گزارش سوال یک

پروژه جبرانی پایانترم

گزارش STACK_BASED_ALU

این پروژه به صورت انفرادی زده شده
محمدپویان شمس‌الدین 40111082

الف)

کد زده شده:

```
Ln#
 1  module STACK_BASED_ALU(
 2          input clk,
 3          input signed[31:0] input_data,
 4          input[2:0] opcode,
 5          output reg signed[31:0] output_data,
 6          output reg overflow,
 7          output reg invalid
 8  );
 9          reg signed[31:0] stack[1023:0];
10
11          integer i, index, temp;
12          reg negative;
13
14          initial begin
15                  output_data = 0;
16                  overflow = 0;
17                  invalid = 0;
18                  index = 0;
19                  for (i = 0; i < 1024; i = i + 1)
20                          stack[i] = 0;
21          end
22
23          always @ (posedge clk) begin
24                  if (opcode[2] == 1) begin
25                          if (opcode == 3'b100) begin
26                                  if (index < 2) begin
27                                          invalid = 1;
28                                          output_data = 0;
29                                          overflow = 0;
30                                  end
31                                  else begin
32                                          invalid = 0;
33                                          output_data = stack[index - 1] + stack[index - 2];
34                                          overflow = (((stack[index - 1] > 0) == (stack[index - 2] > 0)) && (output_data > 0 != stack[index - 1] > 0));
35                                  end
36                          end
37                          else if (opcode == 3'b101) begin
38                                  if (index < 2) begin
39                                          invalid = 1;
40                                          output_data = 0;
41                                          overflow = 0;
42                                  end
43                                  else begin
44                                          invalid = 0;
45                                          output_data = 0;
46                                          overflow = 0;
47                                          if (stack[index - 2] != 0) begin
48                                                  negative = stack[index - 2] < 0;
49                                                  if (negative)
50                                                          stack[index - 2] = -stack[index - 2];
51                                                  for (i = 0; i < stack[index - 2]; i = i + 1) begin
52                                                          temp = stack[index - 1] + output_data;
53                                                          overflow = overflow | (((stack[index - 1] > 0) == (output_data > 0)) && (temp > 0 != output_data > 0));
54                                                          output_data = temp;
55                                                  end
56                                                  if (negative) begin
57                                                          stack[index - 2] = -stack[index - 2];
58                                                          output_data = -output_data;
59                                                  end
60                                          end
61                                  end
62                          end
63                          else if (opcode == 3'b110) begin
64                                  output_data = 0;
65                                  overflow = 0;
66                                  if (index < 1024) begin
67                                          stack[index] = input_data;
68                                          index = index + 1;
69                                          invalid = 0;
70                                  end
71                                  else
72                                          invalid = 1;
73                          end
74                          else if (opcode == 3'b111) begin
75                                  output_data = 0;
76                                  overflow = 0;
77                                  if (index > 0) begin
78                                          index = index - 1;
79                                          invalid = 0;
80                                  end
81                                  else
82                                          invalid = 1;
83                          end
84                  end
85          end
86  endmodule
87
```

