Digital Design for DSP & Communication

EE-278

Mini Project- III

128 – tap Digital FIR filter

Under supervision of

Dr. Charles Chang Choo

By

Dhatri Patel

SJSU ID: 010692526

Verilog Code

```
`timescale 1ps/1ps
module top_module
(
input clk,
input rst,
input start_26,
output reg signed[15:0] result_final_26
);
wire signed[15:0] result_26;
wire [6:0]address_26;
circbuff_26 c1(.clk(clk),.rst(rst),.start_26(start_26),.address_26(address_26));
Mp3_26 m1 (.clk(clk),.rst(rst),.address_26(address_26),.start_26(start_26),.result_26(result_26));
always @(posedge clk)
begin
result_final_26 <= #1 result_26;
end
endmodule
```

//**** Design Module *********

```
`timescale 1ps/1ps
module minip3_26 (clk, rst, address_26, result_26, start_26);
parameter width=16;
input clk, rst;
input [6:0] address_26;
input start_26;
output reg signed [width-1:0] result_26;
wire signed [width-1:0] data_x_26;
wire signed [width:0] data_a_26;
wire signed [(width*2)-1:0] mult_26;
wire signed [width-1:0] accumulator_26;
wire carry1_26;
wire signed [width-1:0] add_1_26;
wire overflow;
reg start_1_26, start_2_26;

// first ram memory

altera_ram x1 (.address(address_26),.clk(clk),.data(16'h00),.wren(1'b0),.q(data_x_26));
```

```
altera_ram a1 (.address(address_26),.clk(clk),.data(16'h00), .wren(1'b0),.q(data_a_26));
altera_mult m1 (.data_a(data_x_26),.data_b(data_a_26),.result(mult_26));
// altera addition module
altera_add z1
(.dataa(add_1_26), .datab(result_26), .cout(carry1_26), .overflow(overflow), .result(accumulator_26));
assign add_1_26 = (mult_26[14]) ?  (mult_26[30:15] + 1'b1) : (mult_26[30:15]);
always @(posedge clk or posedge rst)
begin
start_1_26 <= #1 start_26; // flag signal
start_2_26 <= #1 start_1_26;
if(rst)
begin
result_26 <= #1 16'h0000;
end
else begin
if(start_2_26)begin

if (overflow) begin
result_26 <= #1 16'hFFFF;
end
else begin
result_26 <= #1 accumulator_26;
end
end
else begin
result_26 <= 16'h0000;
end
end
end
endmodule
```

//** Circular buffer ****

```
`timescale 1ps/1ps
module circbuff_26(
input clk,
input rst,
input start_26,
output reg [6:0] address_26);
wire signed [6:0] start_c_26;
assign start_c_26 = (start_26) ? (address_26 + 7'h01) : address_26;
always @ (posedge clk)
begin
if(rst) begin
address_26 <= #1 7'h00;
end
```

```
else begin
address_26 <= #1 start_c_26;
end
end
endmodule
```

//** RAM module ****

```
`timescale 1 ps / 1 ps
// synopsys translate_on
module altera_ram (
        address,
        clk,
        data,
        wren,
        q);

        input    [6:0]  address;
        input      clk;
        input    [15:0]  data;
        input      wren;
        output   [15:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
        tri1       clk;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

        wire [15:0] sub_wire0;
        wire [15:0] q = sub_wire0[15:0];

        altsyncram      altsyncaltera_ramcomponent (
                                .address_a (address),
                                .clock0 (clock),
                                .data_a (data),
                                .wren_a (wren),
                                .q_a (sub_wire0),
                                .aclr0 (1'b0),
                                .aclr1 (1'b0),
                                .address_b (1'b1),
                                .addressstall_a (1'b0),
                                .addressstall_b (1'b0),
                                .byteena_a (1'b1),
                                .byteena_b (1'b1),
```

```
                                .clock1 (1'b1),
                                .clocken0 (1'b1),
                                .clocken1 (1'b1),
                                .clocken2 (1'b1),
                                .clocken3 (1'b1),
                                .data_b (1'b1),
                                .eccstatus (),
                                .q_b (),
                                .rden_a (1'b1),
                                .rden_b (1'b1),
                                .wren_b (1'b0));
        defparam
                altsyncaltera_ramcomponent.clock_enable_input_a = "BYPASS",
                altsyncaltera_ramcomponent.clock_enable_output_a = "BYPASS",
                altsyncaltera_ramcomponent.init_file = "",
                altsyncaltera_ramcomponent.intended_device_family = "Cyclone V",
                altsyncaltera_ramcomponent.lpm_hint = "ENABLE_RUNTIME_MOD=NO",
                altsyncaltera_ramcomponent.lpm_type = "altsyncram",
                altsyncaltera_ramcomponent.numwords_a = 128,
                altsyncaltera_ramcomponent.operation_mode = "SINGLE_PORT",
                altsyncaltera_ramcomponent.outdata_aclr_a = "NONE",
                altsyncaltera_ramcomponent.outdata_reg_a = "UNREGISTERED",
                altsyncaltera_ramcomponent.power_up_uninitialized = "FALSE",
                altsyncaltera_ramcomponent.read_during_write_mode_port_a =
"NEW_DATA_NO_NBE_READ",
                altsyncaltera_ramcomponent.widthad_a = 7,
                altsyncaltera_ramcomponent.width_a = 16,
                altsyncaltera_ramcomponent.width_byteena_a = 1;


endmodule
```

//** Multiplication module ****

```
`timescale 1 ps / 1 ps
module altera_mult (
        data_a,
        data_b,
        result);

        input    [8:0]  data_a;
        input    [8:0]  data_b;
        output   [17:0]  result;

        wire [17:0] sub_wire0;
```

```
	wire [17:0] result = sub_wire0[17:0];

	lpm_mult	lpm_mult_component (
				.dataa (data_a),
				.datab (data_b),
				.result (sub_wire0),
				.aclr (1'b0),
				.clken (1'b1),
				.clock (1'b0),
				.sclr (1'b0),
				.sum (1'b0));
	defparam
		lpm_mult_component.lpm_hint = "MAXIMIZE_SPEED=5",
		lpm_mult_component.lpm_representation = "SIGNED",
		lpm_mult_component.lpm_type = "LPM_MULT",
		lpm_mult_component.lpm_widtha = 9,
		lpm_mult_component.lpm_widthb = 9,
		lpm_mult_component.lpm_widthp = 18;


endmodule
```

