Faculty of Engineering
Ain Shams University

Computer and Systems Engineering Department

Computer Organization (2)

Project Name	MIPS Simulator	
Description	MIPS Processor simulator designed to simulate a big variety of assembly code instructions on the run.	
Languages	Verilog and Python	
Students	Group (3)	
	Boula Nashat	33779
	Peter Rateb	33757
	Beshoy Anwar	33758
	Mennat'allah Mohamed	33907
Presented to	Eng. Ahmed Fathy	
Date	4/12/2017	

Table of Content:

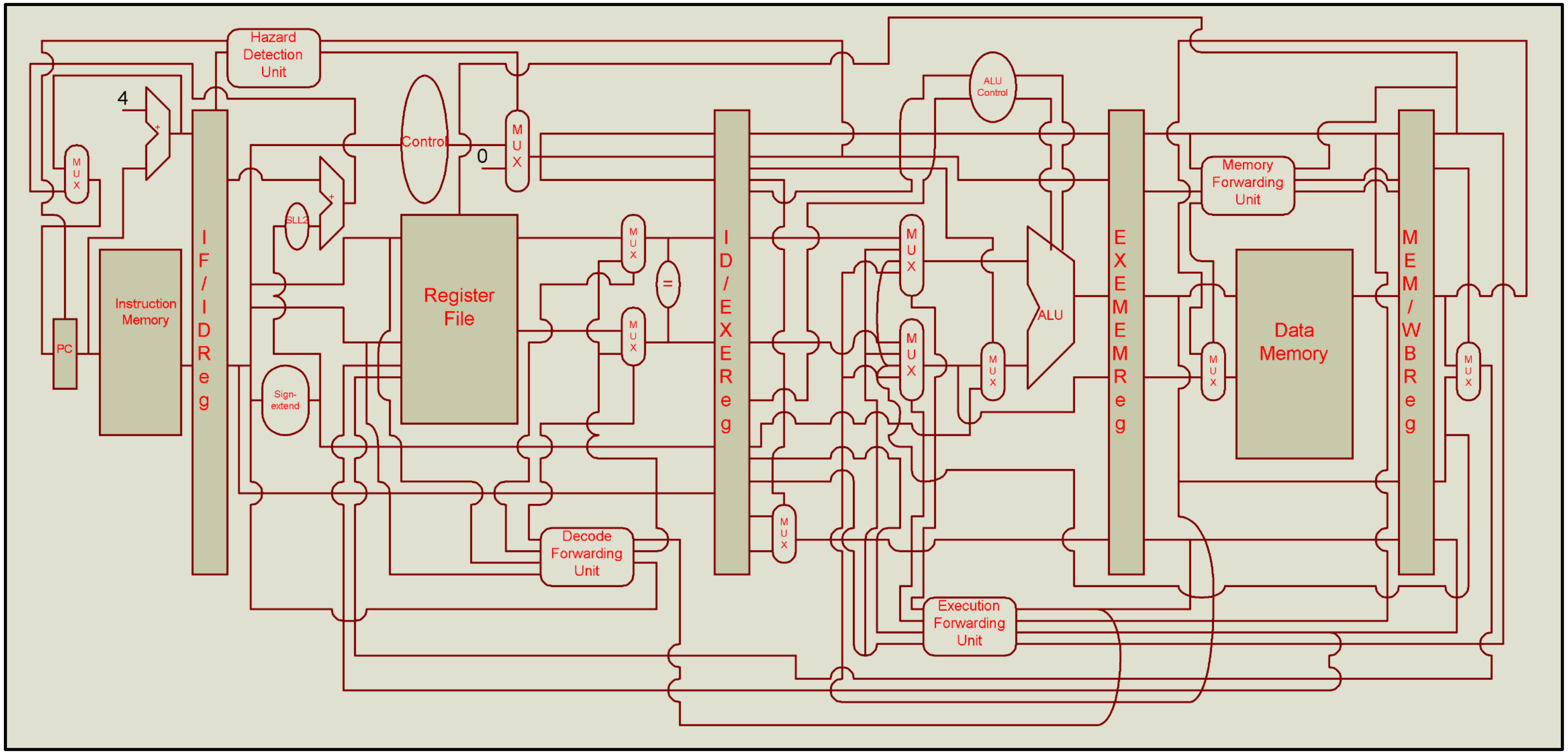
Introduction:

The simulator is a tool to simulate any given assembly code using an implemented MIPS processor architecture that supports pipe-lining and forwarding.

Objectives:

To simulate a big variety of assembly instructions, simulate it right, fast and efficient.

Used MIPS Architecture:



Pipeline Registers:

How pipeline registers was implemented.

IF/ID Register	
Instruction	PC Output
32 bits	32 bits

ID/EXE Register										
Control Signals							Sign Extended Output	Read Data 2	Read Data 1	Instruction
RegDst	Mem Read	Mem to Reg	ALU Operation	Mem Write	ALU Src	Reg Write				
1 bit	1 bit	1 bit	2 bits	1 bit	1 bit	1 bit	32 bits	32 bits	32 bits	32 bits

EXE/MEM Register								
Control Signals				Overflow Flag	Zero Flag	Write Reg	Read Data 2	Result of ALU
Reg Write	Mem Write	Mem to Reg	Mem Read					
1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	5 bits	32 bits	32 bits

MEM/WB Register				
Control Signals			Write Reg	Mem Read
Mem to Reg	ALU Result	Reg Write		
1 bit	32 bits	1 bit	5 bits	32 bits

Memory and Registers Assumptions:

- Instruction memory contains 1024 32-bits-long registers.
- Register file contains 32 32-bits-long registers.
- Data memory contains 1024 32-bits-long registers.
- Register file writing happens in the first half cycle of the clock (posedge) and reading from it happens in the second half cycle of the clock (negedge).

Implemented Data Hazards Fixes:

This is a list of all implemented forwarding list

Hazard Detection Unit

Memory to Instruction Decoding forwarding

Execution to Instruction Decoding forwarding

Memory to Execution forwarding

Execution to Execution forwarding

Memory to Memory forwarding

Implemented Control Hazards Fixes:

Static not taken prediction
and stalling if predict is miss

Supported Instructions:

This is a list of all supported MIPS instructions:

R	add
	sub
	or
	and
	nor
	sll
	srl
	slt
I	beq
	sw
	lw
	ori
	addi

Used Tests:

This is a list of some of the tests conducted to make sure the simulator is working perfectly.

Trivial Test

```
addi $s3, $zero, 18
sw $zero, 4($zero)
ori $s4, $zero, 16
nor $s5, $zero, $s3
or $s3, $zero, $s3
lw $s6, 4($zero)
slt $t8, $s5, $s4
sll $t2, $s4, 2
srl $t3, $s3, 1
```

```
20130012
ac000004
34140010
0013a827
00139825
8c160004
02b4c02a
00145080
00135842
```

Expected Output

```
$s3 = 18
mem4 = 0
$s4 = 16
$s5 = -19
$s3 = 18
$s6 = 0
$t8 = 1
$t2 = 64
$t3 = 9
```

1 stall and
EX to ID forwarding

```
addi $s0, $zero, 10
addi $s3, $zero, 5
addi $s4, $zero, 5
beq $s3, $s4, L1
add $s0, $s0, $s3
L1: sub $s0, $s0, $s3
```

```
2010000a
20130005
20140005
12930001
02138020
02138022
```

Expected Output

```
$s0 = 10
$s3 = 5
$s4 = 5
$s0 = 5
```

Exe to Exe forwarding

```
sw $zero, 8($zero)
addi $s3, $zero, 9
add $s2, $zero, $zero
and $s6, $s2, $s3
lw $t1, 8($s6)
```

```
ac000008
20130009
00009020
0253b024
8ec90008
```

Expected Output

```
mem8 = 0
$s3 = 9
$s2 = 0
$s6 = 0
$t1 = 0
```

Mem to Exe forwarding

```
addi $s2, $zero, 9
add $t5, $zero, $zero
sw $zero, 4($t5)
lw $s5, 4($t5)
addi $s6, $zero, 16
sw $s2, 8($s5)
```

```
20120009
00006820
ada00004
8db50004
20160010
aeb20008
```

Expected Output

```
$s2 = 9
$t5 = 0
mem4 = 0
$s5 = 0
$s6 = 16
mem8 = 9
```

Mem to Mem forwarding

```
addi $s3, $zero, 4
sw $s3, 4($zero)
lw $s5, 4($zero)
sw $s5, 8($zero)
```

```
20130004
ac130004
8c150004
ac150008
```

Expected Output

```
$s3 = 4
mem4 = 4
$s5 = 4
mem8 = 4
```

Exe to Mem and Exe to Exe

```
addi $s0, $zero, 0
addi $s1, $zero, 1
addi $s2, $zero, 2
addi $s3, $zero, 3
sub $s1, $s1, $zero
add $s1, $s1, $s2
sll $s1, $s1, 2
```

```
20100000
20110001
20120002
20130003
02208822
02328820
00118880
```

Expected Output

```
$s0 = 0
$s1 = 1
$s2 = 2
$s3 = 3
$s1 = 1
$s1 = 3
$s1 = 12
```

Hazard Detection and Mem to Exe

```
sw $zero, 4($zero)
addi $t3, $zero, 15
lw $s5, 4($zero)
add $s3, $s5, $s5
```

```
ac000004
200b000f
8c150004
02b59820
```

Expected Output

```
mem4 = 0
$t3 = 15
$s5 = 0
$s3 = 0
```

Stall before branch(lw case) and not taken

```
sw $zero, 4($zero)
addi $t2, $zero, 26
addi $t3, $zero, -26
lw $s3, 4($zero)
beq $s3, $t2, L
addi $t2, $t2, 26
L: addi $t2, $t2, -26
```

```
ac000004
200a001a
200bffe6
8c130004
11530001
214a001a
214affe6
```

Expected Output

```
mem4 = 0
$t2 = 26
$t3 = -26
$s3 = 0
$t2 = 52
$t2 = 26
```

Stall before branch(R format)
And taken branch

```
addi $t2, $zero, 25
add $t1, $zero, $zero
beq $t1, $t2, L
sub $s3, $t2, $t1
L: add $s3, $t2, $t2
```

```
200a0019
00004820
11490001
01499822
014a9820
```

Expected Output

```
$t2 = 25
$t1 = 0
$s3 = 25
$s3 = 50
```

Branch taken

```
addi $t2, $zero, 112
addi $t1, $zero, 112
and $s0, $zero, $zero
beq $t1, $t2, L
add $s0, $t2, $t1
L: sub $s0, $t2, $t1
```

```
200a0070
20090070
00008024
11490001
01498020
01498022
```

