

Politecnico di Torino
Collegio di Elettronica, Telecomunicazioni e Fisica

Lab 2: digital arithmetic

Integrated Systems Architecture

Master degree in Electrical Engineering

Group: 22

Authors: Chen Xiao, Li Shancheng, Yang Haifeng, Yang Zeyu

Contents

1	FPU model	2
1.1	Simulation	2
1.2	Synthesis	4
1.2.1	Synthesis with and without compile optimization	4
1.2.2	Simulation after synthesis	4
1.3	R4-MBE multiplier	4
1.3.1	Radix 4 Modified Booth Encode	5
1.3.2	Sign extension simplify	5
1.3.3	Dadda like adder tree	6
1.3.4	Modified Booth Encode Multiplier simulation	8
1.4	Explanations, comparisons, and comments	8
2	APPENDIX A	9

1 FPU model

1.1 Simulation

Use the executable fp16_num on the server, generate 10 different half-precision floating point multiplications as the Table 1, take these results as reference to compare the results generate by the cvfpu project developed by the OpenHW group.

Verified the file data_gen16.vhd, put the Table 1 operand 1 to ctvalA , operand 2 to ctvalB and the result in the memory, as the code Listing 2 show.

Show as the code Listing1, add one input valid signal to the entity of data_gen16 (line 9), add one 16-bit input result signal (line 8). When the input valid signal is high, then compare the result from fpnew_fma.sv with the result in the memory, if they are different then assert as an error to the QuestaSim to print at the Transcript window, show as the code Listing 3 from line 17 to 20. The simulation results are saved in the text file name "results.txt" in binary format.

Run the simulation compare these 10 results, till the input valid signal went to low, the Transcript window did not give any error.

Table 1: Half-precision floating point multiplications

operand 1 decimal	operand 1 binary	operand 2 decimal	operand 2 binary	result decimal	result binary
700	(0-11000-0101111000)	0.549805	(0-01110-0001100110)	384.75	(0-10111-1000000011)
0.0116806	(0-01000-0111111011)	118.938	(0-10101-1101101111)	1.38965	(0-01111-0110001111)
-106.625	(1-10101-1010101010)	0.00171185	(0-00101-1100000011)	-0.182495	(1-01100-0111010111)
-0.721191	(1-01110-0111000101)	0.333008	(0-01101-0101010100)	-0.240112	(1-01100-1110101111)
-1.60547	(1-01111-1001101100)	0.340088	(0-01101-0101110001)	-0.545898	(1-01110-0001011110)
7.99894e-05	(0-00001-0100111110)	6552	(0-11011-1001100110)	0.523926	(0-01110-0000110001)
2228	(0-11010-0001011010)	-1.00098	(1-01111-0000000001)	-2230	(1-11010-0001011011)
28.5625	(0-10011-1100100100)	-7.05859	(1-10001-1100001111)	-201.625	(1-10110-1001001101)
60256	(0-11110-1101011011)	0.000240326	(0-00010-1111100000)	14.4844	(0-10010-1100111110)
-184.25	(1-10110-0111000010)	40	(0-10100-0100000000)	-7368	(1-11011-1100110010)

Listing 1: Entity of data generator

```

1  entity data_gen16 is
2    port (
3      CLK      : in std_logic;
4      RST_n    : in std_logic;
5      D0       : out std_logic_vector(15 downto 0);
6      D1       : out std_logic_vector(15 downto 0);
7      D2       : out std_logic_vector(15 downto 0);
8      res      : in std_logic_vector(15 downto 0);
9      valid     : in std_logic;
10     VOUT      : out std_logic;
11     END_SIM   : out std_logic;
12 end entity data_gen16;
```