//** Addition module *******

```
`timescale 1 ps / 1 ps
module altera_add (
	dataa,
	datab,
	cout,
	overflow,
	result);

	input	[8:0]  dataa;
	input	[8:0]  datab;
	output	 cout;
	output	 overflow;
	output	[8:0]  result;

	wire  sub_wire0;
	wire  sub_wire1;
	wire [8:0] sub_wire2;
	wire  cout = sub_wire0;
	wire  overflow = sub_wire1;
	wire [8:0] result = sub_wire2[8:0];

	lpm_add_sub	LPM_ADD_SUB_component (
```

```
                                .dataa (dataa),
                                .datab (datab),
                                .cout (sub_wire0),
                                .overflow (sub_wire1),
                                .result (sub_wire2)
                                // synopsys translate_off
                                ,
                                .aclr (),
                                .add_sub (),
                                .cin (),
                                .clken (),
                                .clock ()
                                // synopsys translate_on
                                );
        defparam
                LPM_ADD_SUB_component.lpm_direction = "ADD",
                LPM_ADD_SUB_component.lpm_hint =
"ONE_INPUT_IS_CONSTANT=NO,CIN_USED=NO",
                LPM_ADD_SUB_component.lpm_representation = "SIGNED",
                LPM_ADD_SUB_component.lpm_type = "LPM_ADD_SUB",
                LPM_ADD_SUB_component.lpm_width = 9;


endmodule
```

//**** 4 MAC Design Module ****

```
`timescale 1ps/1ps
module Mp3_26 (clk, rst, address_26, result_26, start_26);
parameter width=16;
input clk, rst;
input [6:0] address_26;
input start_26;
output reg signed[15:0] result_26;
reg signed [15:0] result_1_26, result_2_26, result_3_26, result_4_26;
wire signed [15:0] data_x1_26, data_x2_26, data_x3_26, data_x4_26;
wire signed [15:0] data_a1_26, data_a2_26, data_a3_26, data_a4_26;
wire signed [31:0] mult_1_26, mult_2_26, mult_3_26, mult_4_26;
wire signed[15:0] accumulator1_26, accumulator2_26, accumulator3_26, accumulator4_26;
wire carry1_26, carry2_26, carry3_26, carry4_26;
wire signed [15:0] add_1_26, add_2_26, add_3_26, add_4_26;
wire overflow1,overflow2,overflow3,overflow4;
reg start_1_26, start_2_26, start_3_26, start_4_26, start_5_26;
altera_ramx1 x1 (.address(address_26),
```

```
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x1_26));

altera_ramx2 x2 (.address(address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x2_26));

altera_ramx3 x3 (.address(address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x3_26));

altera_ramx4 x4 (.address(address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x4_26));

altera_rama1 a1 (.address(7'd31-address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a1_26));

altera_rama2 a2 (.address(7'd31-address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a2_26));

altera_rama3 a3 (.address(7'd31-address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a3_26));

altera_rama4 a4 (.address(7'd31-address_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a4_26));
```

```
altera_mult m1 (.data_a(data_x1_26),
.data_b(data_a1_26),
.result(mult_1_26));

altera_mult m2 (.data_a(data_x2_26),
.data_b(data_a2_26),
.result(mult_2_26));

altera_mult m3 (.data_a(data_x3_26),
.data_b(data_a3_26),
.result(mult_3_26));

altera_mult m4 (.data_a(data_x4_26),
.data_b(data_a4_26),
.result(mult_4_26));

altera_add DUT1 (.dataa(add_1_26),
.datab(result_1_26),
.cout(carry1_26),
.overflow(overflow1),
.result(accumulator1_26));

altera_add DUT2 (.dataa(add_2_26),
.datab(result_2_26),
.cout(carry2_26),
.overflow(overflow2),
.result(accumulator2_26));

altera_add DUT3 (.dataa(add_3_26),
.datab(result_3_26),
.cout(carry3_26),
.overflow(overflow3),
.result(accumulator3_26));

altera_add DUT4 (.dataa(add_4_26),
.datab(result_4_26),
.cout(carry4_26),
.overflow(overflow4),
.result(accumulator4_26));

assign add_1_26 = (mult_1_26[14]) ? (mult_1_26[30:15] + 1'b1) : (mult_1_26[30:15]);
assign add_2_26 = (mult_2_26[14]) ? (mult_2_26[30:15] + 1'b1) : (mult_2_26[30:15]);
assign add_3_26 = (mult_3_26[14]) ? (mult_3_26[30:15] + 1'b1) : (mult_3_26[30:15]);
assign add_4_26 = (mult_4_26[14]) ? (mult_4_26[30:15] + 1'b1) : (mult_4_26[30:15]);
```

```
always @(posedge clk or posedge rst)
begin
start_1_26<=#1 start_26;
start_2_26<=#1 start_1_26;
start_3_26<=#1 start_2_26;
start_4_26<=#1 start_3_26;
start_5_26<=#1 start_4_26;

if(rst)
begin
result_1_26 <=#1 16'h0000;
result_2_26 <=#1 16'h0000;
result_3_26 <=#1 16'h0000;
result_4_26 <=#1 16'h0000;
end
else begin
if(start_1_26) begin
result_1_26<= #1 accumulator1_26;
result_2_26<= #1 accumulator2_26;
result_3_26<= #1 accumulator3_26;
result_4_26<= #1 accumulator4_26;
end
else begin
result_1_26<= #1 16'h0000;
result_2_26<= #1 16'h0000;
result_3_26<= #1 16'h0000;
result_4_26<= #1 16'h0000;
end

if(overflow1 || overflow2 ||overflow3 || overflow4)
begin
result_26 <=#1 16'hFFFF;
end
else begin
result_26 <=#1 accumulator1_26+accumulator2_26+accumulator3_26+accumulator4_26;
end
end
end
endmodule
```