Expected Output

```
$t2 = 112
$t1 = 112
$s0 = 0
$s0 = 0
```

El wa7sh test

```
addi $t1,$0,8
addi $t0,$0,8
//ALU to ALu forwarding(rs)
addi $s1,$0,28
ori $s2,$s1,30
//ALU to ALu forwarding(rt)
srl $s3,$s1,1
sw $s3,0($t0)
sw $s2,4($t0)
//Mem to Mem forwarding
lw $s4,4($t0)
sw $s4,12($t0)
// 1 stall before branch ,
not taken branch, exe to
decode forwarding
add $s4,$s1,$t0
beq $s4,$s1,label1
sub $s4,$0,$0
label1: sll $s4,$t0,2
sw $t0,14($t1)
// 2stall before branch ,
taken branch
lw $t1,14($t1)
beq $t1,$t0,label2
addi $s5,$0,236
addi $s6,$0,326
//ALU to ALU forwarding (rs)
label2: addi $t1,$0,16
sw $t0,15($t1)
slt $t3,$t1,$t0
//Mem to ALU forwarding (rs)
lw $s6,4($t0)
addi $t5,$0,5
sw $t3,8($s6)
```

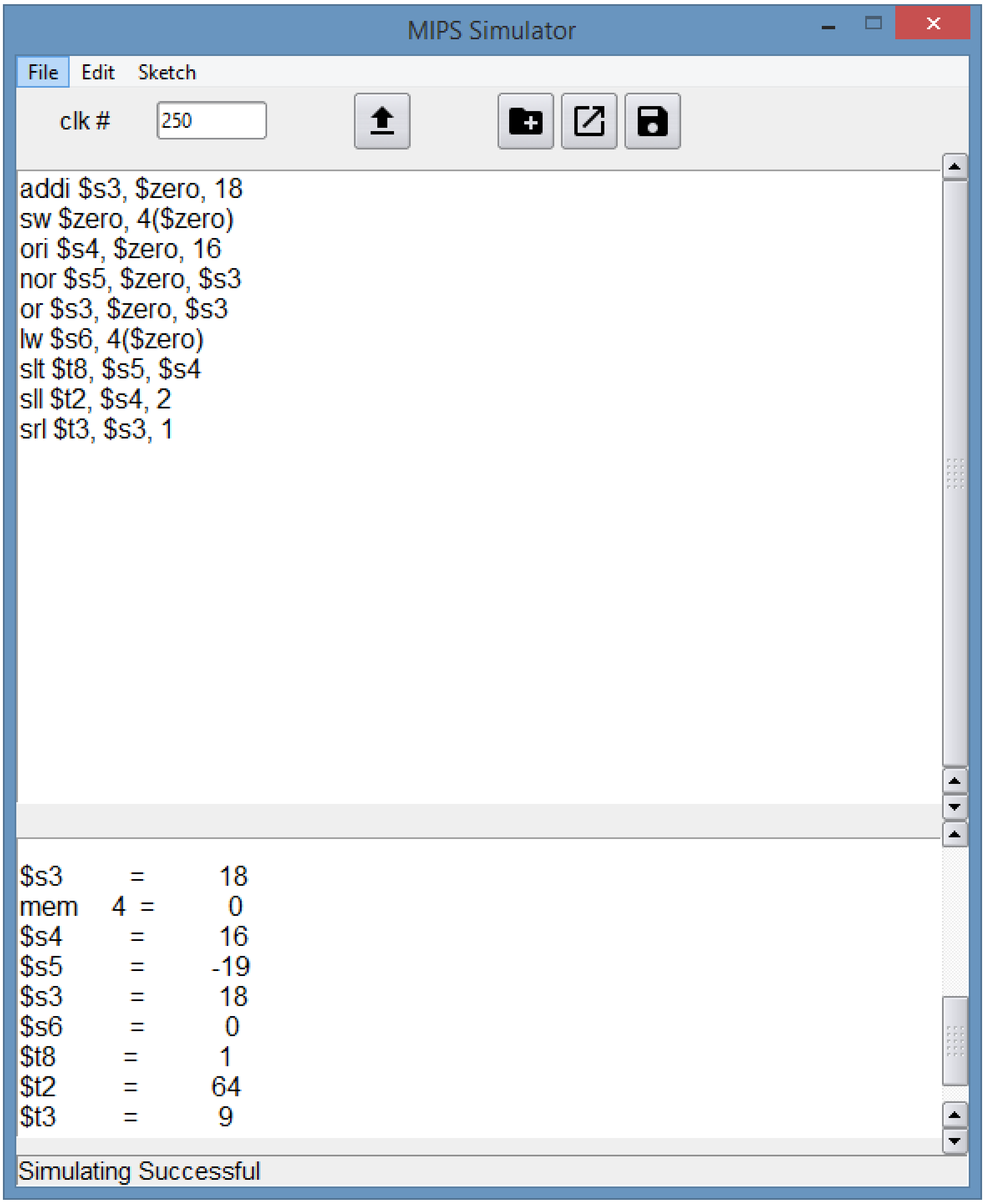
```
20090008
20080008
2011001c
3632001e
00119842
ad130000
ad120004
8d140004
ad14000c
0228a020
12340001
0000a022
0008a080
ad28000e
8d29000e
11090002
201500ec
20160146
20090010
ad28000f
0128582a
8d160004
200d0005
aecb0008
```

Expected Output

```
$t1 = 8
$t0 = 8
$s1 = 28
$s2 = 30
$s3 = 14
mem8 = 14
mem12 = 30
$s4 = 30
mem20 = 30
$s4 = 36
$s4 = 0
$s4 = 32
mem22 = 8
$t1 = 8
$t1 = 16
mem31 = 8
$t3 = 0
$s6 = 30
$t5 = 5
mem38 = 0
```

Simulator Outputs:

1. Trivial test.

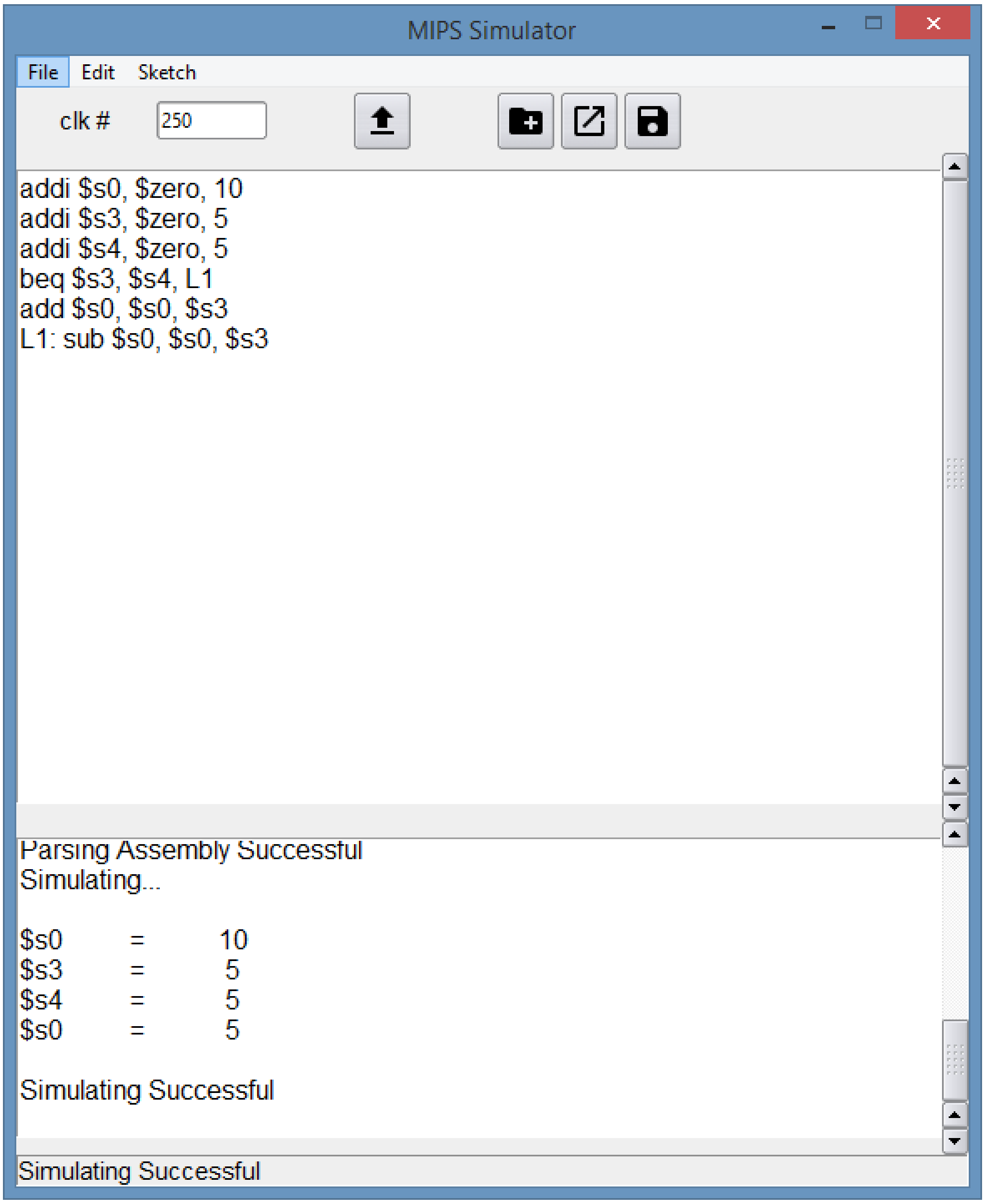


```
addi $s3, $zero, 18
sw $zero, 4($zero)
ori $s4, $zero, 16
nor $s5, $zero, $s3
or $s3, $zero, $s3
lw $s6, 4($zero)
slt $t8, $s5, $s4
sll $t2, $s4, 2
srl $t3, $s3, 1
```

```
$s3        =       18
mem    4  =        0
$s4        =       16
$s5        =      -19
$s3        =       18
$s6        =        0
$t8        =        1
$t2        =       64
$t3        =        9
```

