s0 seçme girişiyle verilir ve out çıkışı alınır.

Mux8x1(i0, i1, i2, i3, i4, i5, i6, i7, s0, s1, s2, out)

2 adet 4x1 mux ve 1 adet 2x1 mux kullanılarak sonuç elde edilir. 4x1 muxların girişleri i0-i3, i4-7 ve seçme girişleri s0 s1 dir. Sonuçlar 2x1 mux'a s2 seçme girişle verilir out oluşur.

Mux32bit2x1(A, B, X, S)

Mux2x1 modülünün 32 bitlik inputlar almış halidir. A ve B 32 bitlik inputlar, S 32 bitlik outputdur. X seçme girişine göre A ve B den birini seçer.

Shift_right(A, X, Out)

A input, X kaydırılma adetini belirleyen 5 bitlik binary sayı ve Out 32 bitlik çıkıştır. İnputlar A[0] A[1] şeklinde 2x1 muxlara verilir bu sıralı şekilde devam eder, ilk biti ifade eden blok için X[0] seçme girişi olacaktır. Çıkan sonuçlarda 2 arttırarak w[0] w[2] şeklinde devam eder. En son 16 bitlik kaydırma bloğu olur. Çünkü 5 bitlik bir sayı gösteriminde en fazla 31 adet kaydırma yapılabilir ve buda 16 + 8 + 4 + 2 + 1 e eşittir. 16 bitlik bloğun çıkışı 32 bitlik Out çıkışını verir. Toplamda 160 adet 2x1 mux kullanılmış olur.

Shift_left(A, X, Out)

Shift right da olduğu gibi inputlar, kaydırma işlemlerini 2x1 mux yardımıyla yapar. Bu sefer kayma soldan olacağından A[1] A[0] şeklindedir. Önemli husus lojik sol kaydırmada kaydırılan yerlerden kalan boşluğa sıfır (1'b0) atanacağıdır.

ALU(A, B, S, Out)

A ve B 32 bitlik inputlar. S 3 bitlik seçme girişi ve Out çıkıştır. 32 adet 8x1 mux ile yapılır. 0. Giriş and, 1. Giriş or, 2. Giriş adding, 3. Giriş xor, 4. Giriş subtraction, 5. Giriş arithmetic right shift, 6. Giriş logic left shift ve son 7. Giriş nor işlemini yapmalıdır. 32 bit inputlar tek bitlik girişler ve çıkış alan mux ile olamayacağından 32 adet kullanılır. İlk muxun çıkışı Out[0], ikinci muxun çıkışı Out[1] i verecek şekilde 32 bitlik Out sağlanmış olur.

Control_unit(select_bits_ALU, function_code)

Gelen function-code a göre ALU nun seçeceği R type instruction select bitleri oluşturulur. Function code 6 bit, select_bits_ALU ise 3 bittir. Ör: (100000) function code geldiğinde modül bunu tanır ve 010 select bitlerini gönderir.

Mips_registers(read_data_1, read_data_2, write_data, read_reg_1, read_reg_2, write_reg, signal_reg_write, clk)

Gelen write sinyaline göre "register.mem" dosyasından okuma veya yazdırma işlemi yapar. Okunan datalar read_data_1 ve 2 de, yazdırılacak data write_data da tutulur. Okunacak ve yazılacak registerlar ise read_reg_1,2 ve write_reg de tutulur. Parametrelerden anlaşılacağı üzere 2 data okunurken tek data yazdırılır. Yazma sinyali de signal_reg_writedadır.

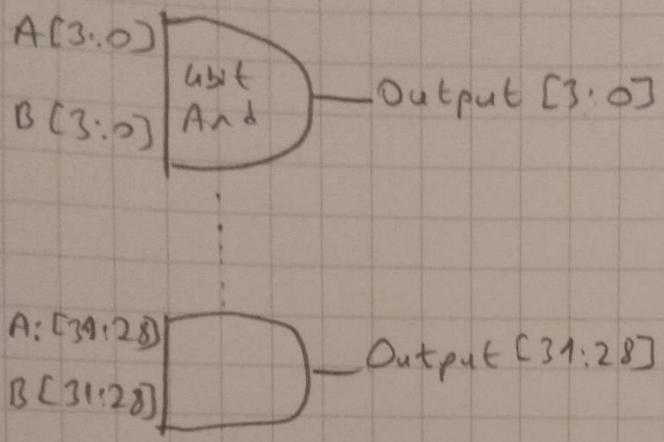
Mips32(instruction, result)

Ana modül budur. 32 bitlik gelen instructionı rs, rt, rd, shamt, function-code olarak algılar. Mips_registers modülü yardımıyla data okuma yazma yapar. Control unit ile function code u anlamlandırır ve ALU ya gönderir. Shamt'a sign-extend uygular.