//**** 16 MAC Design Module ****

```
`timescale 1ps/1ps
module Mp3_26 (clk, rst, addr_26, result_26, start_26);
parameter width=16;
input clock, rst;
input [6:0] addr_26;
input start_26;
output reg signed[15:0] result_26;
reg signed [15:0]
result_1_26,result_2_26,result_3_26,result_4_26,result_5_26,result_6_26,result_7_26,result_8
_26
,result_9_26,result_10_26,result_11_26,result_12_26,result_13_26,result_14_26,result_15_26,result_16_
26;
wire signed [15:0]
data_x1_26,data_x2_26,data_x3_26,data_x4_26,data_x5_26,data_x6_26,data_x7_26,
data_x8_26,data_x9_26,data_x10_26,data_x11_26,data_x12_26,data_x13_26,data_x14_26,data_x
15_26,data_x16_26;
wire signed [15:0]
data_a1_26,data_a2_26,data_a3_26,data_a4_26,data_a5_26,data_a6_26,data_a7_26,data_a8_26
,data_a9_26,data_a10_26,data_a11_26,data_a12_26,data_a13_26,data_a14_26,data_a15_26,data_a16_26
;
wire signed [31:0]
mult_1_26,mult_2_26,mult_3_26,mult_4_26,mult_5_26,mult_6_26,mult_7_26,mult_8_26
,mult_9_26,mult_10_26,mult_11_26,mult_12_26,mult_13_26,mult_14_26,mult_15_26,mult_16_26;
wire signed[15:0]
acc1_26,acc2_26,acc3_26,acc4_26,acc5_26,acc6_26,acc7_26,acc8_26,acc9_26,acc10_26
,acc11_26,acc12_26,acc13_26,acc14_26,acc15_26,acc16_26;
wire
carry1_26,carry2_26,carry3_26,carry4_26,carry5_26,carry6_26,carry7_26,carry8_26
,carry9_26,carry10_26,carry11_26,carry12_26,carry13_26,carry14_26,carry15_26,carry16_26;
wire signed [15:0]
add_1_26,add_2_26,add_3_26,add_4_26,add_5_26,add_6_26,add_7_26,add_8_26,add_9_26
,add_10_26,add_11_26,add_12_26,add_13_26,add_14_26,add_15_26,add_16_26;
wire overflow1,overflow2,overflow3,overflow4,overflow5,overflow6,overflow7,overflow8
,overflow9,overflow10,overflow11,overflow12,overflow13,overflow14,overflow15,overflow16;
reg start_1_26,start_2_26,start_3_26,start_4_26,start_5_26;
ram_x1 x1 (.address(7'd24+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x1_26));
ram_x2 x2 (.address(7'd16+addr_26),
.clk(clk),
.data(16'h00),
```

```
.wren(1'b0),.q(data_x2_26));
ram_x3 x3 (.address(7'd8+ addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x3_26));
ram_x4 x4 (.address(addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x4_26));
ram_x5 x5 (.address(7'd24+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x5_26));
ram_x6 x6 (.address(7'd16+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x6_26));
ram_x7 x7 (.address(7'd8+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x7_26));
ram_x8 x8 (.address(addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x8_26));
ram_x9 x9 (.address(7'd24+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x9_26));
ram_x10 x10 (.address(7'd16+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x10_26));
ram_x11 x11 (.address(7'd8+addr_26),.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x11_26));
ram_x12 x12 (.address(addr_26),
.clk(clk),
```

```
.data(16'h00),
.wren(1'b0),
.q(data_x12_26));
ram_x13 x13 (.address(7'd24+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x13_26));
ram_x14 x14 (.address(7'd16+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x14_26));
ram_x15 x15 (.address(7'd8+addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x15_26));
ram_x16 x16 (.address(addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_x16_26));
ram_a1 a1 (.address(7'd7-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a1_26));
ram_a2 a2 (.address(7'd15-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a2_26));
ram_a3 a3 (.address(7'd23-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),.q(data_a3_26));
ram_a4 a4 (.address(7'd31-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a4_26));
ram_a5 a5 (.address(7'd7-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a5_26));
```

```
ram_a6 a6 (.address(7'd15-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a6_26));
ram_a7 a7 (.address(7'd23-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a7_26));
ram_a8 a8 (.address(7'd31-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a8_26));
ram_a9 a9 (.address(7'd7-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a9_26));
ram_a10 a10 (.address(7'd15-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a10_26));