Listing 2: Data from executable fp16_num

```

1  constant ctvalA : tval_t := (
2    "0110000101111000",
3    "0010000111111011",
4    "110101010101010",
5    "1011100111000101",
6    "1011111001101100",
7    "0000010100111110",
8    "0110100001011010",
9    "0100111100100100",
10   "0111101010101101",
11   "1101100111000010"
12 );
13 constant ctvalB : tval_t := (
14   "0011100001100110",
15   "0101011101101111",
16   "0001011100000011",
17   "0011010101010100",
18   "0011010101110001",
```

```

19  "0110111001100110",
20  "1011110000000001",
21  "1100011100001111",
22  "0000101111100000",
23  "0101000100000000"
24  );
25  constant result : tval_t := (
26  "0101111000000011",
27  "0011110110001111",
28  "1011000111010111",
29  "1011001110101111",
30  "101100001011110",
31  "0011100000110001",
32  "1110100001011011",
33  "1101101001001101",
34  "0100101100111110",
35  "1110111100110010"
36  );

```

Listing 3: Error check

```

1  process (CLK, RST_n) is
2    file res_fp : text open WRITE_MODE is "./results.txt";
3    variable x : integer;
4    variable line_out : line;
5    begin -- process
6      if RST_n = '0' then -- asynchronous reset (active low)
7        cnt <= 0;
8        D0 <= (others => '0');
9        D1 <= (others => '0');
10       D2 <= (others => '0');
11       VOUT <= '0';
12       sEnd_sim <= '0';
13     elsif CLK'event and CLK = '1' then -- rising clock edge
14       if (valid = '1') then
15         write(line_out, res);
16         writeline(res_fp, line_out);
17         if(result(cnt_in)/=res) then
18           assert conv_integer(unsigned(res)) = x report "Results_are_different:_index=" & integer'image(cnt_in)
19             severity error;
20         end if;
21         cnt_in<=cnt_in+1;
22       end if;

```

1.2 Synthesis

1.2.1 Synthesis with and without compile optimization

The first step we synthesis with only compile, according to the text file prj.txt gives us the hierarchy order of the files, and then organize the command order as the Listing 5 show. Pipeline, csa, ultra, pparch, MBE synthesis command order as the Appendix A Listing 6, Listing 7, Listing 8, Listing 9 and Listing 10. Before MBE synthesis, the file of fpnew_fma.sv should change port map to the multiplier we design which discuss later at the report section "Modified Booth Encode Multiplier simulation".

1.2.2 Simulation after synthesis

After synthesis, generated the netlists, the "fpnew_top_NoOptimize.v" which is the netlist respect to the synthesis command without any extra compile optimization. Run the simulation compare these 10 results, till the input valid signal went to low, the Transcript window did not give any error, the behaviour is the same as the system verilog files before synthesis.

And then change the command at line 13 of Listing 4 respect to other netlists of compile options, and the line 23 of Listing 4 respect to the sdf file of the compile options, put these command into the Transcript window of Questasim. Then run the simulation, the results did not give any error neither, as well as the netlist generated respect to Modified Booth Encode multiplier, show as Figure 1. Other simulations results are the same as Figure 1.

Listing 4: Verify the netlist behaviour via simulation

```

1  source /eda/scripts/init_questa_core_prime
2  rmdir work
3  mkdir work
4  vlib work
5  vlog -reportprogress 300 -work work ../src/cf_math_pkg.sv
6  vlog -reportprogress 300 -work work ../src/lzc.sv
7  vlog -reportprogress 300 -work work ../src/rr_arb_tree.sv
8  vlog -reportprogress 300 -work work ../src/fpnew_pkg.sv
9  vlog -reportprogress 300 -work work ../src/fpnew_classifier.sv
10 vlog -reportprogress 300 -work work ../src/fpnew_rounding.sv
11 vlog -reportprogress 300 -work work ../src/fpnew_opgroup_fmt_slice.sv
12 vlog -reportprogress 300 -work work ../src/fpnew_opgroup_block.sv
13 vlog -reportprogress 300 -work work ../netlist/fpnew_top_NoOptimize.v
14 #vlog -reportprogress 300 -work work ../netlist/fpnew_top_ultra.v
15 #vlog -reportprogress 300 -work work ../netlist/fpnew_top_pcsa.v
16 #vlog -reportprogress 300 -work work ../netlist/fpnew_topPipeline.v
17 #vlog -reportprogress 300 -work work ../netlist/fpnew_topPparch.v
18 #vlog -reportprogress 300 -work work ../netlist/fpnew_topMBE.v
19 vcom -reportprogress 300 -work work ../tb/clk_gen.vhd
20 vcom -reportprogress 300 -work work ../tb/data_gen16.vhd
21 vlog -reportprogress 300 -work work ../tb/tb_fpnew_top_rtl.sv
22 vlog -reportprogress 300 -work work ../tb/tb_fpnew_top_net.sv
23 vsim -L /eda/dk/nangate45/verilog/qsim2020.4 -sdftyp /tb_fpnew_top/UUT=../netlist/fpnew_top_NoOptimize.sdf
    work.tb_fpnew_top
24 add wave *
```