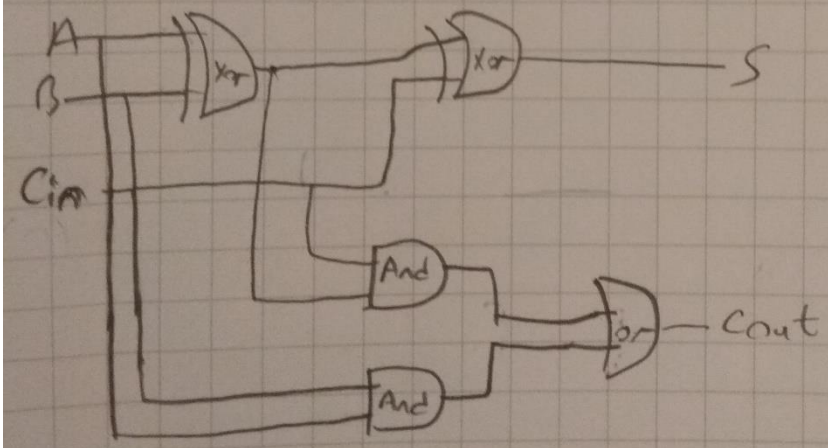
Mips32_testbench ()

Testbench de ise 10 instruction oluşturup mips32 modülünü çağırıyoruz, initialize yardımıyla 10 kez farklı inputlar veriyor, result outputunu alıyoruz. Ekrana her instructionı ve Rtype bölümlerini ve result değerlerini bastırıyoruz.

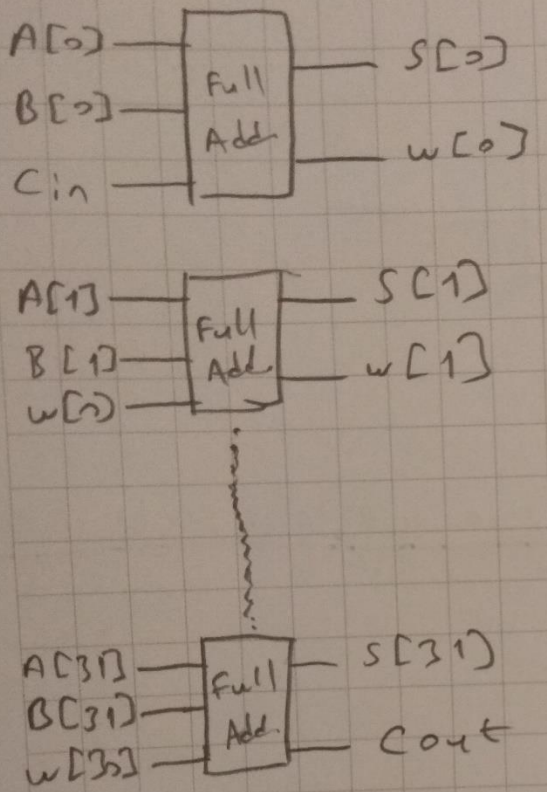
And Gate (32 bit) using 4 bit (A, B, Output)



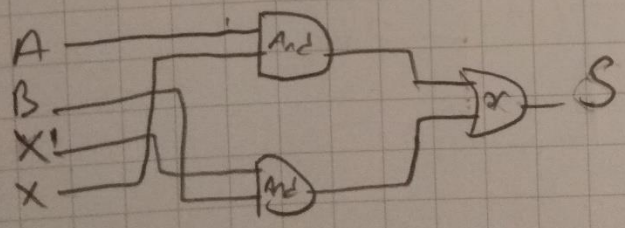
Full Adder (1 bit) (A, B, Cin, S, Cout)



