Custom Yosys Passes Result in Faulty Synthesis and Simulation Errors.

Hello,

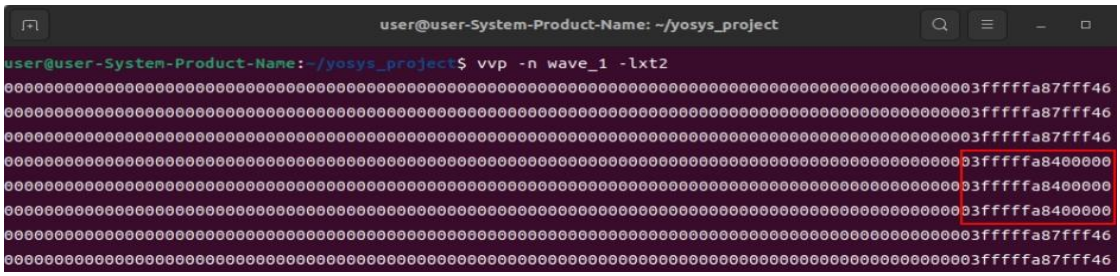
I have encountered an inconsistency issue during synthesis with Yosys.

My Icarus Verilog version is: Icarus Verilog version 13.0 (devel) (s20221226-221-g272771d18), and my Yosys version is 0.41+126.

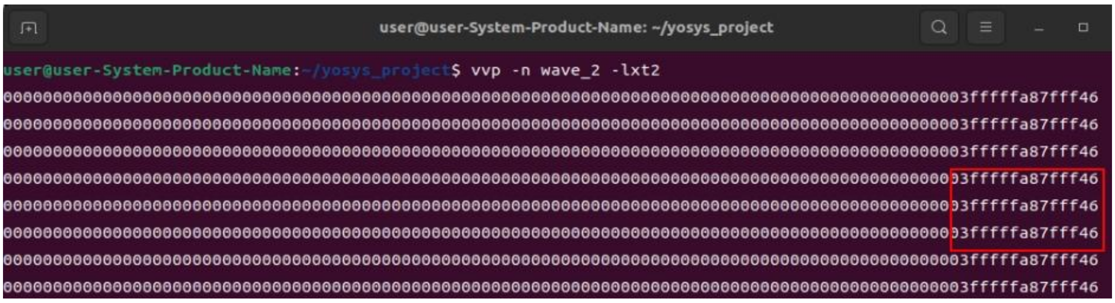
During synthesis, we did not use the default Yosys synthesis process but employed a custom pass optimization sequence. The synthesis commands for both processes are as follows:

1、Using Yosys' default synthesis process:
read_verilog rtl.v
synth
write_verilog syn_yosys.v
2、Custom optimization sequence:
read_verilog rtl.v
hierarchy; proc; opt_expr -mux_bool; opt_clean -purge; memory; opt_reduce -full; wreduce; opt_dff -nodffe; fsm; opt_expr -full; abc;
write_verilog new_syn_yosys.v

The changes in the optimization sequence should not affect the consistency of the code. However, we have observed inconsistencies in the simulation outputs when using the synthesized files generated by these two different synthesis processes with Icarus Verilog (as highlighted in the red box in the attached image).
Default synthesis process，the third and fourth line of output is:



Custom optimization sequence，the third and fourth line of output is:



This inconsistency suggests that the optimization sequence may be causing synthesis errors, leading to inconsistent simulation results.

I would appreciate your assistance in identifying the root cause of this issue.

Attached is my design file (rtl.v) and the executable script file (test.sh).

I have reduced the design file to the best of my ability as shown below：

```
module top(y, clk, wire4, wire3, wire2, wire1, wire0);
output wire [(32'h1b5):(32'h0)] y;
input wire [(1'h0):(1'h0)] clk;
input wire [(4'ha):(1'h0)] wire4;
input wire signed [(3'h5):(1'h0)] wire3;
input wire [(2'h2):(1'h0)] wire2;
input wire signed [(5'h18):(1'h0)] wire1;
input wire [(5'h14):(1'h0)] wire0;
wire signed [(5'h14):(1'h0)] wire38;
wire signed [(4'h8):(1'h0)] wire32;
wire [(5'h16):(1'h0)] wire6;
reg signed [(4'h8):(1'h0)] reg16 = (1'h0);
reg [(3'h4):(1'h0)] reg10 = (1'h0);
assign y = {wire6,wire32,wire38,(1'h0)};
assign wire6 = (-((8'ha3) || {wire2, (((8'hbc) == wire0) | (8'hb5))}));
assign wire32 = ($signed(1'b0) ? reg16[(1'h0):(1'h0)] : 8'ha1);
assign wire38 = $signed((($signed(wire0) >>> {wire6, $signed(wire32)}) ? (+reg10) : (8'ha3)));
endmodule
```

The test script is as follows：

```
yosys -p "
read_verilog rtl.v
synth
write_verilog syn_yosys.v"
iverilog -o wave_1 -y syn_yosys.v yosys_testbench.v
vvp -n wave_1 -lxt2 >> file1.txt
sed -i 's/wave_1/wave_2/g' file1.txt;
mv syn_yosys.v old_syn_yosys.v
```

```
yosys -p "
    read_verilog rtl.v
    hierarchy; proc; opt_expr -mux_bool; opt_clean -purge; memory; opt_reduce -full; wreduce; opt_dff -nodffe; fsm; opt_expr -full; abc;
    write_verilog syn_yosys.v"
iverilog -o wave_2 -y syn_yosys.v yosys_testbench.v
vvp -n wave_2 -lxt2 >> file2.txt
```