ram_a11 a26 (.address(7'd23-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a11_26));
ram_a12 a12 (.address(7'd31-addr_26),.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a12_26));
ram_a13 a13 (.address(7'd7-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a13_26));
ram_a14 a14 (.address(7'd15-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a14_26));
ram_a15 a15 (.address(7'd23-addr_26),
.clk(clk),
.data(16'h00),
```

```
.wren(1'b0),
.q(data_a15_26));
ram_a16 a16 (.address(7'd31-addr_26),
.clk(clk),
.data(16'h00),
.wren(1'b0),
.q(data_a16_26));

mult_proj m1 (.data_a(data_x1_26),
.data_b(data_a1_26),
.result(mult_1_26));
mult_proj m2 (.data_a(data_x2_26),
.data_b(data_a2_26),
.result(mult_2_26));
mult_proj m3 (.data_a(data_x3_26),
.data_b(data_a3_26),
.result(mult_3_26));
mult_proj m4 (.data_a(data_x4_26),
.data_b(data_a4_26),
.result(mult_4_26));
mult_proj m5 (.data_a(data_x5_26),
.data_b(data_a5_26),
.result(mult_5_26));
mult_proj m6 (.data_a(data_x6_26),
.data_b(data_a6_26),.result(mult_6_26));
mult_proj m7 (.data_a(data_x7_26),
.data_b(data_a7_26),
.result(mult_7_26));
mult_proj m8 (.data_a(data_x8_26),
.data_b(data_a8_26),
.result(mult_8_26));
mult_proj m9 (.data_a(data_x9_26),
.data_b(data_a9_26),
.result(mult_9_26));
mult_proj m10 (.data_a(data_x10_26),
.data_b(data_a10_26),
.result(mult_10_26));
mult_proj m11 (.data_a(data_x11_26),
.data_b(data_a11_26),
.result(mult_11_26));
mult_proj m12 (.data_a(data_x12_26),
.data_b(data_a12_26),
.result(mult_12_26));
mult_proj m13 (.data_a(data_x13_26),
.data_b(data_a13_26),
.result(mult_13_26));
mult_proj m14 (.data_a(data_x14_26),
```

```
.data_b(data_a14_26),
.result(mult_14_26));
mult_proj m15 (.data_a(data_x15_26),
.data_b(data_a15_26),
.result(mult_15_26));
mult_proj m16 (.data_a(data_x16_26),
.data_b(data_a16_26),
.result(mult_16_26));

add_proj DUT1 (.dataa(add_1_26),
.datab(result_1_26),
.cout(carry1_26),
.overflow(overflow1),
.result(acc1_26));
add_proj DUT2 (.dataa(add_2_26),
.datab(result_2_26),
.cout(carry2_26),.overflow(overflow2),
.result(acc2_26));
add_proj DUT3 (.dataa(add_3_26),
.datab(result_3_26),
.cout(carry3_26),
.overflow(overflow3),
.result(acc3_26));
add_proj DUT4 (.dataa(add_4_26),
.datab(result_4_26),
.cout(carry4_26),
.overflow(overflow4),
.result(acc4_26));
add_proj DUT5 (.dataa(add_5_26),
.datab(result_5_26),
.cout(carry5_26),
.overflow(overflow5),
.result(acc5_26));
add_proj DUT6 (.dataa(add_6_26),
.datab(result_6_26),
.cout(carry6_26),
.overflow(overflow6),
.result(acc6_26));
add_proj DUT7 (.dataa(add_7_26),
.datab(result_7_26),
.cout(carry7_26),
.overflow(overflow7),
.result(acc7_26));
add_proj DUT8 (.dataa(add_8_26),
.datab(result_8_26),
.cout(carry8_26),
.overflow(overflow8),
```

```
.result(acc8_26));
add_proj DUT9 (.dataa(add_9_26),
.datab(result_9_26),
.cout(carry9_26),
.overflow(overflow9),
.result(acc9_26));
add_proj DUT10 (.dataa(add_10_26),
.datab(result_10_26),
.cout(carry10_26),
.overflow(overflow10),
.result(acc10_26));
add_proj DUT11 (.dataa(add_11_26),
.datab(result_11_26),
.cout(carry11_26),
.overflow(overflow11),
.result(acc11_26));
add_proj DUT12 (.dataa(add_12_26),
.datab(result_12_26),
.cout(carry12_26),
.overflow(overflow12),
.result(acc12_26));
add_proj DUT13 (.dataa(add_13_26),
.datab(result_13_26),
.cout(carry13_26),
.overflow(overflow13),
.result(acc13_26));
add_proj DUT14 (.dataa(add_14_26),
.datab(result_14_26),
.cout(carry14_26),
.overflow(overflow14),
.result(acc14_26));
add_proj DUT15 (.dataa(add_15_26),
.datab(result_15_26),
.cout(carry15_26),
.overflow(overflow15),
.result(acc15_26));
add_proj DUT16 (.dataa(add_16_26),
.datab(result_16_26),
.cout(carry16_26),
.overflow(overflow16),
.result(acc16_26));
assign add_1_26 =(mult_1_26[14])? (mult_1_26[30:15] + 1'b1):(mult_1_26[30:15]);
assign add_2_26 =(mult_2_26[14])? (mult_2_26[30:15] + 1'b1):(mult_2_26[30:15]);
assign add_3_26 =(mult_3_26[14])? (mult_3_26[30:15] + 1'b1):(mult_3_26[30:15]);
assign add_4_26 =(mult_4_26[14])? (mult_4_26[30:15] + 1'b1):(mult_4_26[30:15]);
assign add_5_26 =(mult_5_26[14])? (mult_5_26[30:15] + 1'b1):(mult_5_26[30:15]);
assign add_6_26 =(mult_6_26[14])? (mult_6_26[30:15] + 1'b1):(mult_6_26[30:15]);
```