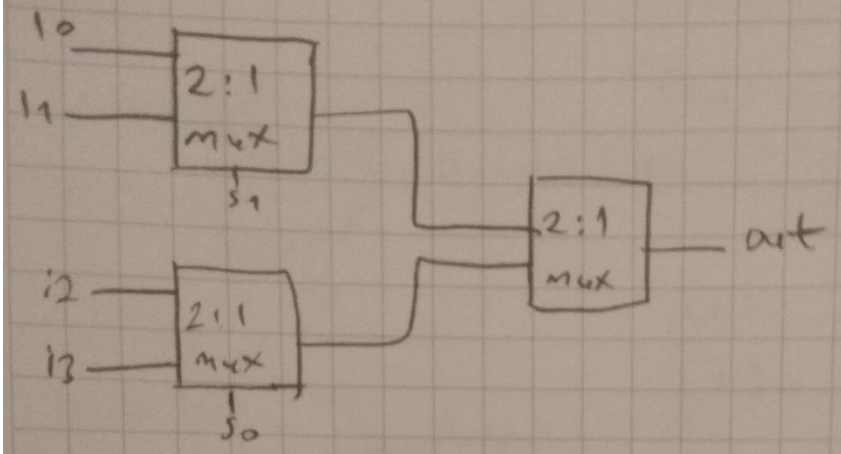
Full Adder (32 bit) (A, B, Cin, S, Cout)



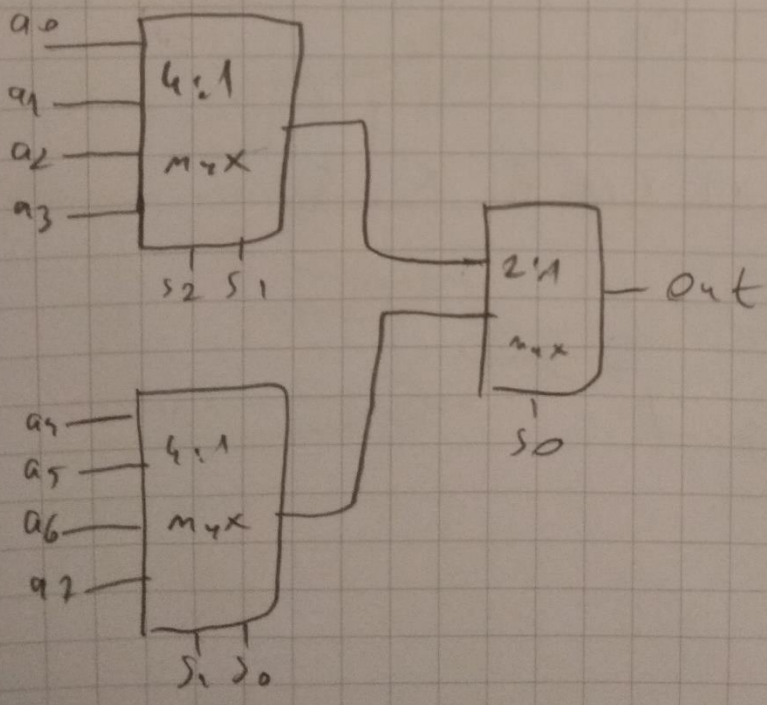
Mux 2x1 (A, B, X, S)



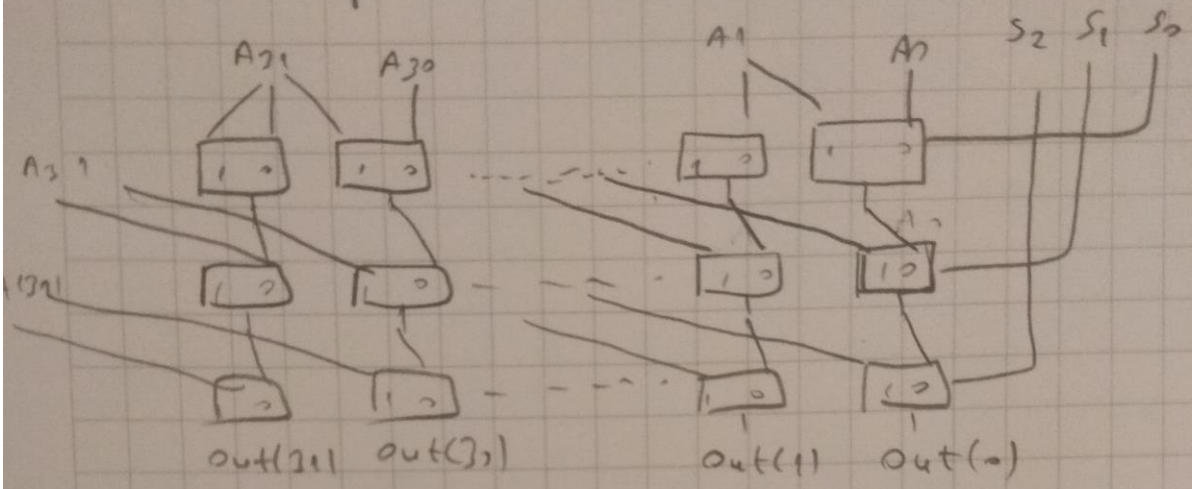
Mux 4x1 (out, i0, i1, i2, i3, s1, s0);