1.3 R4-MBE multiplier

To compare the multiplier that we described at the previous section, according to the half-precision format we need to handle the 10 mantissa bits, with the hidden bit at the most significant position, then we should design the multiplier is 11 bit.

The implementation of this multiplier is based on three parts, Radix 4 Modified Booth Encode, Dadda like adder tree and sign extension simplifying.

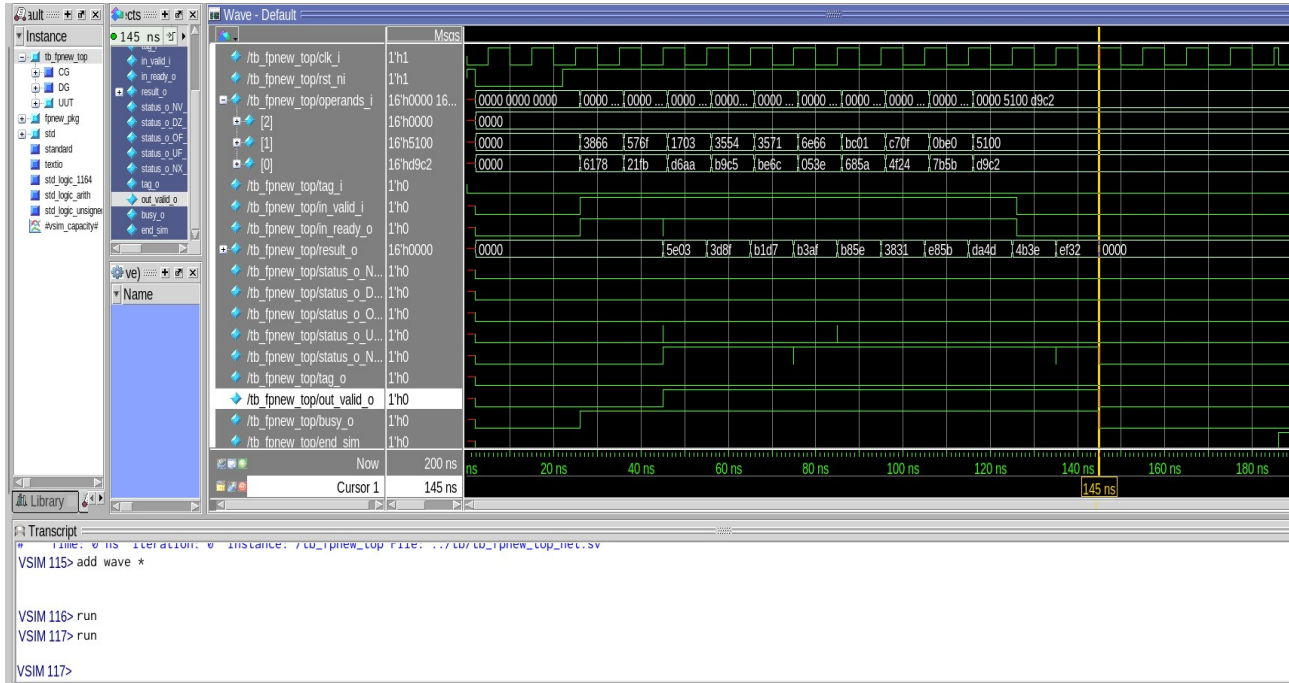


Figure 1: Simulation after synthesis

Table 2: Modified Booth Encode consider sign extension and Dadda tree

Multiplier Bits	Selection
000	0
001	multiplicand
010	multiplicand
011	shift left 1 bit of multiplicand
100	shift left 1 bit of multiplicand and then bitwise complement
101	bitwise complement multiplicand
110	bitwise complement multiplicand
111	bitwise complement 0