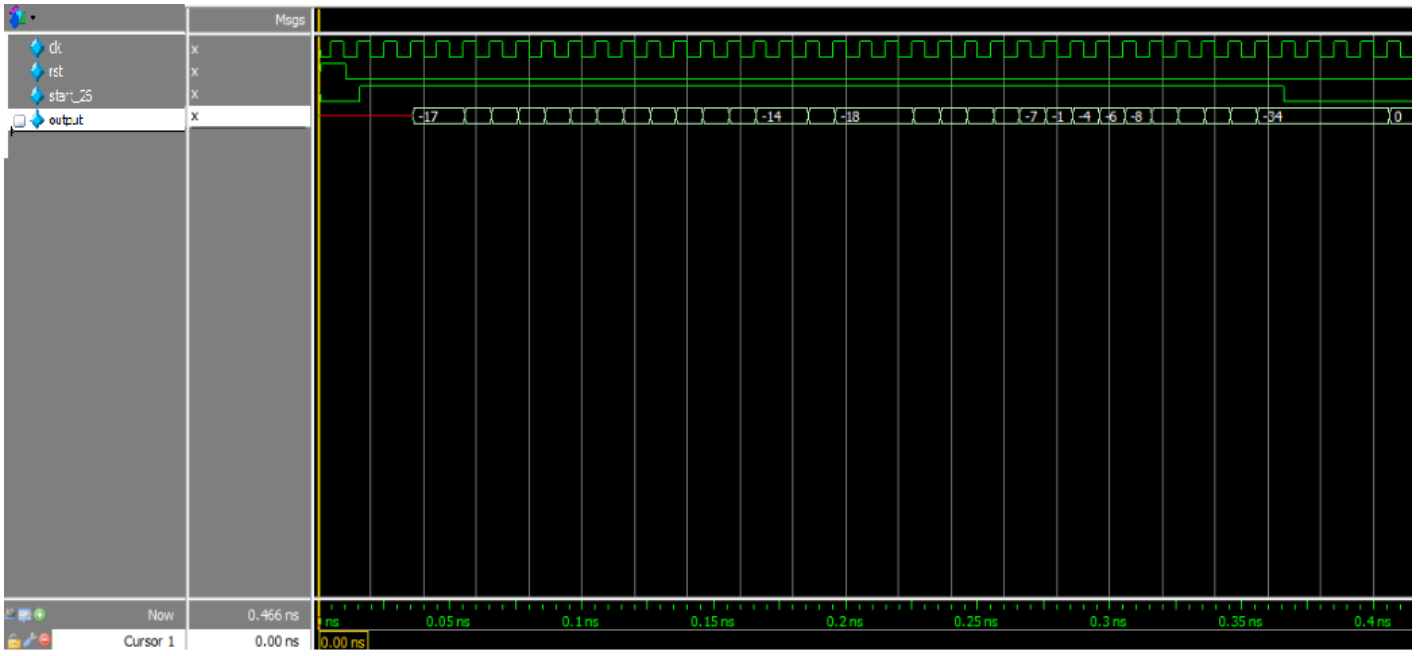
```
assign add_7_26 =(mult_7_26[14])? (mult_7_26[30:15] + 1'b1):(mult_7_26[30:15]);
assign add_8_26 =(mult_8_26[14])? (mult_8_26[30:15] + 1'b1):(mult_8_26[30:15]);
assign add_9_26 =(mult_9_26[14])? (mult_9_26[30:15] + 1'b1):(mult_9_26[30:15]);
assign add_10_26 =(mult_10_26[14])? (mult_10_26[30:15] + 1'b1):(mult_10_26[30:15]);
assign add_11_26 =(mult_11_26[14])? (mult_11_26[30:15] + 1'b1):(mult_11_26[30:15]);
assign add_12_26 =(mult_12_26[14])? (mult_12_26[30:15] + 1'b1):(mult_12_26[30:15]);
assign add_13_26 =(mult_13_26[14])? (mult_13_26[30:15] + 1'b1):(mult_13_26[30:15]);
assign add_14_26 =(mult_14_26[14])? (mult_14_26[30:15] + 1'b1):(mult_14_26[30:15]);
assign add_15_26 =(mult_15_26[14])? (mult_15_26[30:15] + 1'b1):(mult_15_26[30:15]);
assign add_16_26 =(mult_16_26[14])? (mult_16_26[30:15] + 1'b1):(mult_16_26[30:15]);

always @(posedge clk or posedge rst)
begin
start_1_26<=#1 start_26;
start_2_26<=#1 start_1_26;
start_3_26<=#1 start_2_26;
start_4_26<=#1 start_3_26;
start_5_26<=#1 start_4_26;
if(rst)
begin
result_1_26 <=#1 16'h0000;
result_2_26 <=#1 16'h0000;
result_3_26 <=#1 16'h0000;
result_4_26 <=#1 16'h0000;
result_5_26<= #1 16'h0000;
result_6_26<= #1 16'h0000;
result_7_26<= #1 16'h0000;
result_8_26<= #1 16'h0000;
result_9_26<= #1 16'h0000;
result_10_26<= #1 16'h0000;
result_11_26<= #1 16'h0000;
result_12_26<= #1 16'h0000;
result_13_26<= #1 16'h0000;
result_14_26<= #1 16'h0000;
result_15_26<= #1 16'h0000;
result_16_26<= #1 16'h0000;
end
else begin
if(start_1_26)
begin
result_1_26<= #1 acc1_26;
result_2_26<= #1 acc2_26;
result_3_26<= #1 acc3_26;
result_4_26<= #1 acc4_26;
result_5_26<= #1 acc5_26;
result_6_26<= #1 acc6_26;
result_7_26<= #1 acc7_26;
```