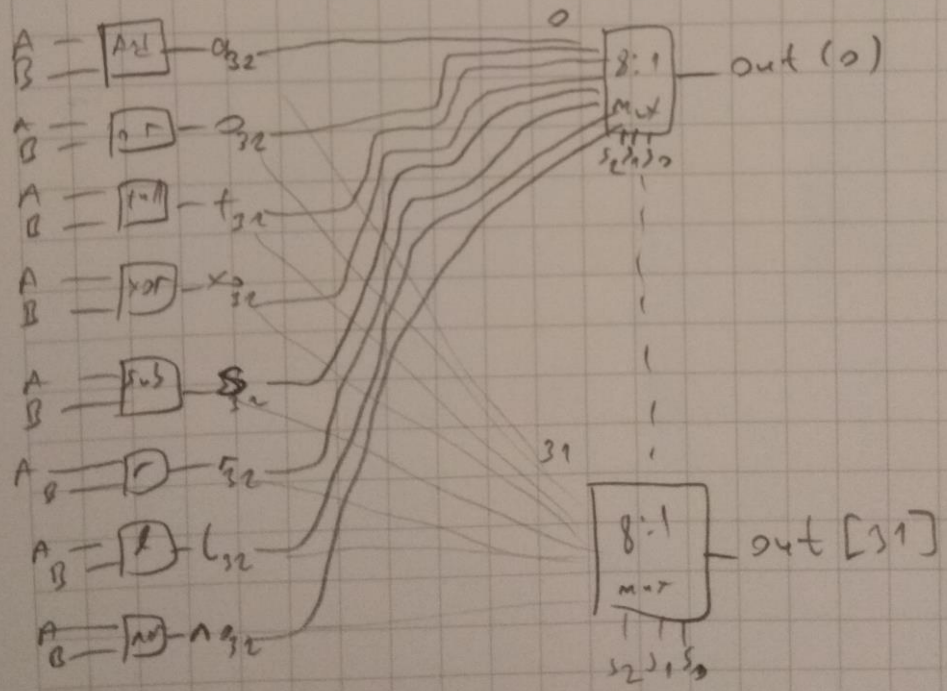
Mux 8x1 (a0, a1, a2, a3, a4, a5, a6, a7, s0, s1, s2, out)



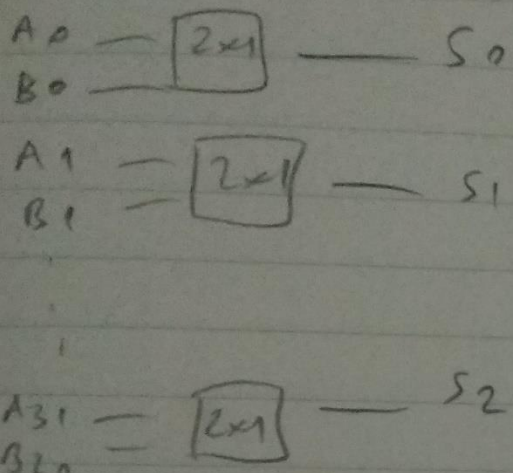
Shift right (A, S, Out);



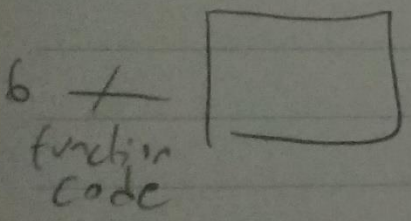
Alu (A, B, S, out)



Mux32bit-2x1(A, B, X, S);



Control Unit (select_bits_ALU, function_code)