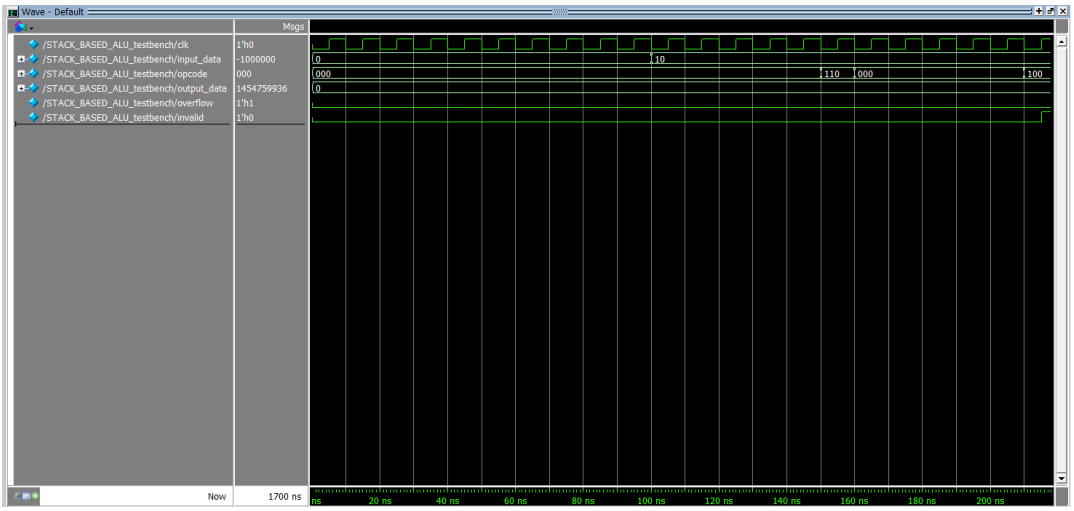
کد test bench:

```
module STACK_BASED_ALU_testbench;
	reg clk;
	reg signed [31:0] input_data;
	reg [2:0] opcode;

	wire signed [31:0] output_data;
	wire overflow;
	wire invalid;

	STACK_BASED_ALU uut(
		.clk(clk),
		.input_data(input_data),
		.opcode(opcode),
		.output_data(output_data),
		.overflow(overflow),
		.invalid(invalid)
	);

	initial clk = 0;

	always #5 clk = ~clk;

	initial begin
		input_data = 0;
		opcode = 0;

		// Wait 10 pulses for initialization
		#100;

		input_data = 10;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 100;
		#10;
		opcode = 0;
		#50;
		input_data = 22;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 100;
		#10;
		opcode = 0;
		#50;
		input_data = 2000000000;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 100;
		#10;
		opcode = 0;
		#50;
		opcode = 111;
		#10;
		opcode = 0;
		#50;
		opcode = 111;
		#10;
		opcode = 0;
		#50;
		opcode = 111;
		#10;
		opcode = 0;
		#50;
		opcode = 111;
		#10;
		opcode = 0;
		#50;
		input_data = -3;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 101;
		#10;
		opcode = 0;
		#50;
		input_data = -5;
		#50;
		opcode = 110;
		#10;
		opcode = 0;
		#50;
		opcode = 101;
		#10;
		opcode = 0;
		#50;
		input_data = 2000000;
```

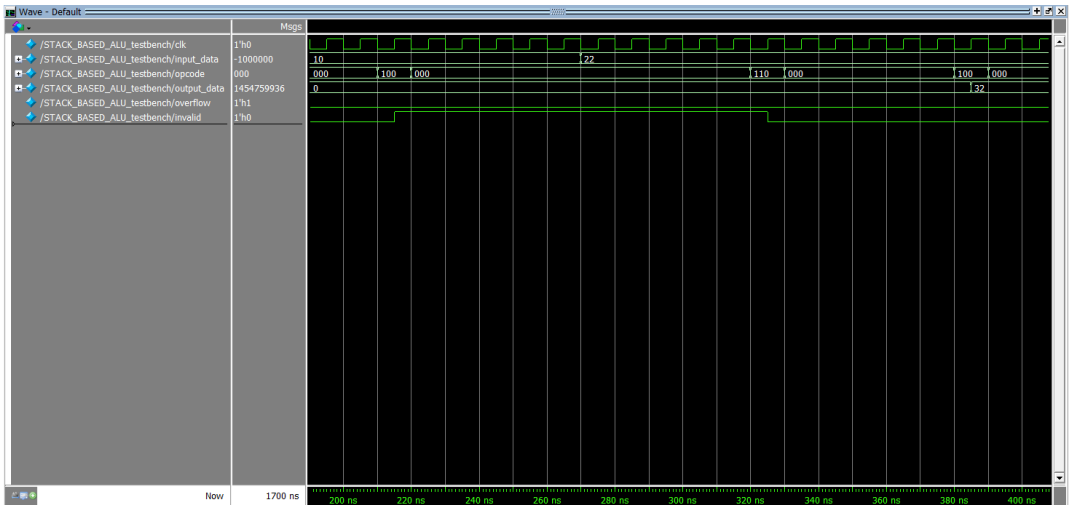
```
105                 #50;
106                 opcode = 110;
107                 #10;
108                 opcode = 0;
109                 #50;
110                 input_data = -1000000;
111                 #50;
112                 opcode = 110;
113                 #10;
114                 opcode = 0;
115                 #50;
116                 opcode = 101;
117                 #10;
118                 opcode = 0;
119                 #50;
120         end
121 endmodule
122
```

اجرای test bench:

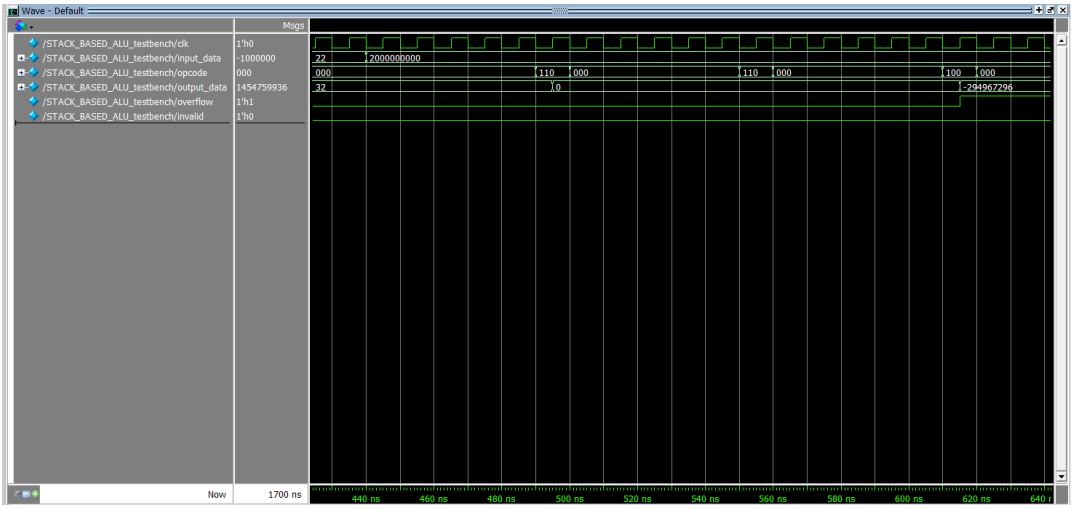
در ابتدا مقادیر را صفر تنظیم می‌کنیم. سپس عدد 10 را push می‌کنیم. بعد عمل Add انجام می‌دهیم، که به دلیل اینکه فقط یک عدد داریم خروجی invalid بدست می‌آید. بعد عدد 22 را push می‌کنیم. سپس مجددا عمل Add انجام می‌دهیم و این بار با موفقیت عدد 32 در خروجی ظاهر می‌شود. در ادامه عدد 2,000,000,000 را دو مرتبه push می‌کنیم. بعد عمل Add انجام می‌دهیم و خروجی به همراه overflow را مشاهده می‌کنیم. سپس 4 بار عمل pop انجام می‌دهیم تا stack خالی شود. بعد یک بار دیگر عمل pop انجام می‌دهیم که به دلیل خالی بودن stack خروجی invalid بدست می‌آید. در ادامه عدد 3- را push می‌کنیم. سپس عمل Multiply انجام می‌دهیم، که به دلیل اینکه فقط یک عدد داریم خروجی invalid بدست می‌آید. بعد عدد 5- را push می‌کنیم. سپس مجددا عمل Multiply انجام می‌دهیم و این بار با موفقیت عدد 15 در خروجی ظاهر می‌شود. در ادامه عدد 2,000,000 و بعد عدد 1,000,000- را push می‌کنیم. بعد عمل Multiply انجام می‌دهیم و خروجی به همراه overflow را مشاهده می‌کنیم و برنامه به پایان می‌رسد.