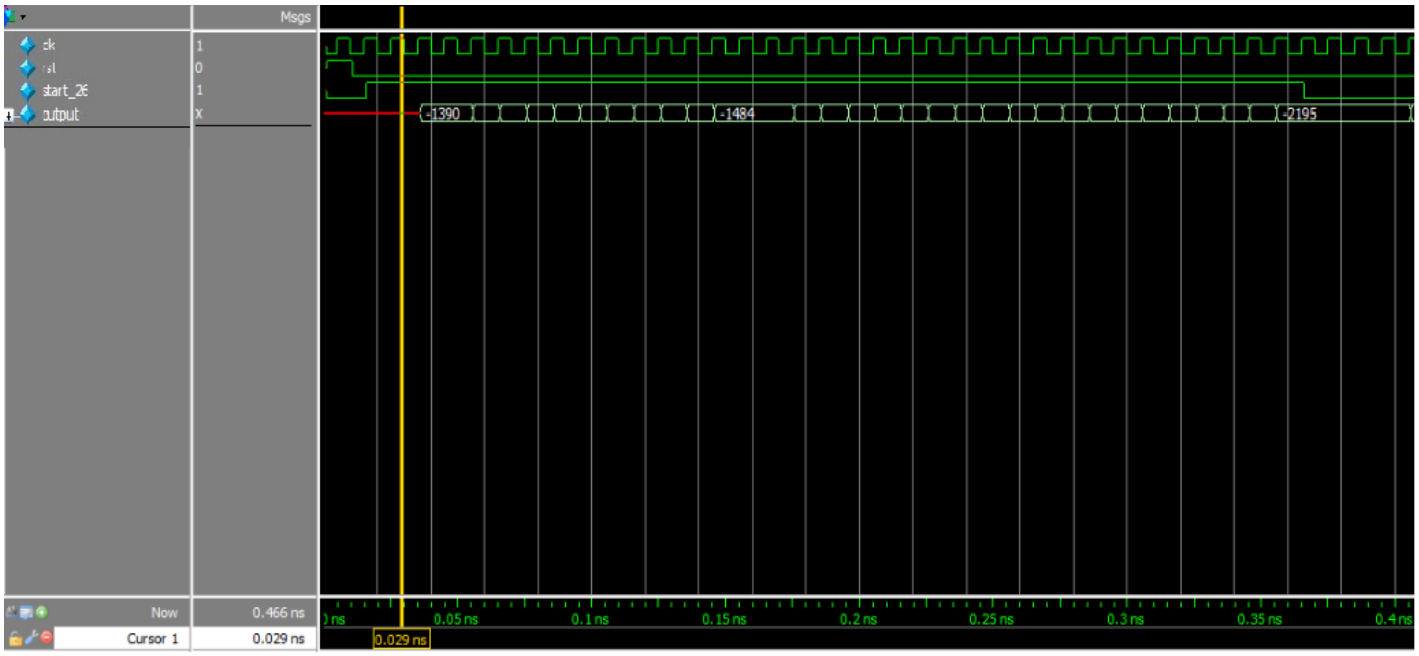
```
result_8_26<= #1 acc8_26;
result_9_26<= #1 acc9_26;
result_10_26<= #1 acc10_26;
result_11_26<= #1 acc11_26;
result_12_26<= #1 acc12_26;
result_13_26<= #1 acc13_26;
result_14_26<= #1 acc14_26;
result_15_26<= #1 acc15_26;
result_16_26<= #1 acc16_26;
end
else begin
result_1_26<= #1 16'h0000;
result_2_26<= #1 16'h0000;
result_3_26<= #1 16'h0000;
result_4_26<= #1 16'h0000;
result_5_26<= #1 16'h0000;
result_6_26<= #1 16'h0000;
result_7_26<= #1 16'h0000;
result_8_26<= #1 16'h0000;
result_9_26<= #1 16'h0000;
result_10_26<= #1 16'h0000;
result_11_26<= #1 16'h0000;
result_12_26<= #1 16'h0000;
result_13_26<= #1 16'h0000;
result_14_26<= #1 16'h0000;
result_15_26<= #1 16'h0000;
result_16_26<= #1 16'h0000;
end

if(overflow1 || overflow2 ||overflow3 || overflow4|| overflow5|| overflow6
|| overflow7|| overflow8|| overflow9|| overflow10|| overflow11|| overflow12
|| overflow13|| overflow14|| overflow15|| overflow16)
begin
result_26 <=#1 16'hFFFF;
end
else begin
result_26 <=#1 acc1_26+acc2_26+acc3_26+acc4_26+acc5_26+acc6_26+acc7_26
+acc8_26+acc9_26+acc10_26+acc11_26+acc12_26+acc13_26+acc14_26+acc15_26+
acc16_26;
end
end
end
endmodule
```