按照不考虑sign extension的方法, Table 2的下半负数还要加1, 而这里的加一放到dadda like tree

1.3.1 Radix 4 Modified Booth Encode

Generate 7 3-bit number from second operand, as the select signal of the multiplexer, then multiplicand take these 7 select signal to produce 7 partial products. These partial products are generated by multiplicand, according to the 7 select signal select the entry in the Table 2, without adding 1 because this adding 1 is moved to the Dadda like tree presents as half adder or full adder.

1.3.2 Sign extension simplify

Booth encode generate the partial products in some case is negative, so we need to extend the sign bit of each partial products. Traditionally, in 2's complement format, turn a binary number from positive to negative should complement each bit and then add a 1 in the LSB, the worst case is that all the partial products are negative, then we should complement the bit from the LSB(first bit) to the MSB(21th bit), and add all partial products become more area consuming. But there is one solution to simplify.

According to the file "G.W. Bewick. Fast multiplication: algorithms and implementation - Appendix A - Sign Extension in Booth Multiplier. PhD thesis, Stanford, 1994.", each partial products are simplified as the Figure 2, where $s = 0$ if partial product is positive (top 4 entries from Table 3), $s = 1$ if partial product is negative (bottom 4 entries from table Table 3). In this way, the top left area of adder plane is neglected. In order to simplify the sign extension, so the partial products are

Table 3: Modified Booth Encode

Multiplier Bits	Selection
000	+ 0
001	+ Multiplicand
010	+ Multiplicand
011	+ 2 x Multiplicand
100	-2 x Multiplicand
101	- Multiplicand
110	- Multiplicand
111	- 0

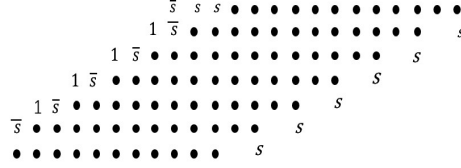


Figure 2: Sign extension simplify

extended to 12 bit except the bottom one is 11 bit.

It is worthy to point out that 0 complement is not 0, so the last entry of the table there is minus 0.

1.3.3 Dadda like adder tree

The full adders allows for a compression ratio of at most 3/2, from 3 input(operand 1, operand 2, carry in), to two output(carry out, sum). There are 7 partial products, so we need 4 level to compress.

- Level 0 compress to 6,
- Level 1 compress to 4,
- Level 2 compress to 3,
- Level 3 compress to 2.

In the end, use the carry propagate adder to add the two products from the level 3.

Dadda like tree allocate half adders and full adders as Figure 4, where q0 is the sum of s0 and p1_10, p1_10 is the 10th bit of the original partial product 0, s0 is generated by the Table 3 according to the original partial product 0 (described at previous section), and so on so forth, which respect dot notation as the Figure 5.