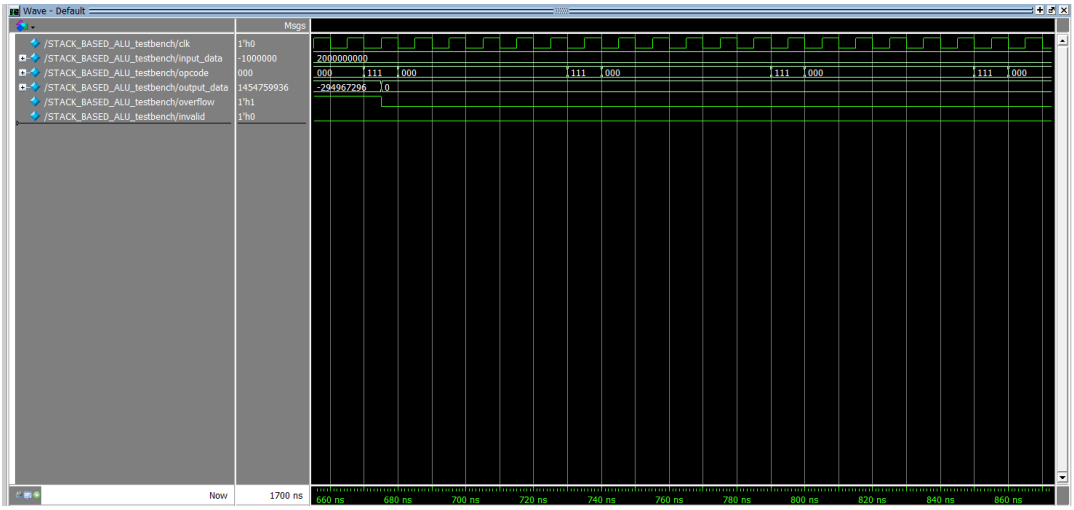
تنظیم اولیه و push کردن 10



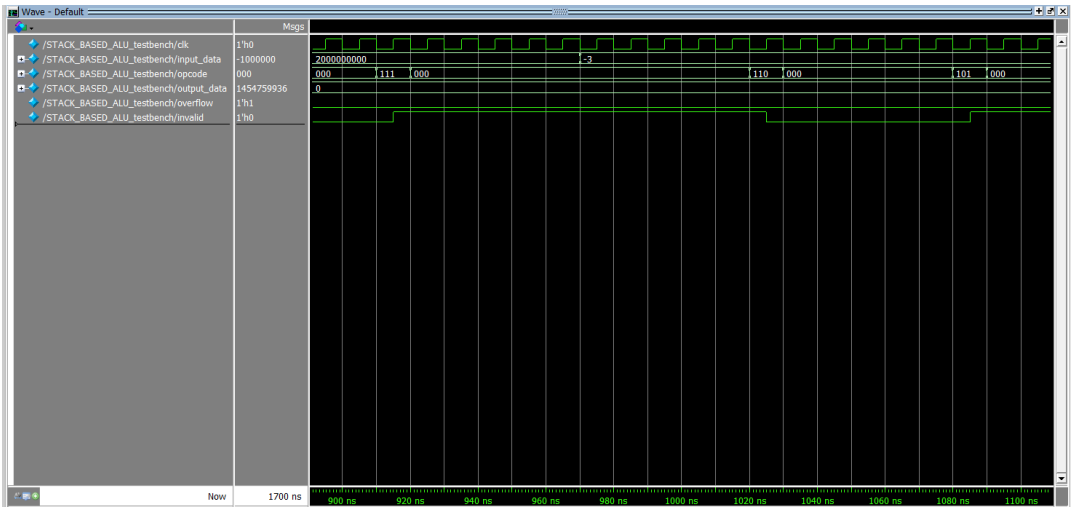
انجام Add و مشاهده invalid، push کردن 22، انجام Add و مشاهده خروجی 32



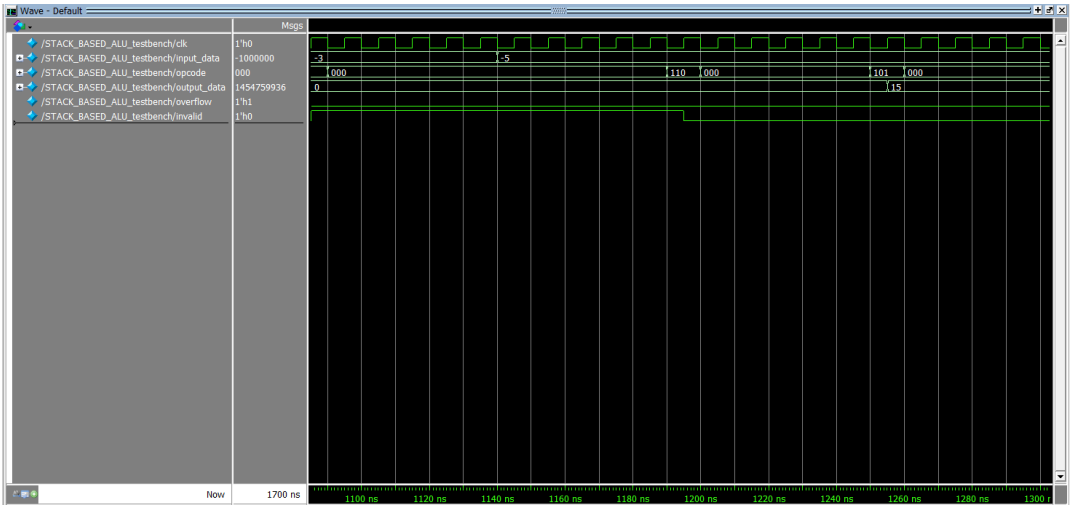
دو مرتبه push کردن 2,000,000,000، انجام Add و مشاهده overflow