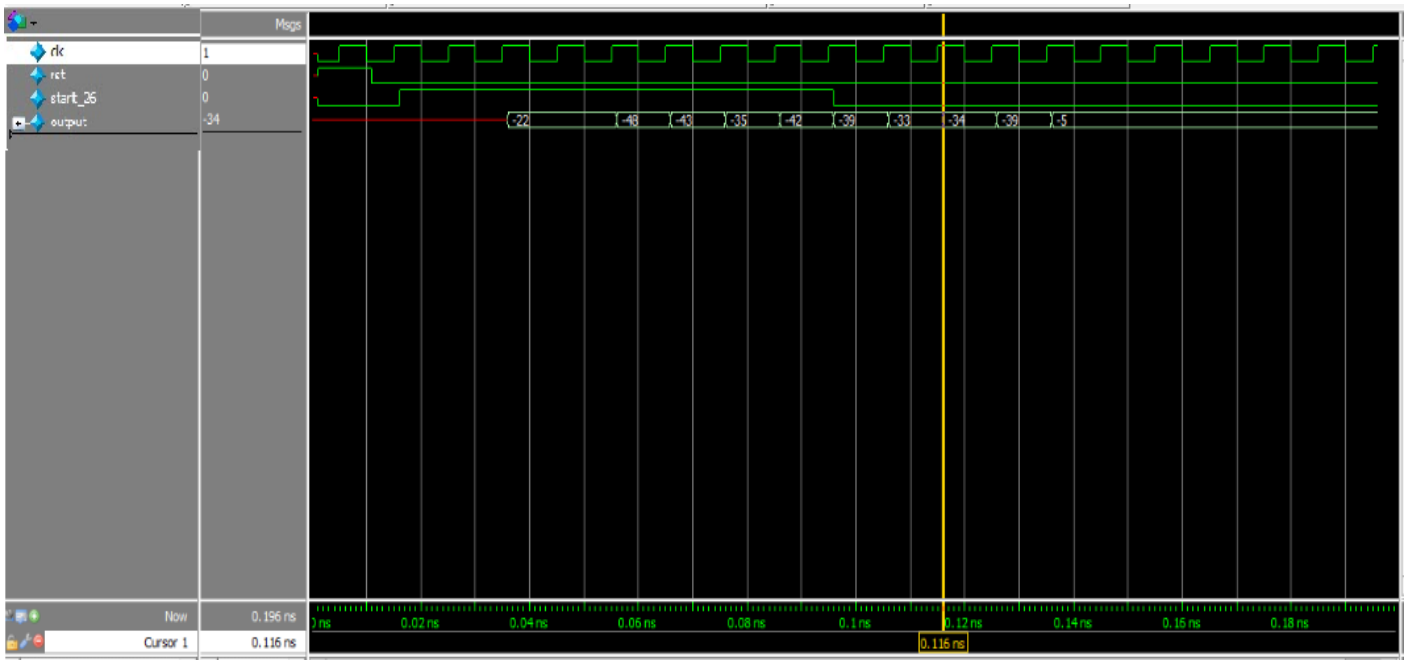
Test-bench

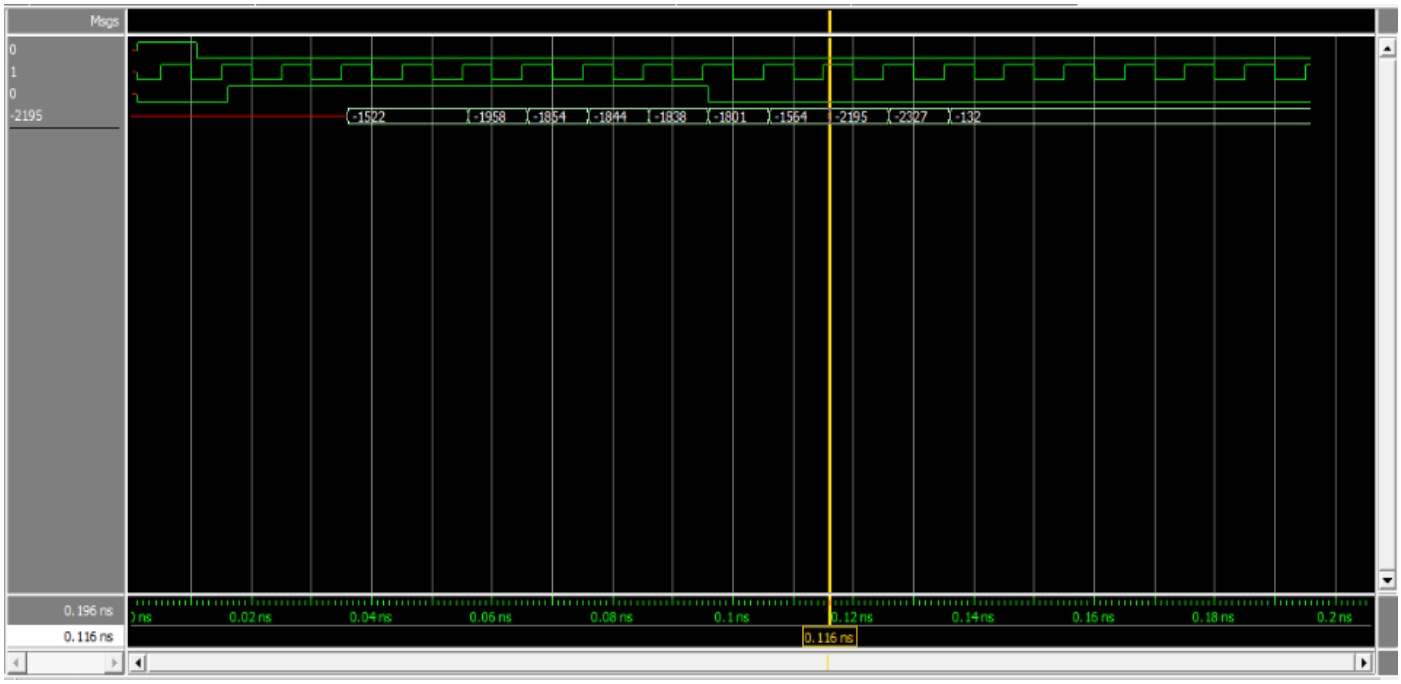
```
`timescale 1 ps / 1 ps

module mp3_tb();
parameter width=16;
reg clk,rst,start_26;
wire signed [15:0] output;

top_module  t1 (.clk(clk), .rst(rst), .start_26(start_26), .result_final_26(output));

initial begin
        #5 clock=~clock;
end

initial begin
#1;
clk = 1'b0;
rst = 1'b1;
start_26 = 1'b0;
#10
reset_26 = 1'b0;
#5
start_26 = 1'b1;
#80
start_26 = 1'b0;
#100;
$finish;
end
endmodule
```