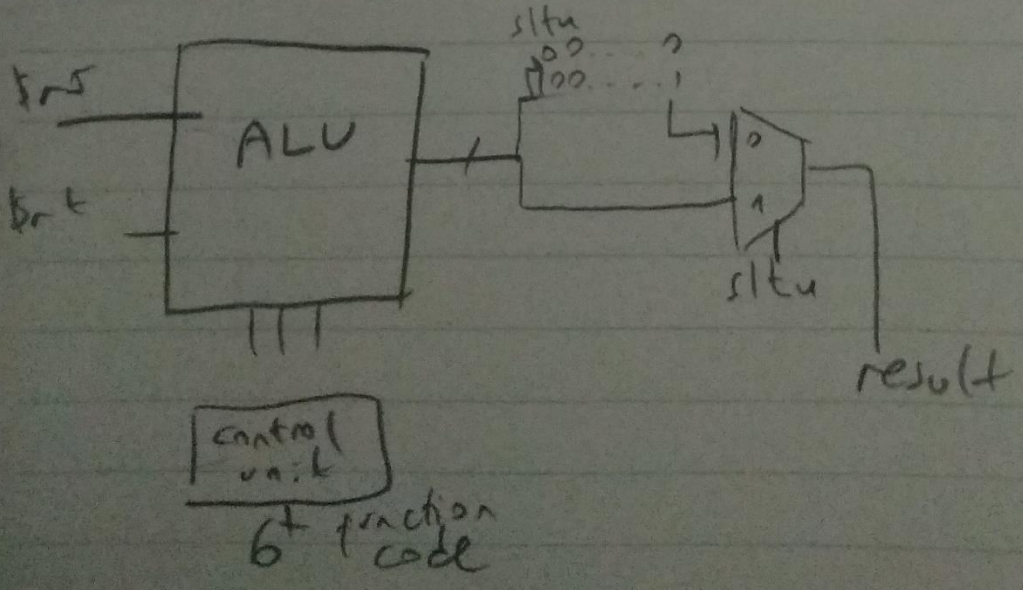


$S_0 = F_2 \cdot F_0 + F_5 \cdot F_1 F_0'$

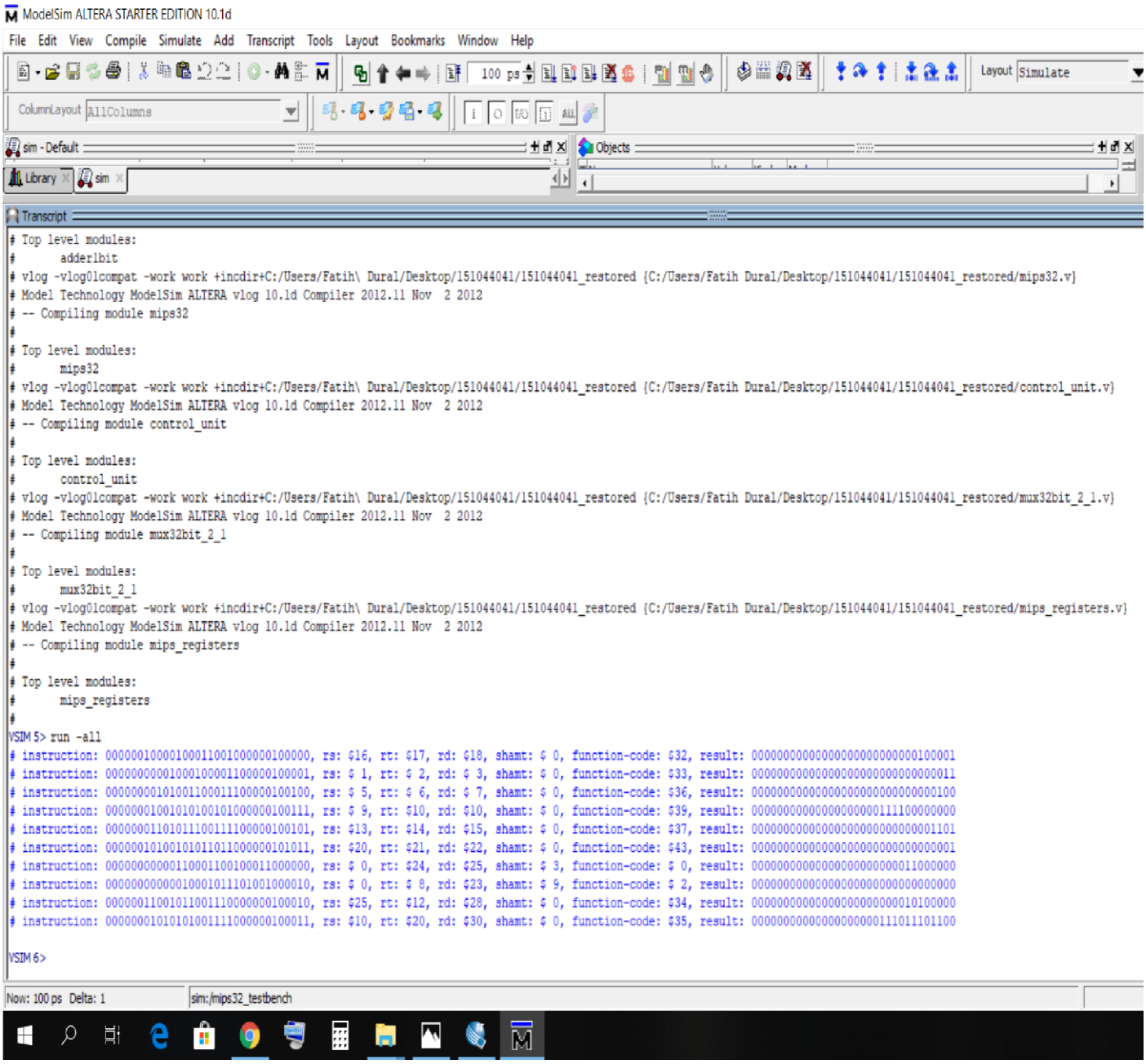
$S_1 = F_1' \text{ xor } F_2$

$S_2 = F_1 + (F_5 + F_0)'$

Mips32 (instruction, result)



add
addu
and
nor
or
sltu
sll
srl
sub
subu


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
sim - Default
Objects
Library
sim
Transcript
Top level modules:
adder1bit
vlog -vlog01compat -work work +incdir+C:/Users/Fatih\ Dural/Desktop/151044041/151044041_restored {C:/Users/Fatih Dural/Desktop/151044041/151044041_restored/mips32.v}
Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
-- Compiling module mips32
Top level modules:
mips32
vlog -vlog01compat -work work +incdir+C:/Users/Fatih\ Dural/Desktop/151044041/151044041_restored {C:/Users/Fatih Dural/Desktop/151044041/151044041_restored/control_unit.v}
Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
-- Compiling module control_unit
Top level modules:
control_unit
vlog -vlog01compat -work work +incdir+C:/Users/Fatih\ Dural/Desktop/151044041/151044041_restored {C:/Users/Fatih Dural/Desktop/151044041/151044041_restored/mux32bit_2_1.v}
Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
-- Compiling module mux32bit_2_1
Top level modules:
mux32bit_2_1
vlog -vlog01compat -work work +incdir+C:/Users/Fatih\ Dural/Desktop/151044041/151044041_restored {C:/Users/Fatih Dural/Desktop/151044041/151044041_restored/mips_registers.v}
Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
-- Compiling module mips_registers
Top level modules:
mips_registers