Simulating Successful

2. 1 stall and EX to ID Forwarding

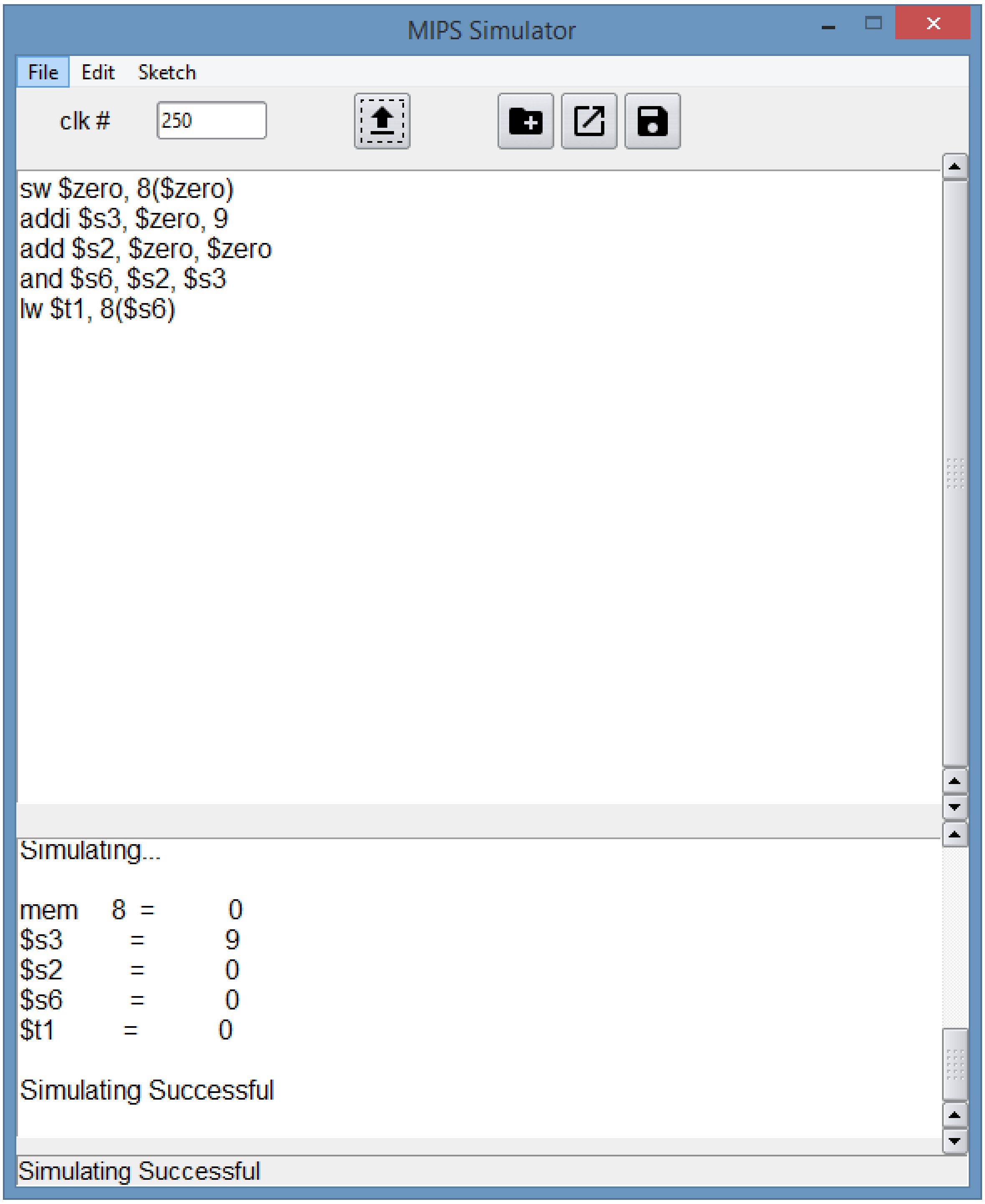


```
addi $s0, $zero, 10
addi $s3, $zero, 5
addi $s4, $zero, 5
beq $s3, $s4, L1
add $s0, $s0, $s3
L1: sub $s0, $s0, $s3
```

```
Parsing Assembly Successful
Simulating...

$s0      =      10
$s3      =       5
$s4      =       5
$s0      =       5

Simulating Successful
```

Simulating Successful

3. Exe to Exe Forwarding

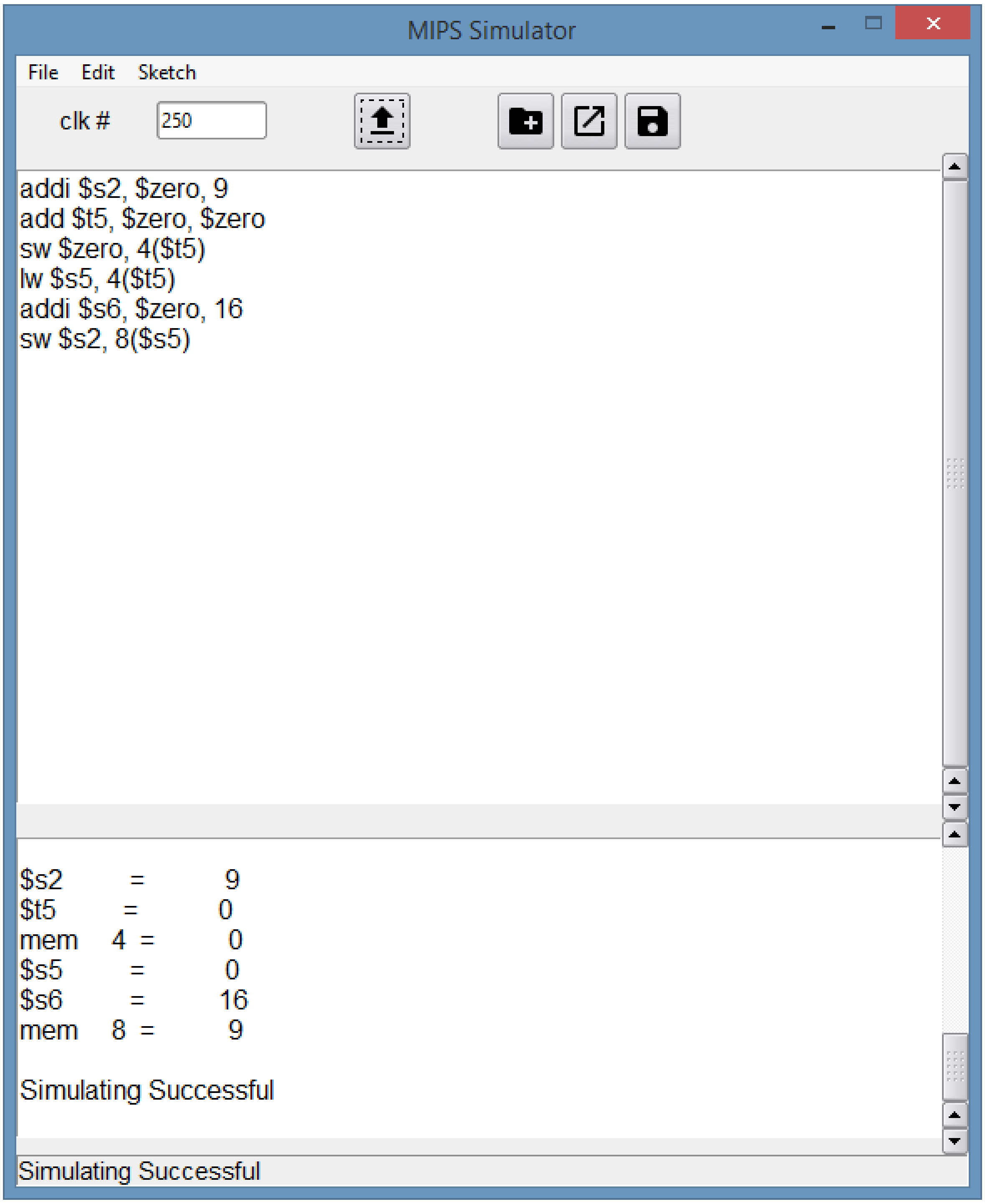


```
sw $zero, 8($zero)
addi $s3, $zero, 9
add $s2, $zero, $zero
and $s6, $s2, $s3
lw $t1, 8($s6)
```

```
Simulating...

mem    8  =        0
$s3       =        9
$s2       =        0
$s6       =        0
$t1       =        0

Simulating Successful
```

Simulating Successful

4. Mem to Exe forwarding

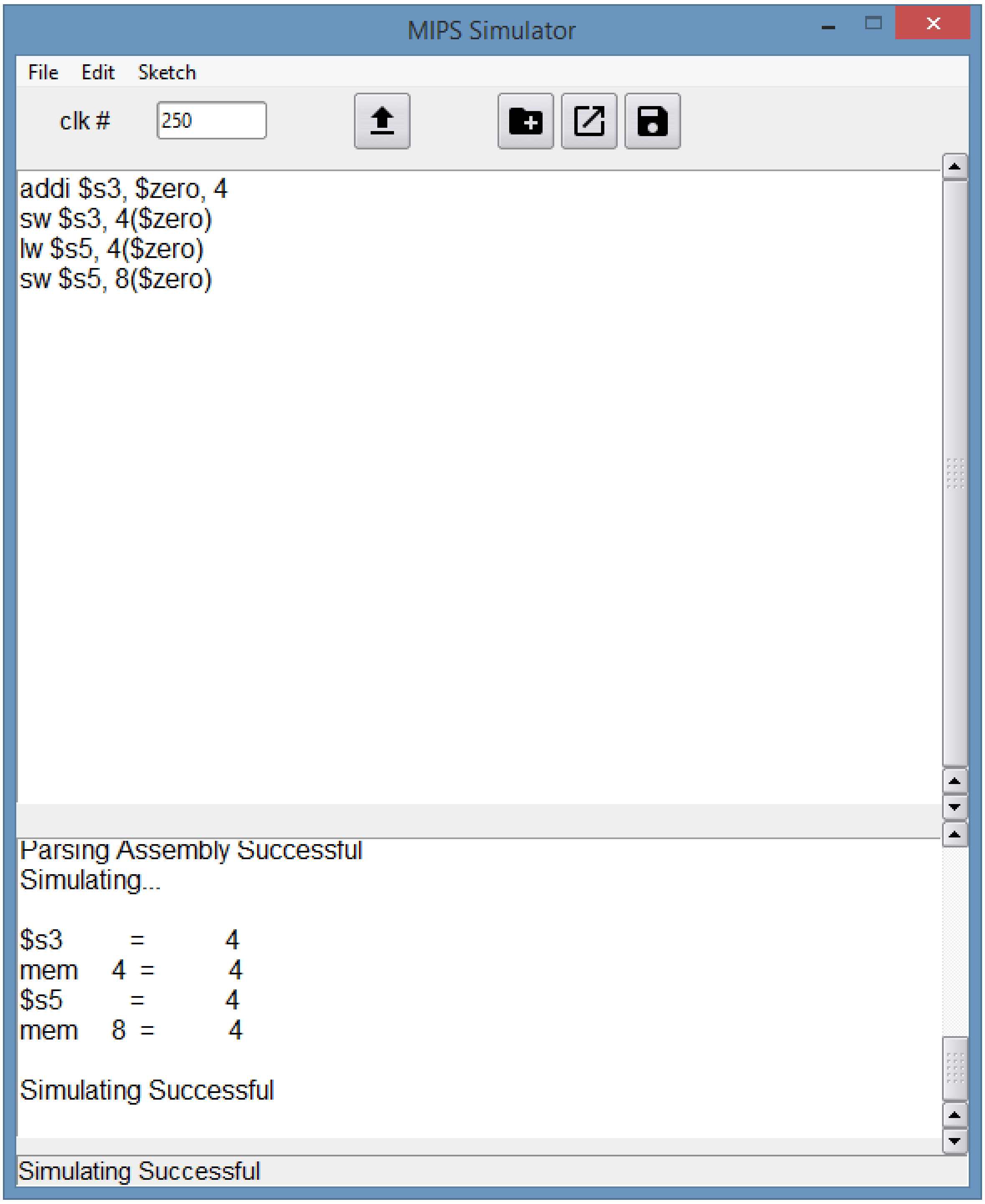


```
addi $s2, $zero, 9
add $t5, $zero, $zero
sw $zero, 4($t5)
lw $s5, 4($t5)
addi $s6, $zero, 16
sw $s2, 8($s5)
```

```
$s2       =       9
$t5       =       0
mem    4  =       0
$s5       =       0
$s6       =      16
mem    8  =       9

Simulating Successful
```