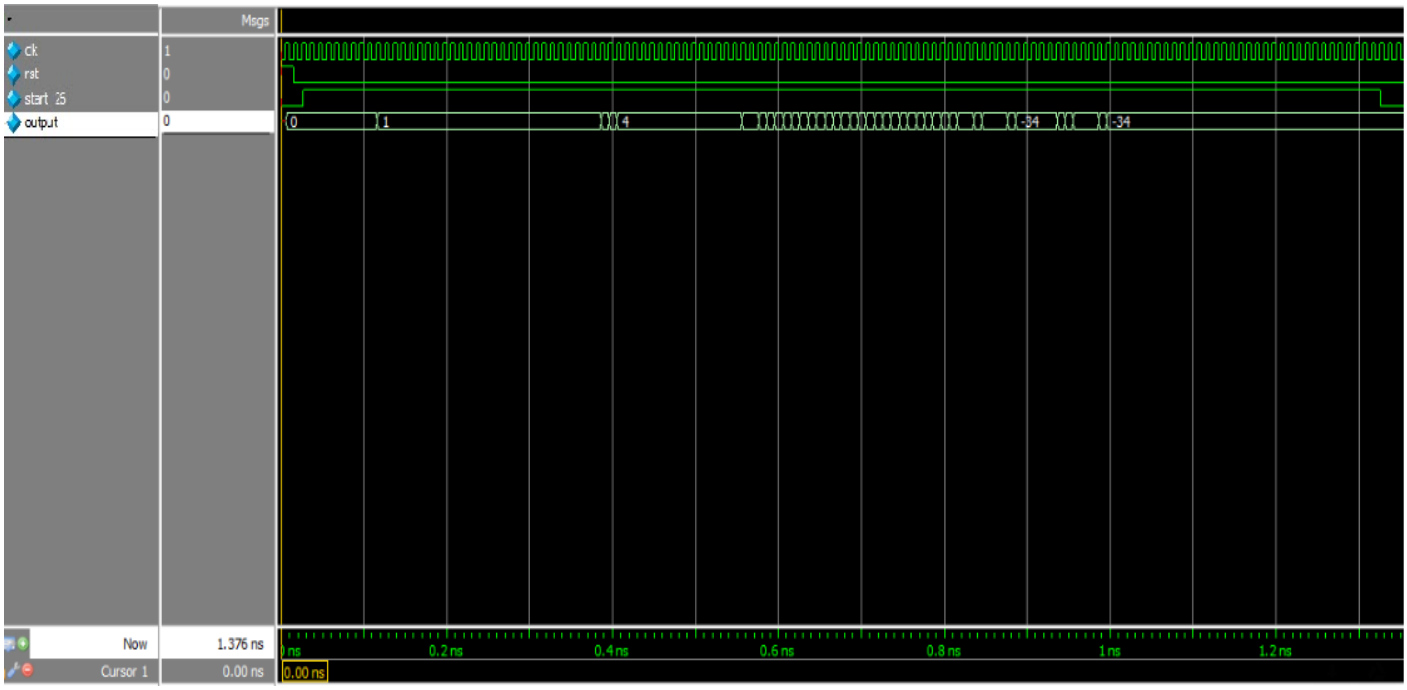
Waveforms

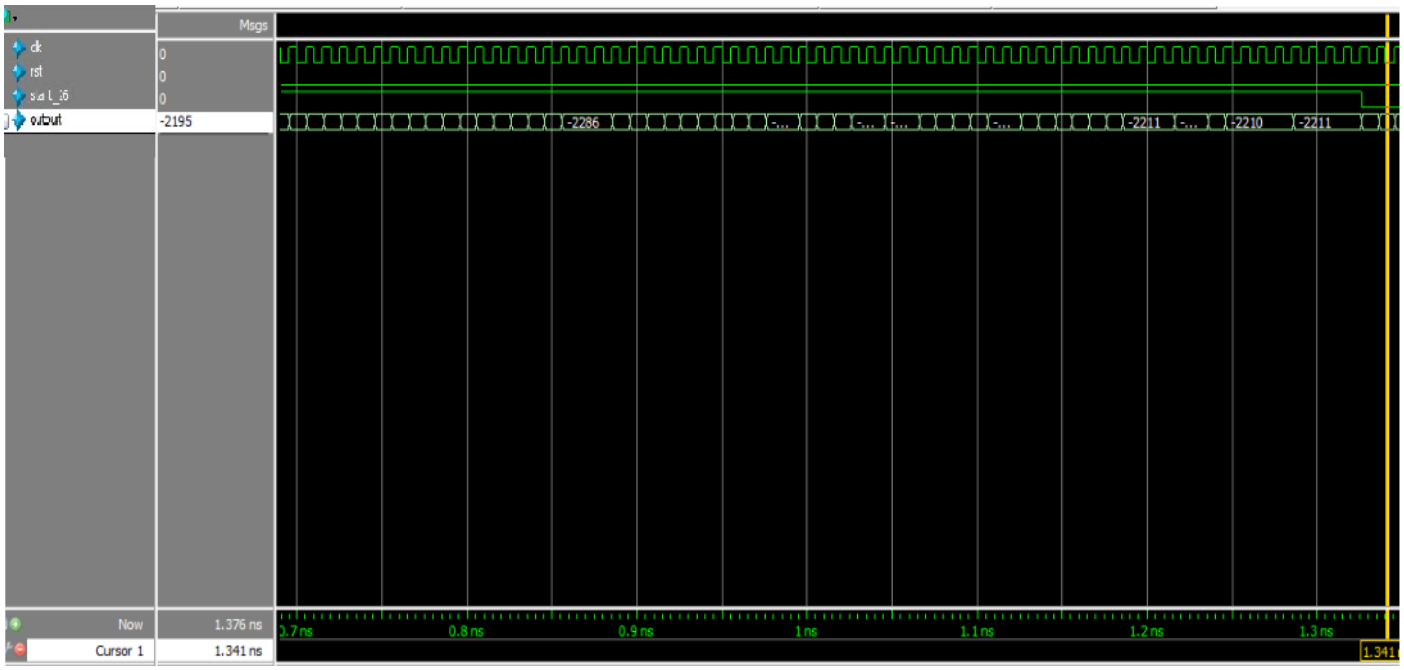

Msgs
clk
rst
start_25
output
-17
-14
-18
-7
-1
-4
-6
-8
-84
0
Now
0.466 ns
Cursor 1
0.00 ns
0.05 ns
0.1 ns
0.15 ns
0.2 ns
0.25 ns
0.3 ns
0.35 ns
0.4 ns


Msgs
start_26
output
-1390
-1484
-2195
Now
0.466 ns
Cursor 1
0.029 ns
0.05 ns
0.1 ns
0.15 ns
0.2 ns
0.25 ns
0.3 ns
0.35 ns
0.4 ns


Msgs
clk
rst
start_26
output
1
0
0
-34
-22
-48
-43
-35
-42
-39
-33
-34
-39
-5
Now
0.196 ns
Cursor 1
0.116 ns
0.02 ns
0.04 ns
0.06 ns
0.08 ns
0.1 ns
0.12 ns
0.14 ns
0.16 ns
0.18 ns
0.116 ns


Msgs
0
1
0
-2195
-1522
-1958
-1854
-1844
-1838
-1801
-1564
-2195
-2327
-132
0.196 ns
0.116 ns
0.02 ns
0.04 ns
0.06 ns
0.08 ns
0.1 ns
0.12 ns
0.14 ns
0.16 ns
0.18 ns
0.2 ns
0.116 ns


Msgs
clk
rst
start_25
output
1
0
0
0
1
4
-34
-34
Now
1.376 ns
Cursor 1
0.00 ns
0.2 ns
0.4 ns
0.6 ns
0.8 ns
1 ns
1.2 ns


Msgs
clk
rst
output
-2195
-2286
-2211
-2210
-2211
Now
1.376 ns
Cursor 1
1.341 ns
0.8 ns
0.9 ns
1 ns
1.1 ns
1.2 ns
1.3 ns