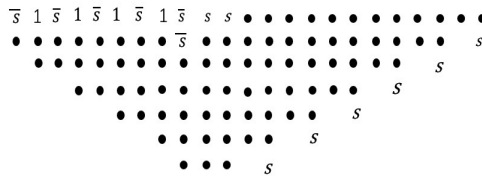


Figure 3: Align data

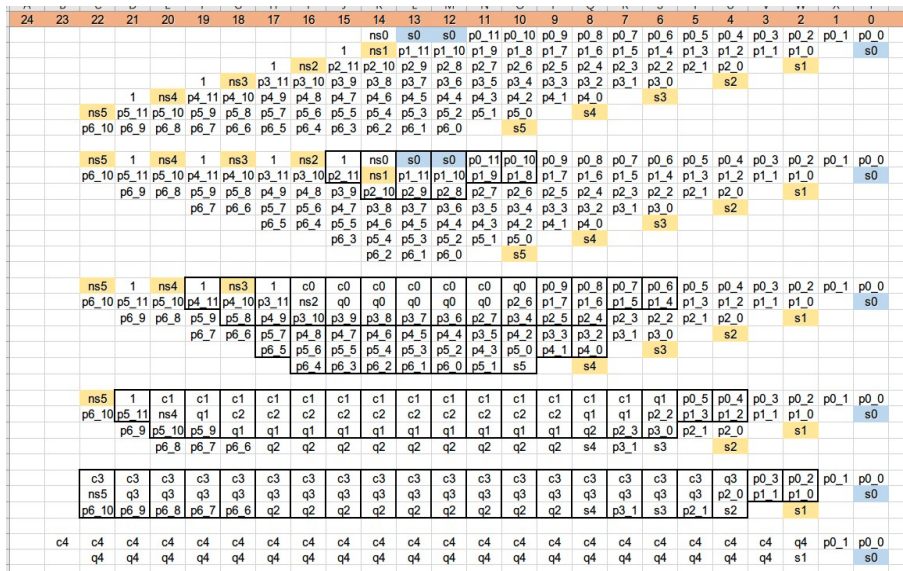


Figure 4: Dadda like tree allocate adders

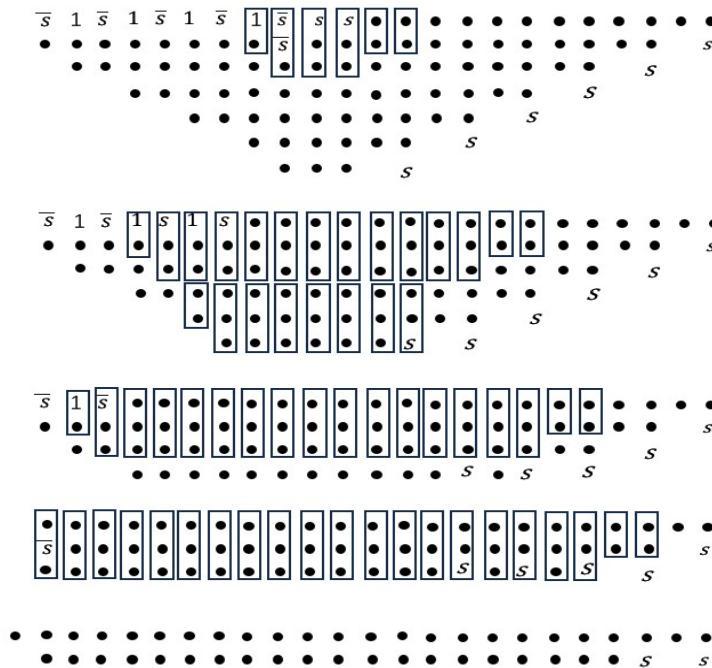


Figure 5: Dadda like tree allocate adders dot notation

D	E	F	G	H	I	J	K	L	M	N	O
2047	2040	2033	2026	2019	2012	2005	1998	1991	1984	1977	1970
2046	2041	2036	2031	2026	2021	2016	2011	2006	2001	1996	1991
4188162	4163640	4139188	4114806	4090494	4066252	4042080	4017978	3993946	3969984	3946092	3922270




 /tb_csa/DIN...	0	0	2047	2040	2033	2026	2019	2012	2005	1998	1991	1984	1977	1970
 /tb_csa/DIN...	0	0	2046	2041	2036	2031	2026	2021	2016	2011	2006	2001	1996	1991
 /tb_csa/DOUT...	0	0	4188162	4163640	4139188	4114806	4090494	4066252	4042080	4017978	3993946	3969984	3946092	3922270

Figure 6: Stand-alone component of 11 bit multiplier simulation

Table 4: Comparison of synthesis

compile option	$T_{min}(ns)$	$F_{max}(MHz)$	area(μm^2)
pparch	2.4	416.7	3928.3
pipeline	2.5	400	4055.2
csa	2.5	400	3951.2
ultra	3.0	333.3	3737.6
MBE	3.0	333.3	3696.1
only compile	3.5	285.7	4343.8

1.3.4 Modified Booth Encode Multiplier simulation

First we implemented a simple test file for the Stand alone component of 11 bit multiplier, as an unsigned 11 bit multiplier, read the text file as operands input, and do the multiplication, output the results to the waveform.