Simulating Successful

5. Mem to Mem forwarding

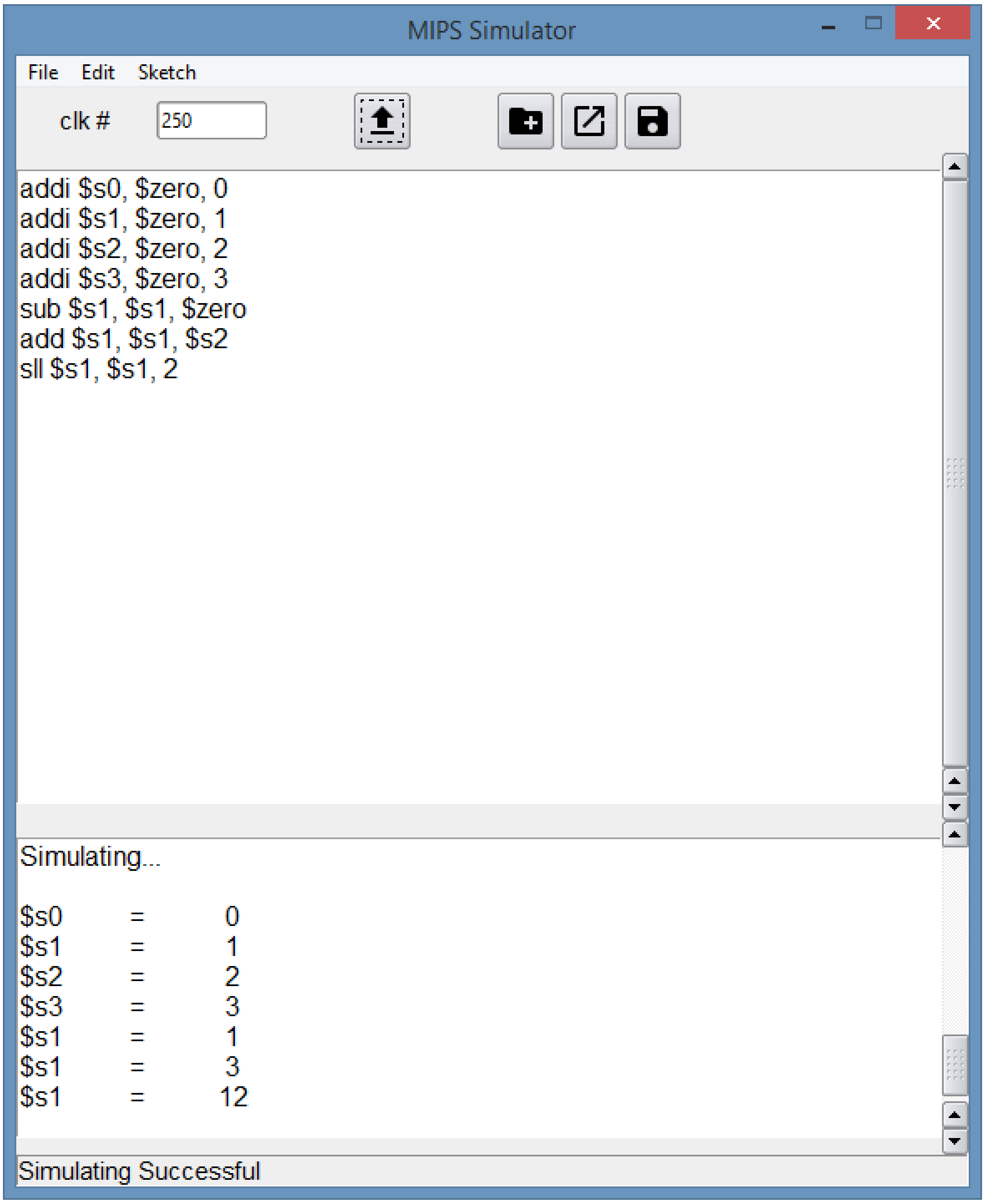


```
addi $s3, $zero, 4
sw $s3, 4($zero)
lw $s5, 4($zero)
sw $s5, 8($zero)
```

```
Parsing Assembly Successful
Simulating...

$s3         =          4
mem    4  =          4
$s5         =          4
mem    8  =          4

Simulating Successful
```

Simulating Successful

6. Exe to Mem and Exe to Exe

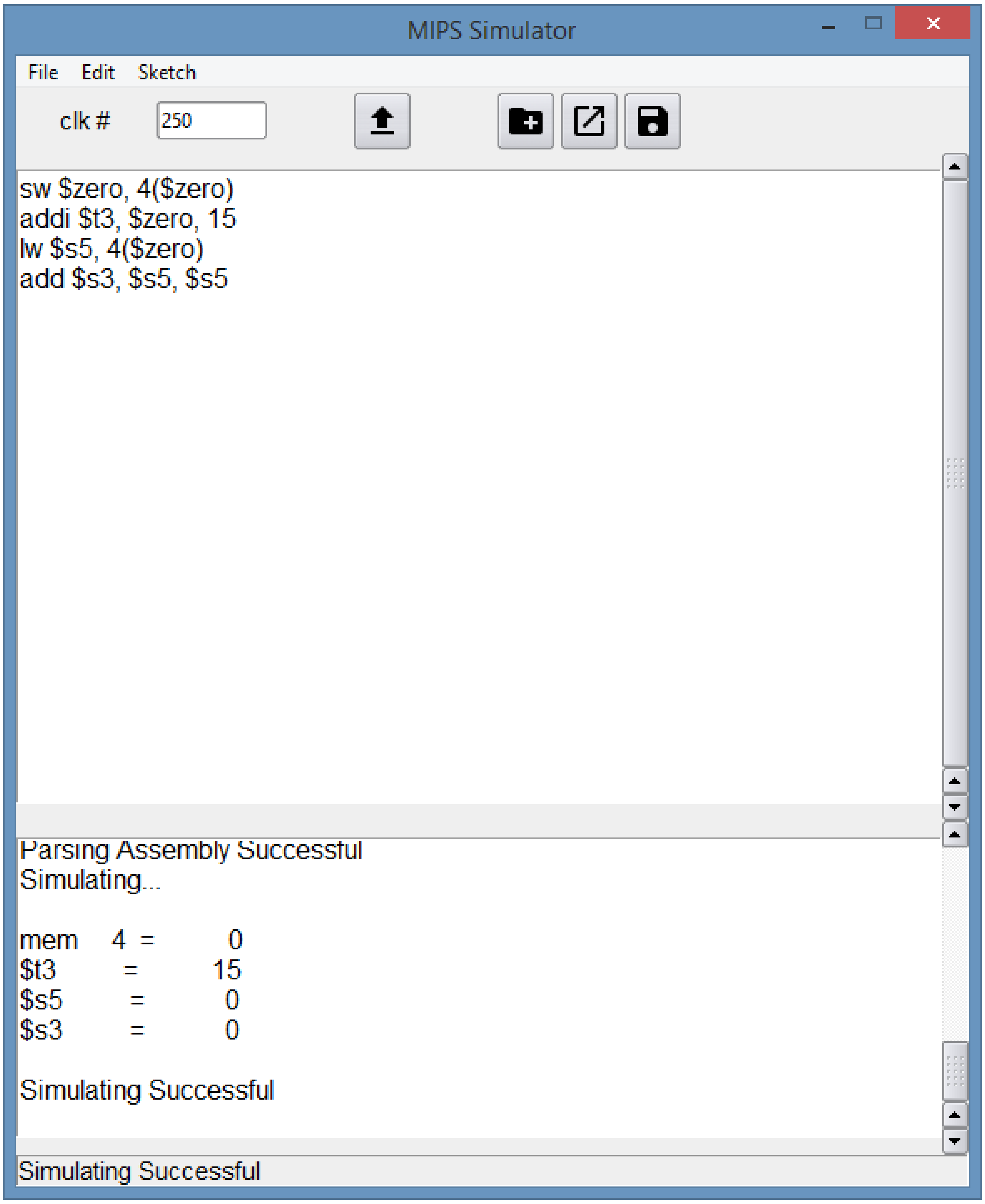


```
addi $s0, $zero, 0
addi $s1, $zero, 1
addi $s2, $zero, 2
addi $s3, $zero, 3
sub $s1, $s1, $zero
add $s1, $s1, $s2
sll $s1, $s1, 2
```

```
Simulating...

$s0     =     0
$s1     =     1
$s2     =     2
$s3     =     3
$s1     =     1
$s1     =     3
$s1     =    12
```