VSIM 5> run -all
instruction: 00000010000100011001000000100000, rs: $16, rt: $17, rd: $18, shamt: $ 0, function-code: $32, result: 00000000000000000000000000100001
instruction: 00000000001000100001100000100001, rs: $ 1, rt: $ 2, rd: $ 3, shamt: $ 0, function-code: $33, result: 00000000000000000000000000000011
instruction: 00000000101001100011100000100100, rs: $ 5, rt: $ 6, rd: $ 7, shamt: $ 0, function-code: $36, result: 00000000000000000000000000000100
instruction: 00000001001010100101000000100111, rs: $ 9, rt: $10, rd: $10, shamt: $ 0, function-code: $39, result: 00000000000000000000001111100000000
instruction: 00000001101011100111100000100101, rs: $13, rt: $14, rd: $15, shamt: $ 0, function-code: $37, result: 00000000000000000000000000001101
instruction: 00000010100101011011000000101011, rs: $20, rt: $21, rd: $22, shamt: $ 0, function-code: $43, result: 00000000000000000000000000000001
instruction: 00000000000011000110010001100000, rs: $ 0, rt: $24, rd: $25, shamt: $ 3, function-code: $ 0, result: 00000000000000000000000011000000
instruction: 00000000000010001011010010000010, rs: $ 0, rt: $ 8, rd: $23, shamt: $ 9, function-code: $ 2, result: 00000000000000000000000000000000
instruction: 00000011001011001110000000100010, rs: $25, rt: $12, rd: $28, shamt: $ 0, function-code: $34, result: 00000000000000000000000010100000
instruction: 00000001010101001111000000100011, rs: $10, rt: $20, rd: $30, shamt: $ 0, function-code: $35, result: 00000000000000000001110111101100
VSIM 6>
Now: 100 ps Delta: 1
sim:/mips32_testbench