Simulation of the multiplier show as the waveform, the products are generated every clock cycle, as the Figure 6 , the upper part of the picture is table come from excel, the results are calculated by formula function, the lower part of the picture is waveform from QuestaSim. Every products are identical.

Then we change the source code file of fpnew_fma.sv, comment the line of code:

"assign product = mantissa_a * mantissa_b;"

Replace with this following new code line:

"mbe_mantissa_mul uumbe(.a(mantissa_a),.b(mantissa_b),.out(product));"

In this way to port map our MBE 11 bit multiplier instead of the symbol star "*".

The test bench files are the same as the previous report section stated, take that 10 different half-precision floating point multiplications (Table 1) again to check the result of multiplier.

Run the simulation till the input valid signal went to low, the Transcript window did not give any error.

should not be increase the throughput, but increase the max frequency. Lab1 said 3 streams input is high throughput that is unfolded with pipeline, unfolded without pipeline is not high max frequency.

1.4 Explanations, comparisons, and comments

As show in the Table 4, in terms of max frequency the best is pparch with optimize_registers, optimization with pipeline the Latches are inserted after every FA levels such that improve the throughput; the worst is the one with only compile. Modified Booth Encode, which is design by ourselves, probably due to the last stage carry propagate adder, the critical path includes 23 adders, so the delay is long.

In terms of area, the best is MBE, the worst is the one with only compile, MBE is optimized with ultra. In the way we implement the MBE, actually the structure is more or less similar to the synthesis solution with carry save adder, probably due to the sign extension simplify, then it save some area.

Compare the area report of csa and only compile, csa's number of references and sequential cells is larger than only compile, the other aspects of csa are all smaller than no optimization. Probably because the following reason, synthesis of no optimization the first partial product is composed of FAs

对比CSA和只有compile
的情况参考part2_A 106
107页

except the LSB and MSB, while synthesis of csa optimization the first partial product are all HAs. Further more, from the second to the last partial products the MSB are FAs, while optimizing with csa, from the second to the last partial products the MSB are HAs. Such that optimizing with csa save more area compare to the one without optimization.

The "ultra" optimize solution whose period is behind pipeline, according to the user guide of Design Compiler, who provides automatic leakage power optimization, maybe due to the consideration of this leakage power optimization, the throughput and area performance is not the best.

Due to the experimental requirement of using the Dadda tree structure, which is ALAP (As Late As Possible), it utilizes fewer resources compared to the Wallace tree. However, the total delay correspondingly increases

2 APPENDIX A

Listing 5: Synthesis with only compile

```

1  rmdir work
2  mkdir work
3  source /eda/scripts/init_design_vision
4  set define_design_lib WORK -path ./work
5  set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6  set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7  set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8  set synthetic_library [list "dw_foundation.sldb" ]
9  dc_shell-xg-t
10 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
11 analyze -f sverilog -lib WORK ../src/lzc.sv
12 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv
13 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
14 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
15 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
16 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
17 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
20 set power_preserve_rtl_hier_names true
21 elaborate fpnew_top -lib WORK
22 create_clock -name MY_CLK -period 3.5 clk_i
23 set_dont_touch_network MY_CLK
24 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
25 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
26 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
27 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
28 set_load $OLOAD [all_outputs]
29 compile
30 report_timing > NoOptimize3p5ns.txt
31 report_area > NoOptimize3p5nsArea.txt
32 report_resources > NoOptimize3p5Resources.txt
33 change_names -hierarchy -rules verilog
34 write_sdf ../netlist/fpnew_top_NoOptimize.sdf
35 write -f verilog -hierarchy -output ../netlist/fpnew_top_NoOptimize.v
36 write_sdc ../netlist/fpnew_top_NoOptimize.sdc