Simulating Successful

7. Hazard Detection and Mem to Exe

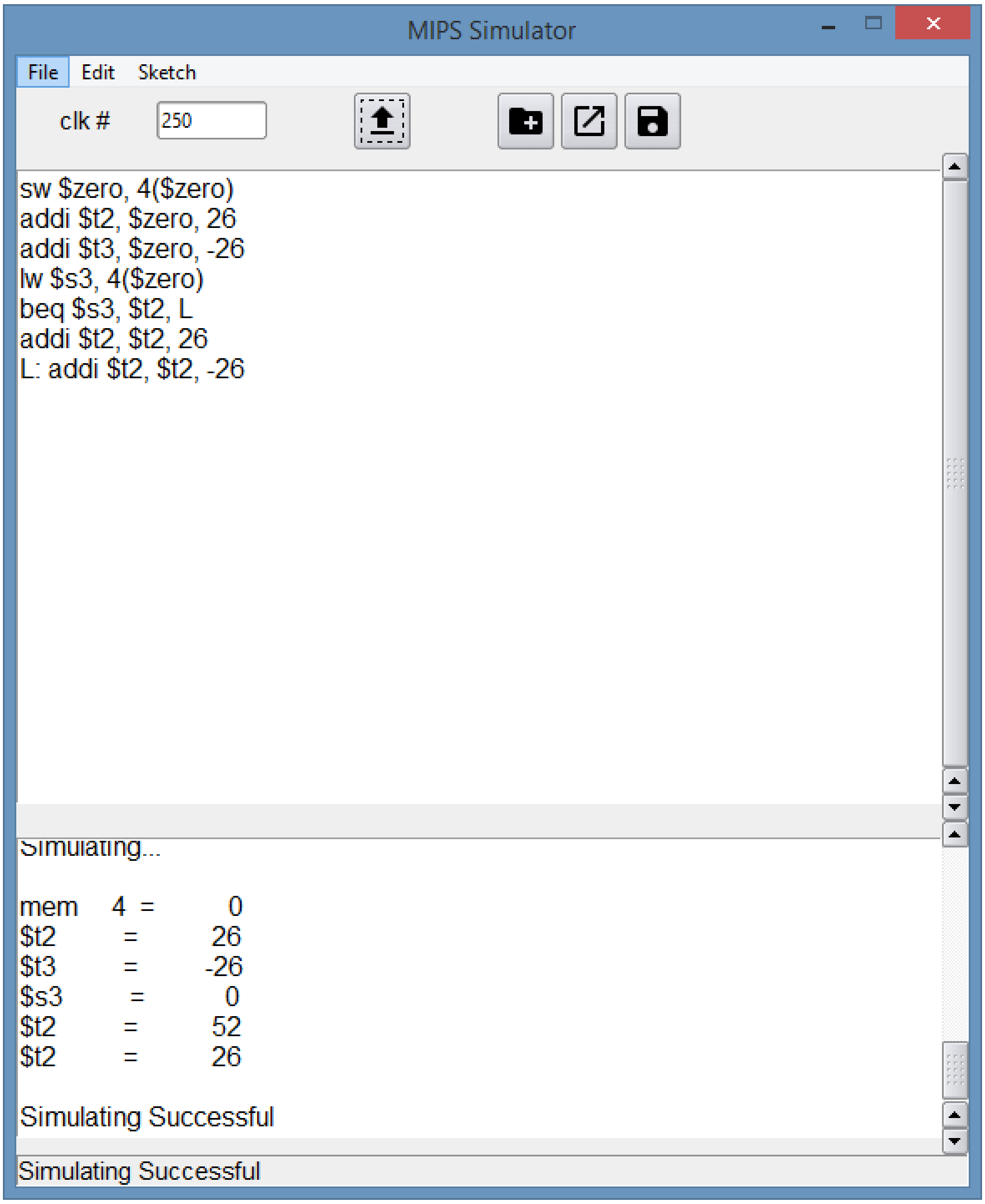


```
sw $zero, 4($zero)
addi $t3, $zero, 15
lw $s5, 4($zero)
add $s3, $s5, $s5
```

```
Parsing Assembly Successful
Simulating...

mem    4  =       0
$t3       =      15
$s5       =       0
$s3       =       0

Simulating Successful
```

Simulating Successful

8. Stall before branch

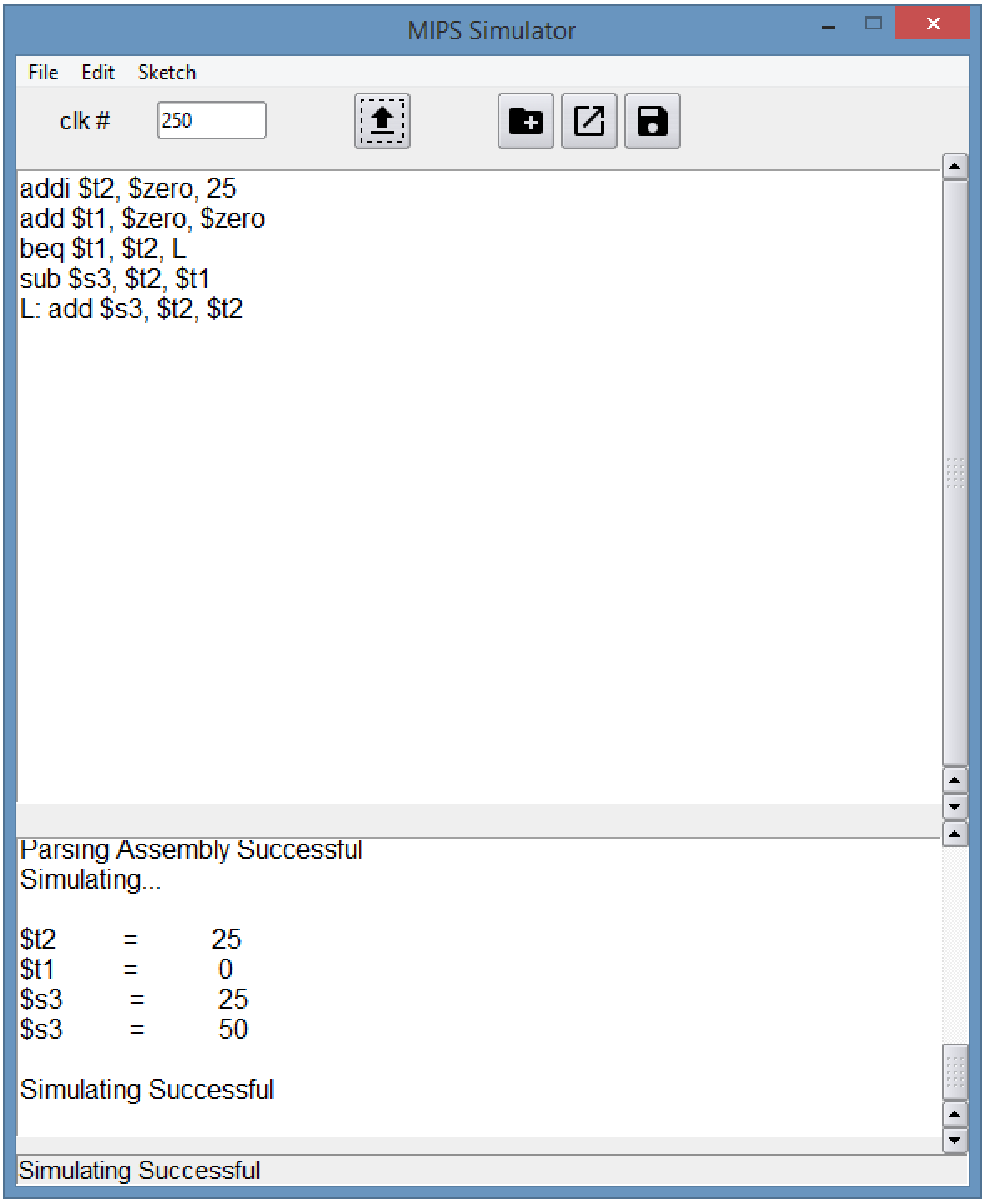


```
sw $zero, 4($zero)
addi $t2, $zero, 26
addi $t3, $zero, -26
lw $s3, 4($zero)
beq $s3, $t2, L
addi $t2, $t2, 26
L: addi $t2, $t2, -26
```

```
Simulating...

mem    4  =        0
$t2       =       26
$t3       =      -26
$s3       =        0
$t2       =       52
$t2       =       26

Simulating Successful
```

Simulating Successful

9. Stall before branch (R-format)

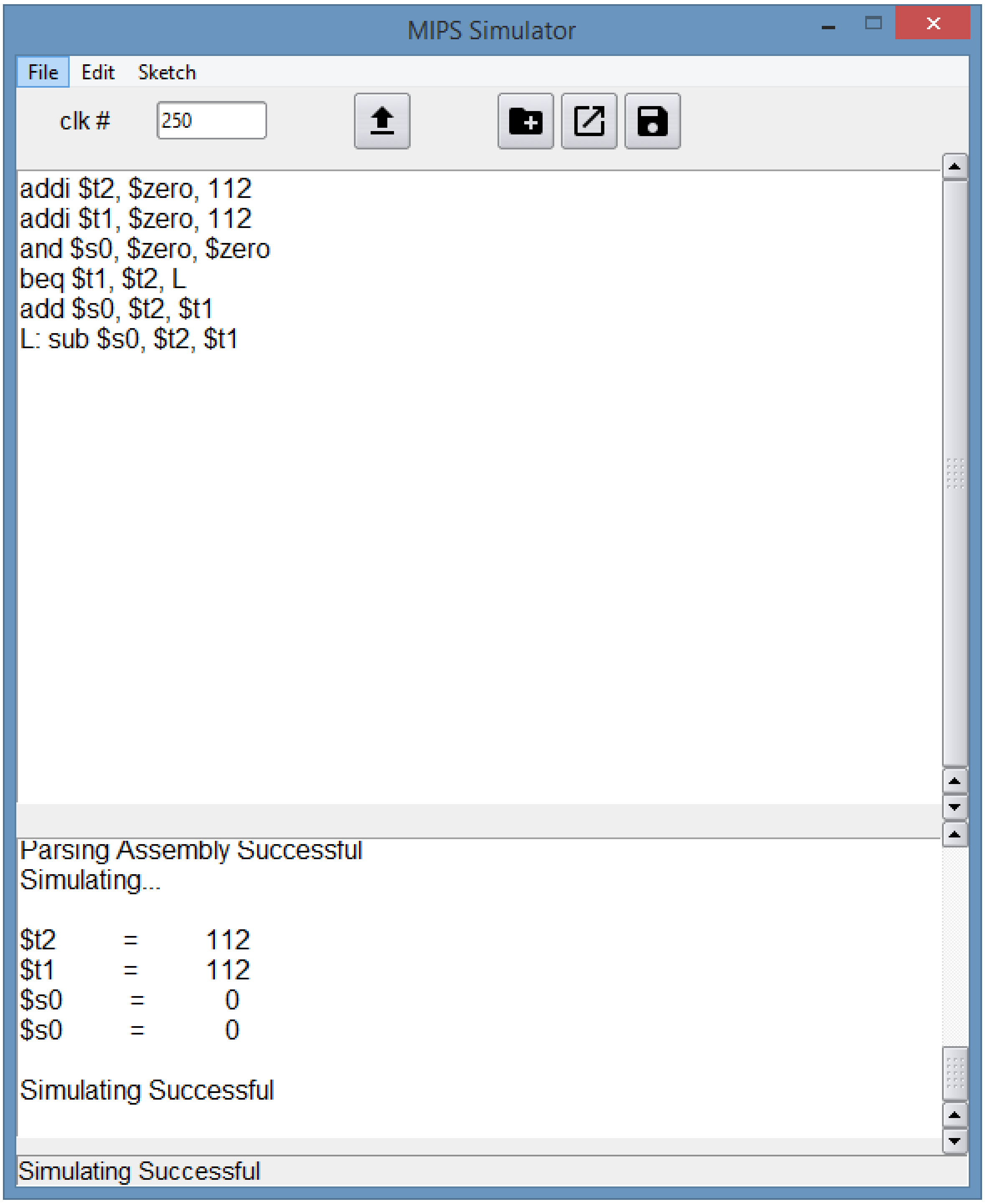


```
addi $t2, $zero, 25
add $t1, $zero, $zero
beq $t1, $t2, L
sub $s3, $t2, $t1
L: add $s3, $t2, $t2
```

```
Parsing Assembly Successful
Simulating...

$t2      =      25
$t1      =       0
$s3      =      25
$s3      =      50

Simulating Successful
```