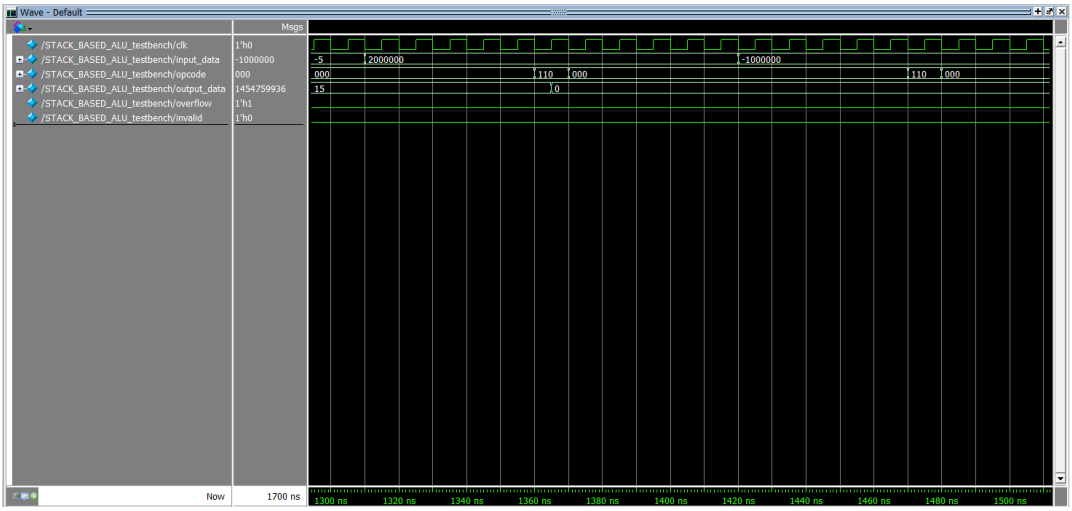
چهار مرتبه pop کردن تا خالی شدن stack



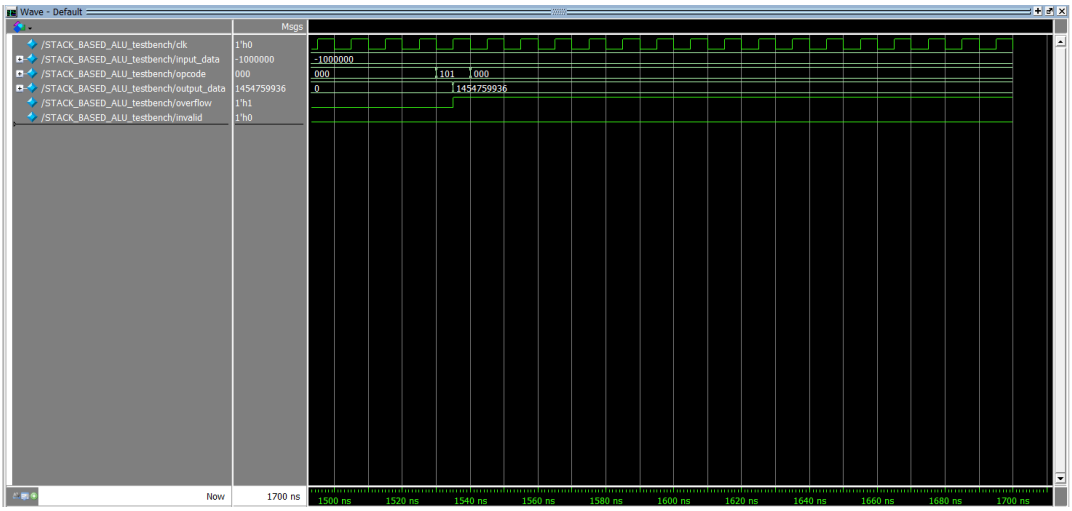
pop کردن مجدد و مشاهده invalid، push کردن 3-، انجام Multiply و مشاهده invalid



push کردن 5-، انجام Multiply و مشاهده خروجی 15



push کردن 2,000,000 و 1,000,000-



انجام Multiply، مشاهده overflow و اتمام برنامه