```

Listing 6: Synthesis with pipeline

```

1  rmdir work
2  mkdir work
3  source /eda/scripts/init_design_vision
4  set define_design_lib WORK -path ./work
5  set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6  set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7  set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8  set synthetic_library [list "dw_foundation.sldb" ]
9  dc_shell-xg-t
10 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
11 analyze -f sverilog -lib WORK ../src/lzc.sv
12 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv

```

```

13 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
14 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
15 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
16 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
17 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
20 set power_preserve_rtl_hier_names true
21 elaborate fpnew_top -lib WORK
22 create_clock -name MY_CLK -period 2.5 clk_i
23 set_dont_touch_network MY_CLK
24 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
25 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
26 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
27 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
28 set_load $OLOAD [all_outputs]
29 compile
30 optimize_registers
31 report_timing > pipeline2p5ns.txt
32 report_area > pipeline2p5nsArea.txt
33 report_resources > pipe2p5nsResources.txt
34 change_names -hierarchy -rules verilog
35 write_sdf ../netlist/fpnew_topPipeline.sdf
36 write -f verilog -hierarchy -output ../netlist/fpnew_topPipeline.v
37 write_sdc ../netlist/fpnew_toppipeline.sdc

```

Listing 7: Synthesis with csa

```

1  rmdir work
2  mkdir work
3  source /eda/scripts/init_design_vision
4  set define_design_lib WORK -path ./work
5  set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6  set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7  set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8  set synthetic_library [list "dw_foundation.sldb" ]
9  dc_shell-xgt-t
10 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
11 analyze -f sverilog -lib WORK ../src/lzc.sv
12 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv
13 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
14 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
15 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
16 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
17 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
20 set power_preserve_rtl_hier_names true
21 elaborate fpnew_top -lib WORK
22 create_clock -name MY_CLK -period 2.5 clk_i
23 set_dont_touch_network MY_CLK
24 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
25 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
26 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
27 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
28 set_load $OLOAD [all_outputs]
29 ungroup -all -flatten
30 set_implementation DW02_mult/csa [find cell *mult*]
31 compile
32 optimize_registers
33 report_timing > csa2d5ns.txt
34 report_area > csa2d5nsArea.txt
35 report_resources > csa2d5nsResources.txt
36 change_names -hierarchy -rules verilog
37 write_sdf ../netlist/fpnew_topcsa.sdf
38 write -f verilog -hierarchy -output ../netlist/fpnew_topcsa.v
39 write_sdc ../netlist/fpnew_topcsa.sdc

```

Listing 8: Synthesis with ultra

```

1  rmdir work
2  mkdir work
3  source /eda/scripts/init_design_vision

```

```

4 set define_design_lib WORK -path ./work
5 set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6 set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7 set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8 set synthetic_library [list "dw_foundation.sldb" ]
9 dc_shell-xg-t
10 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
11 analyze -f sverilog -lib WORK ../src/lzc.sv
12 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv
13 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
14 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
15 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
16 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
17 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
20 set power_preserve_rtl_hier_names true
21 elaborate fpnew_top -lib WORK
22 create_clock -name MY_CLK -period 3.0 clk_i
23 set_dont_touch_network MY_CLK
24 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
25 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
26 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
27 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
28 set_load $OLOAD [all_outputs]
29 compile_ultra
30 report_timing > ultra3p0ns.txt
31 report_area > ultra1p3p0Area.txt
32 report_resources > ultra3p0Resources.txt
33 change_names -hierarchy -rules verilog
34 write_sdf ../netlist/fpnew_top_ultra.sdf
35 write -f verilog -hierarchy -output ../netlist/fpnew_top_ultra.v
36 write_sdc ../netlist/fpnew_top_ultra.sdc