Simulating Successful

10.Branch taken

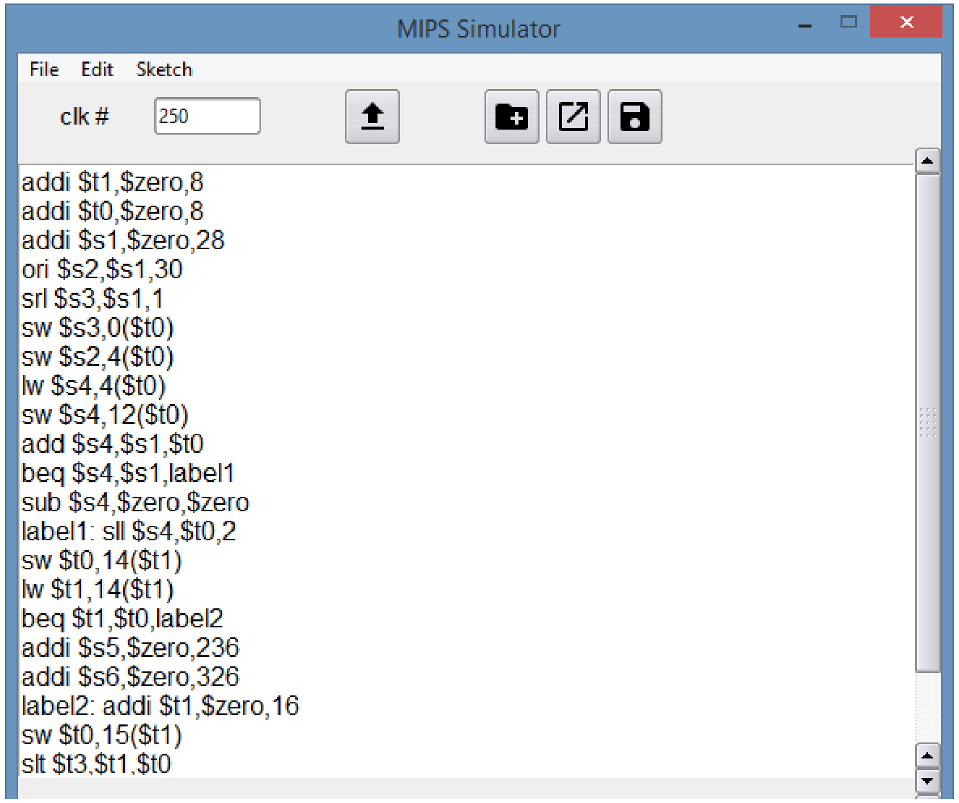


```
addi $t2, $zero, 112
addi $t1, $zero, 112
and $s0, $zero, $zero
beq $t1, $t2, L
add $s0, $t2, $t1
L: sub $s0, $t2, $t1
```

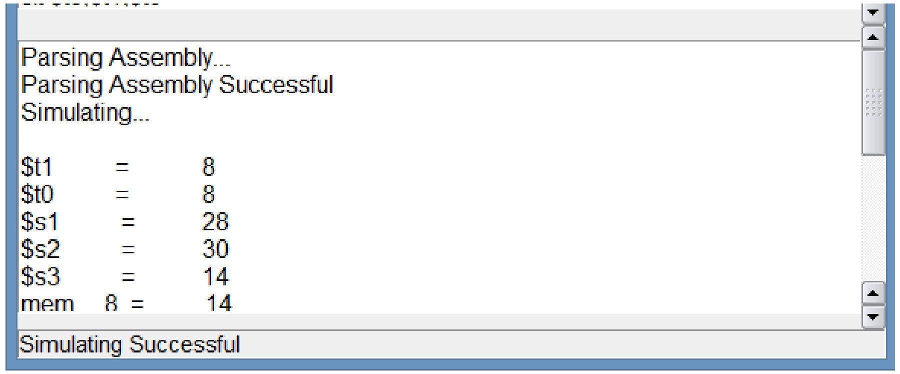
```
Parsing Assembly Successful
Simulating...

$t2       =      112
$t1       =      112
$s0       =        0
$s0       =        0

Simulating Successful
```

Simulating Successful

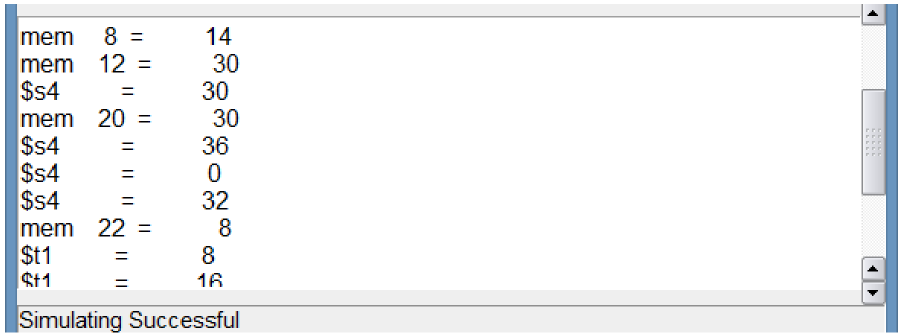
11.El wa7sh test



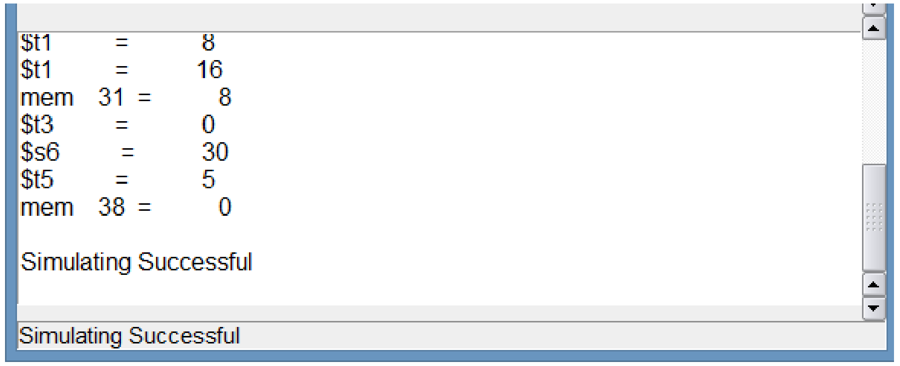
```
addi $t1,$zero,8
addi $t0,$zero,8
addi $s1,$zero,28
ori $s2,$s1,30
srl $s3,$s1,1
sw $s3,0($t0)
sw $s2,4($t0)
lw $s4,4($t0)
sw $s4,12($t0)
add $s4,$s1,$t0
beq $s4,$s1,label1
sub $s4,$zero,$zero
label1: sll $s4,$t0,2
sw $t0,14($t1)
lw $t1,14($t1)
beq $t1,$t0,label2
addi $s5,$zero,236
addi $s6,$zero,326
label2: addi $t1,$zero,16
sw $t0,15($t1)
slt $t3,$t1,$t0
```



```
Parsing Assembly...
Parsing Assembly Successful
Simulating...

$t1     =     8
$t0     =     8
$s1     =     28
$s2     =     30
$s3     =     14
mem   8 =     14
Simulating Successful
```



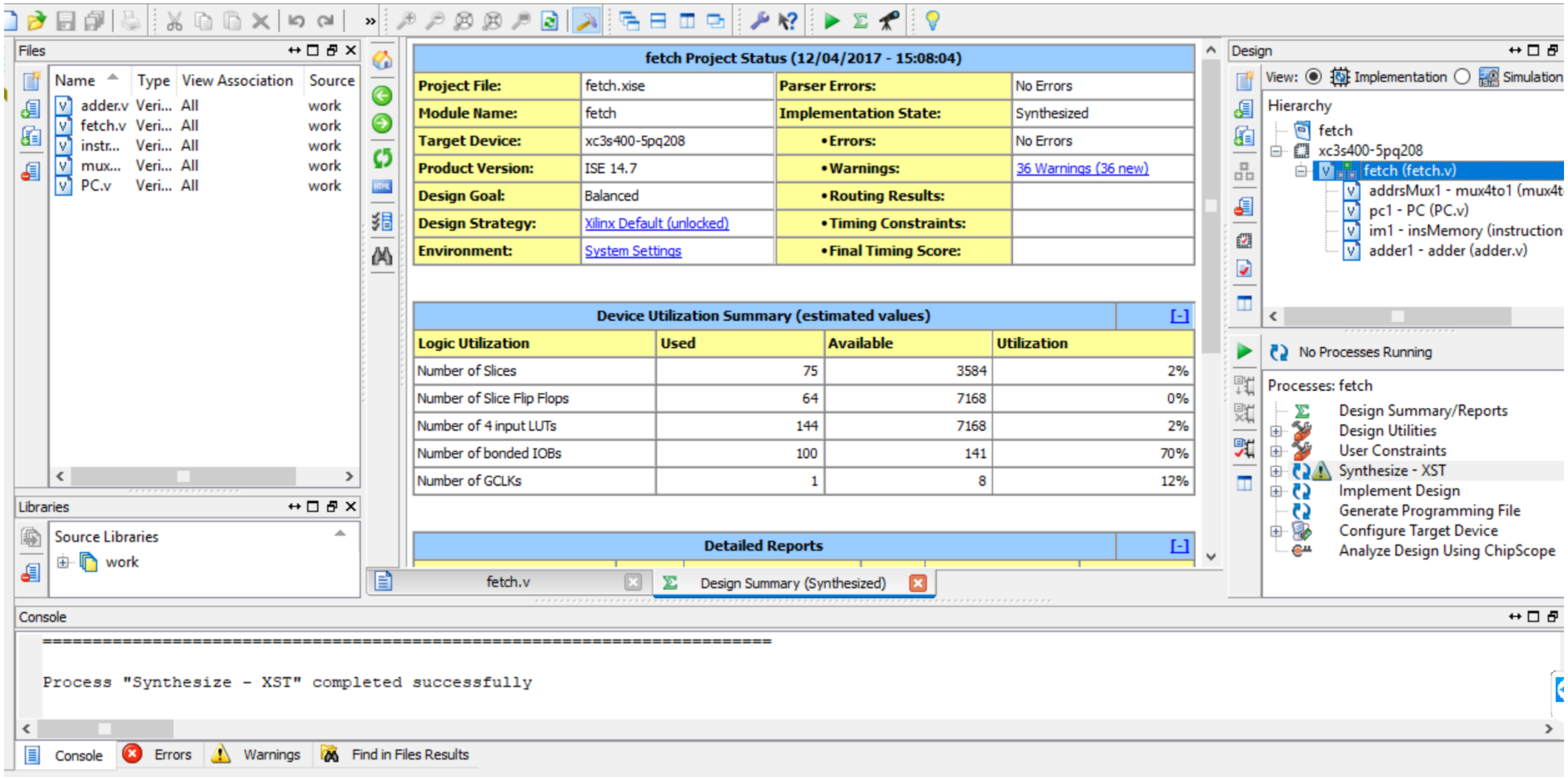
```
mem   8 =     14
mem  12 =      30
$s4     =     30
mem  20 =      30
$s4     =     36
$s4     =     0
$s4     =     32
mem  22 =      8
$t1     =     8
$t1     =     16
Simulating Successful
```



```
$t1     =     8
$t1     =     16
mem  31 =      8
$t3     =     0
$s6     =     30
$t5     =     5
mem  38 =      0

Simulating Successful
Simulating Successful
```

Xilinx Synthesis reports:

1. Instruction Fetch Stage:



Files

Name	Type	View Association	Source
adder.v	Veri...	All	work
fetch.v	Veri...	All	work
instr...	Veri...	All	work
mux...	Veri...	All	work
PC.v	Veri...	All	work

Libraries

Source Libraries
work

fetch Project Status (12/04/2017 - 15:08:04)

Project File:	fetch.xise	**Parser Errors:**	No Errors
Module Name:	fetch	**Implementation State:**	Synthesized
Target Device:	xc3s400-5pq208	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	36 Warnings (36 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slices	75	3584	2%
Number of Slice Flip Flops	64	7168	0%
Number of 4 input LUTs	144	7168	2%
Number of bonded IOBs	100	141	70%
Number of GCLKs	1	8	12%

Detailed Reports [-]

fetch.v | Design Summary (Synthesized)

Design

View: Implementation Simulation

Hierarchy
- fetch
- xc3s400-5pq208
  - fetch (fetch.v)
    - addrsMux1 - mux4to1 (mux4t
    - pc1 - PC (PC.v)
    - im1 - insMemory (instruction
    - adder1 - adder (adder.v)

No Processes Running

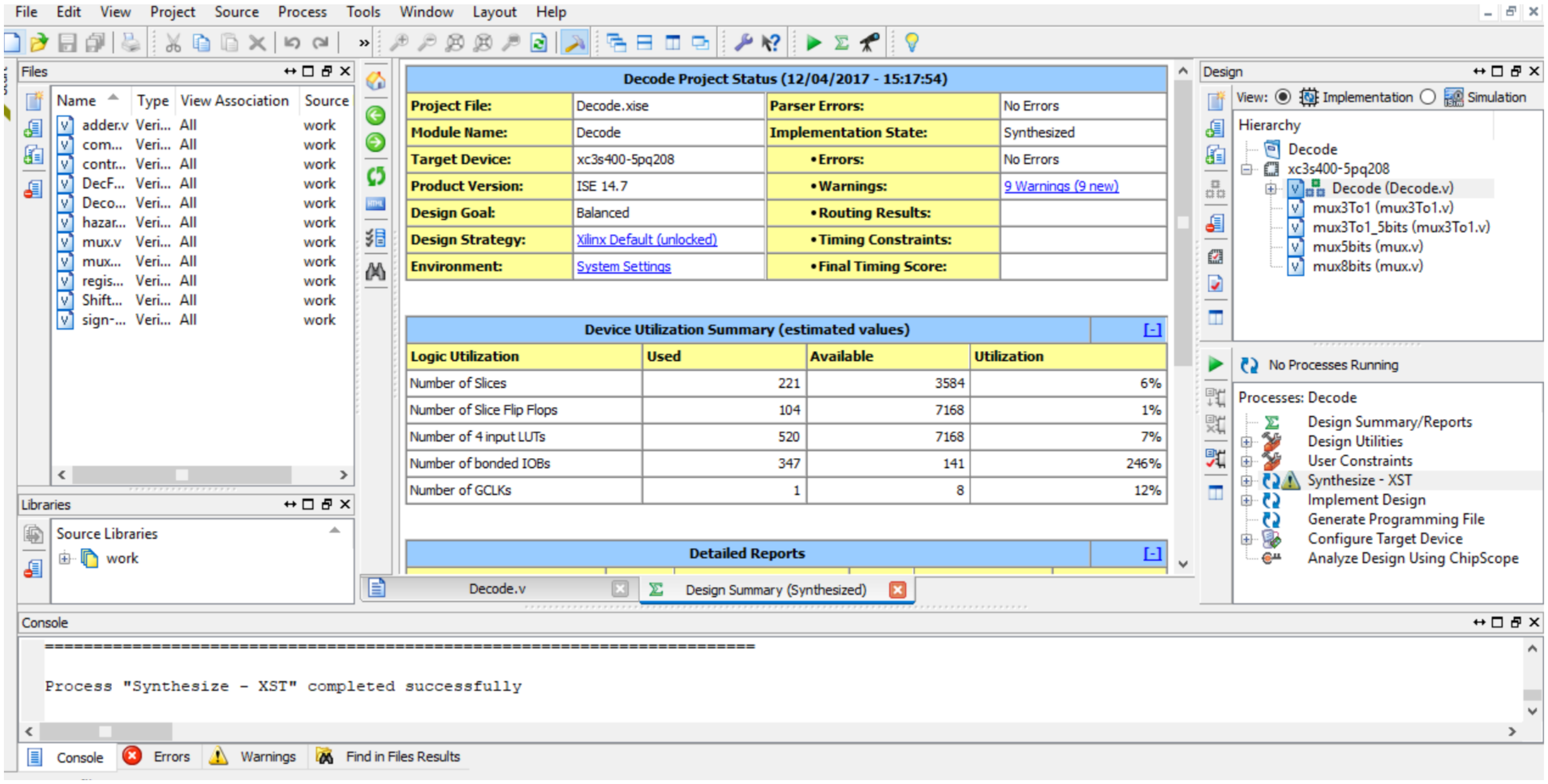
Processes: fetch
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

Console

```
======================================================================

Process "Synthesize - XST" completed successfully
```

Console | Errors | Warnings | Find in Files Results

2. Decode and Write Back Stages:


File Edit View Project Source Process Tools Window Layout Help
Files
Libraries
Source Libraries
work
Design
View: Implementation Simulation
Console
Errors
Warnings
Find in Files Results
Decode.v
Design Summary (Synthesized)

Name	Type	View Association	Source
adder.v	Veri...	All	work
com...	Veri...	All	work
contr...	Veri...	All	work
DecF...	Veri...	All	work
Deco...	Veri...	All	work
hazar...	Veri...	All	work
mux.v	Veri...	All	work
mux...	Veri...	All	work
regis...	Veri...	All	work
Shift...	Veri...	All	work
sign-...	Veri...	All	work

Decode Project Status (12/04/2017 - 15:17:54)

Project File:	Decode.xise	**Parser Errors:**	No Errors
Module Name:	Decode	**Implementation State:**	Synthesized
Target Device:	xc3s400-5pq208	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	9 Warnings (9 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slices	221	3584	6%
Number of Slice Flip Flops	104	7168	1%
Number of 4 input LUTs	520	7168	7%
Number of bonded IOBs	347	141	246%
Number of GCLKs	1	8	12%

Detailed Reports [-]

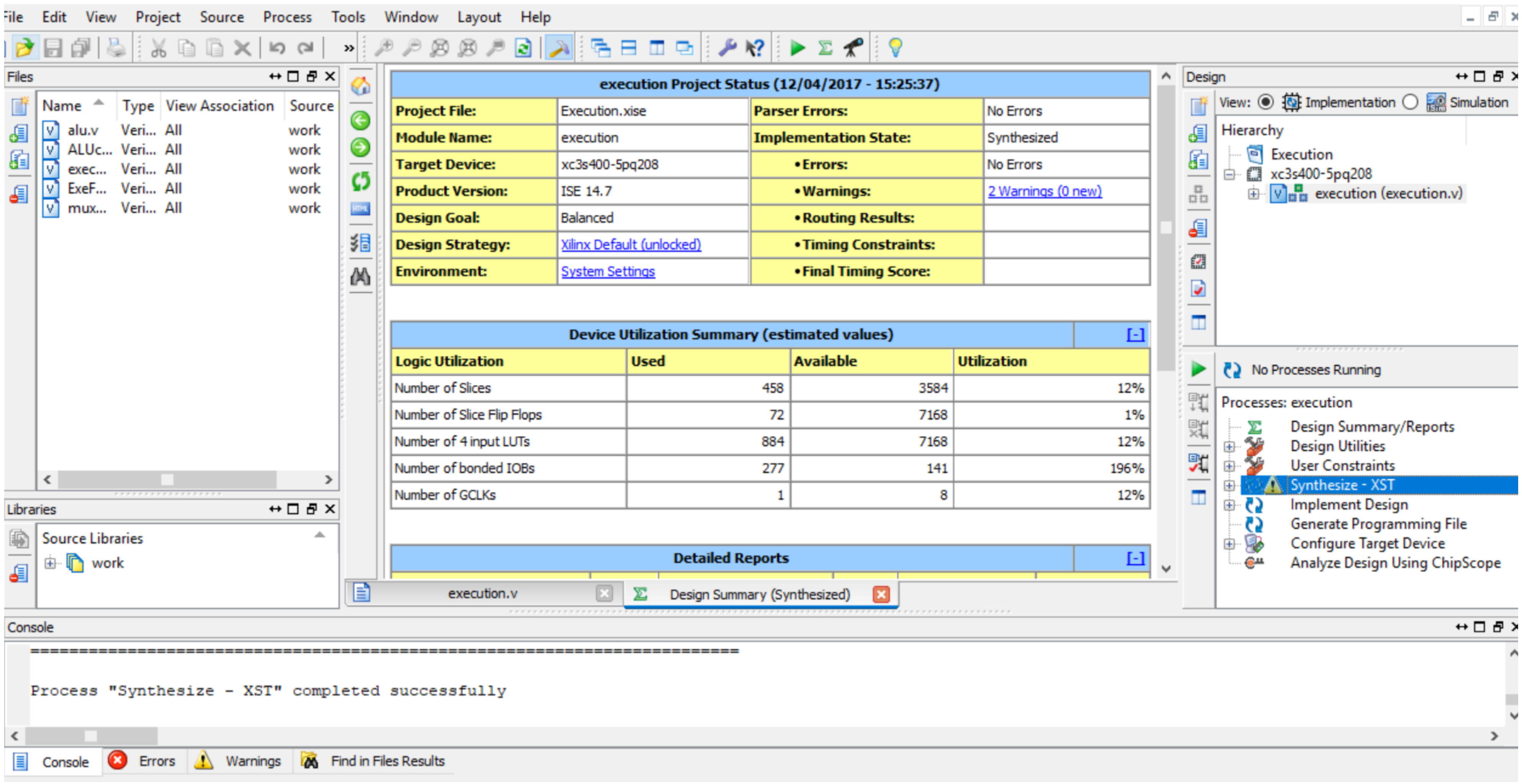
Hierarchy
- Decode
- xc3s400-5pq208
  - Decode (Decode.v)
    - mux3To1 (mux3To1.v)
    - mux3To1_5bits (mux3To1.v)
    - mux5bits (mux.v)
    - mux8bits (mux.v)