```

Listing 9: Synthesis with pparch

```

1 rmdir work
2 mkdir work
3 source /eda/scripts/init_design_vision
4 set define_design_lib WORK -path ./work
5 set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6 set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7 set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8 set synthetic_library [list "dw_foundation.sldb" ]
9 dc_shell-xg-t
10 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
11 analyze -f sverilog -lib WORK ../src/lzc.sv
12 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv
13 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
14 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
15 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
16 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
17 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
20 set power_preserve_rtl_hier_names true
21 elaborate fpnew_top -lib WORK
22 create_clock -name MY_CLK -period 2.4 clk_i
23 set_dont_touch_network MY_CLK
24 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
25 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
26 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
27 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
28 set_load $OLOAD [all_outputs]
29 ungroup -all -flatten
30 set_implementation DW02_mult/pparch [find cell *mult*]
31 compile
32 optimize_registers
33 report_timing > pparch2d4ns.txt
34 report_area > pparch2d4nsArea.txt
35 report_resources > pparch2d4nsResources.txt
36 change_names -hierarchy -rules verilog
37 write_sdf ../netlist/fpnew_topPparch.sdf
38 write -f verilog -hierarchy -output ../netlist/fpnew_topPparch.v

```

```
39 write_sdc ../netlist/fpnew_toppparch.sdc
```

Listing 10: Modified Booth Encode Synthesis with ultra

```
1  rmdir work
2  mkdir work
3  source /eda/scripts/init_design_vision
4  set define_design_lib WORK -path ./work
5  set search_path [list . /eda/synopsys/2021-22/RHELx86/SYN 2021.06-SP4/libraries/syn/eda/dk/nangate45/synopsys ]
6  set link_library [list "*" "NangateOpenCellLibrary_typical_ecsm.db" "dw_foundation.sldb" ]
7  set target_library [list "NangateOpenCellLibrary_typical_ecsm.db" ]
8  set synthetic_library [list "dw_foundation.sldb" ]
9  dc_shell-xg-t
10 analyze -f sverilog -lib WORK ../src/CSA.sv
11 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
12 analyze -f sverilog -lib WORK ../src/mux.sv
13 analyze -f sverilog -lib WORK ../src/dadda.sv
14 analyze -f sverilog -lib WORK ../src/boothmul.sv
15 analyze -f sverilog -lib WORK ../src/cf_math_pkg.sv
16 analyze -f sverilog -lib WORK ../src/lzc.sv
17 analyze -f sverilog -lib WORK ../src/rr_arb_tree.sv
18 analyze -f sverilog -lib WORK ../src/fpnew_pkg.sv
19 analyze -f sverilog -lib WORK ../src/fpnew_classifier.sv
20 analyze -f sverilog -lib WORK ../src/fpnew_rounding.sv
21 analyze -f sverilog -lib WORK ../src/fpnew_fma.sv
22 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_fmt_slice.sv
23 analyze -f sverilog -lib WORK ../src/fpnew_opgroup_block.sv
24 analyze -f sverilog -lib WORK ../src/fpnew_top.sv
25 set power_preserve_rtl_hier_names true
26 elaborate fpnew_top -lib WORK
27 create_clock -name MY_CLK -period 3.0 clk_i
28 set_dont_touch_network MY_CLK
29 set_clock_uncertainty 0.07 [get_clocks MY_CLK]
30 set_input_delay 0.5 -max -clock MY_CLK [remove_from_collection [all_inputs] clk_i]
31 set_output_delay 0.5 -max -clock MY_CLK [all_outputs]
32 set OLOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
33 set_load $OLOAD [all_outputs]
34 compile_ultra
35 report_timing > MBE3ns.txt
36 report_area > MBE3nsArea.txt
37 change_names -hierarchy -rules verilog
38 write_sdf ../netlist/fpnew_topMBE.sdf
39 write -f verilog -hierarchy -output ../netlist/fpnew_topMBE.v
40 write_sdc ../netlist/fpnew_topMBE.sdc
```