No Processes Running

Processes: Decode
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

```
================================================================================

Process "Synthesize - XST" completed successfully
```

3. Execution Stage:



Name	Type	View Association	Source
alu.v	Veri...	All	work
ALUc...	Veri...	All	work
exec...	Veri...	All	work
ExeF...	Veri...	All	work
mux...	Veri...	All	work

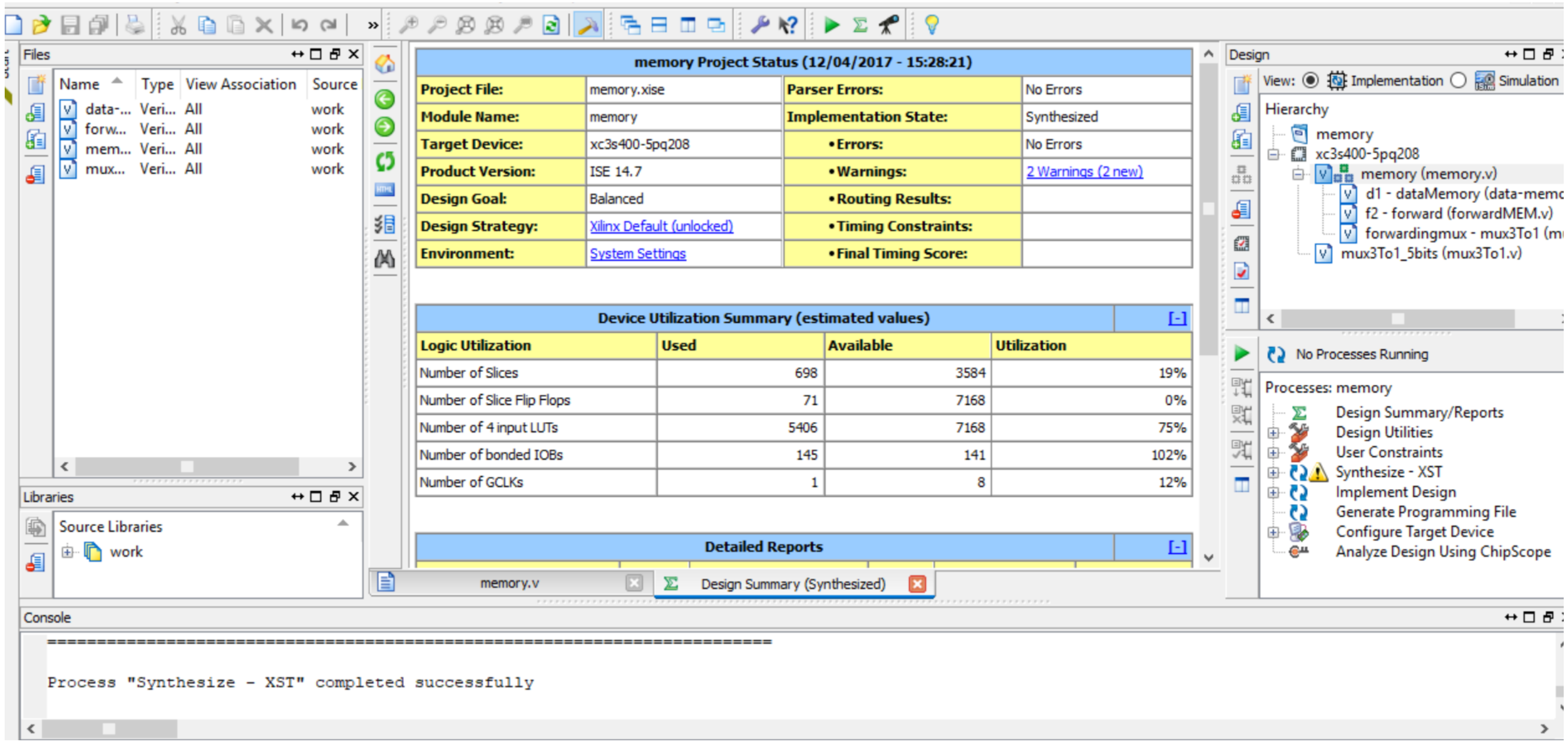
execution Project Status (12/04/2017 - 15:25:37)			
Project File:	Execution.xise	**Parser Errors:**	No Errors
Module Name:	execution	**Implementation State:**	Synthesized
Target Device:	xc3s400-5pq208	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	2 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	458	3584	12%
Number of Slice Flip Flops	72	7168	1%
Number of 4 input LUTs	884	7168	12%
Number of bonded IOBs	277	141	196%
Number of GCLKs	1	8	12%

Detailed Reports [-]

```
==========================================================================

Process "Synthesize - XST" completed successfully
```

4. Memory Stage:



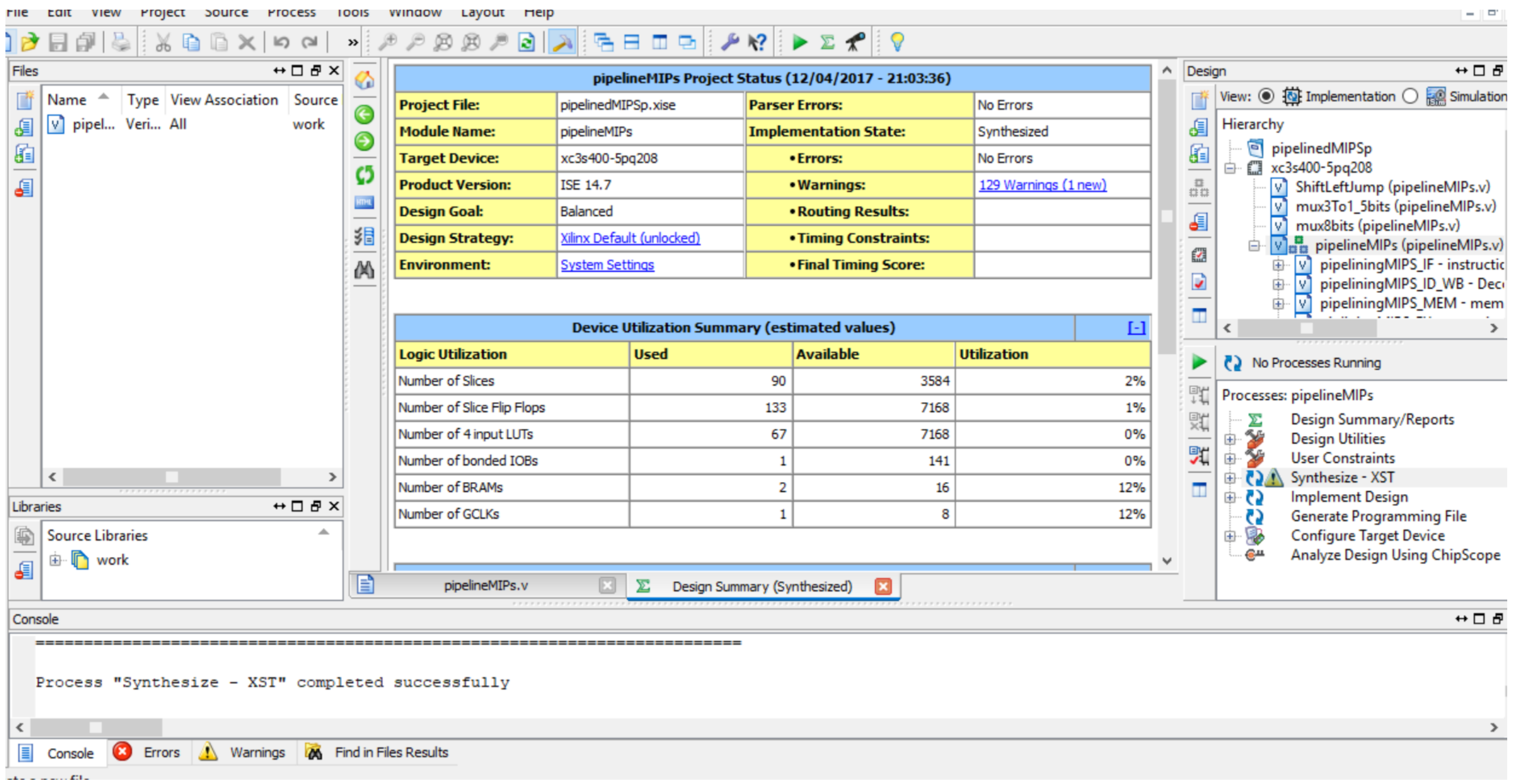
memory Project Status (12/04/2017 - 15:28:21)			
Project File:	memory.xise	**Parser Errors:**	No Errors
Module Name:	memory	**Implementation State:**	Synthesized
Target Device:	xc3s400-5pq208	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	2 Warnings (2 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	698	3584	19%
Number of Slice Flip Flops	71	7168	0%
Number of 4 input LUTs	5406	7168	75%
Number of bonded IOBs	145	141	102%
Number of GCLKs	1	8	12%

Detailed Reports [-]

```
==========================================================================

Process "Synthesize - XST" completed successfully
```

5. MIPS Processor


File Edit View Project Source Process Tools Window Layout Help
Files
Name Type View Association Source
pipel... Veri... All work
Libraries
Source Libraries
work
Design
View: Implementation Simulation
Hierarchy
pipelinedMIPSp
xc3s400-5pq208
ShiftLeftJump (pipelineMIPs.v)
mux3To1_5bits (pipelineMIPs.v)
mux8bits (pipelineMIPs.v)
pipelineMIPs (pipelineMIPs.v)
pipeliningMIPS_IF - instructic
pipeliningMIPS_ID_WB - Dec
pipeliningMIPS_MEM - mem
No Processes Running
Processes: pipelineMIPs
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
pipelineMIPs.v
Design Summary (Synthesized)
Console
Errors
Warnings
Find in Files Results

pipelineMIPs Project Status (12/04/2017 - 21:03:36)

Project File:	pipelinedMIPSp.xise	**Parser Errors:**	No Errors
Module Name:	pipelineMIPs	**Implementation State:**	Synthesized
Target Device:	xc3s400-5pq208	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	129 Warnings (1 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	90	3584	2%
Number of Slice Flip Flops	133	7168	1%
Number of 4 input LUTs	67	7168	0%
Number of bonded IOBs	1	141	0%
Number of BRAMs	2	16	12%
Number of GCLKs	1	8	12%

```
==================================================================

Process "Synthesize - XST" completed successfully
```

Used Libraries:

This is a list of all open-source used libraries

Icarus Verilog compiler and simulator
Python ttkthemes