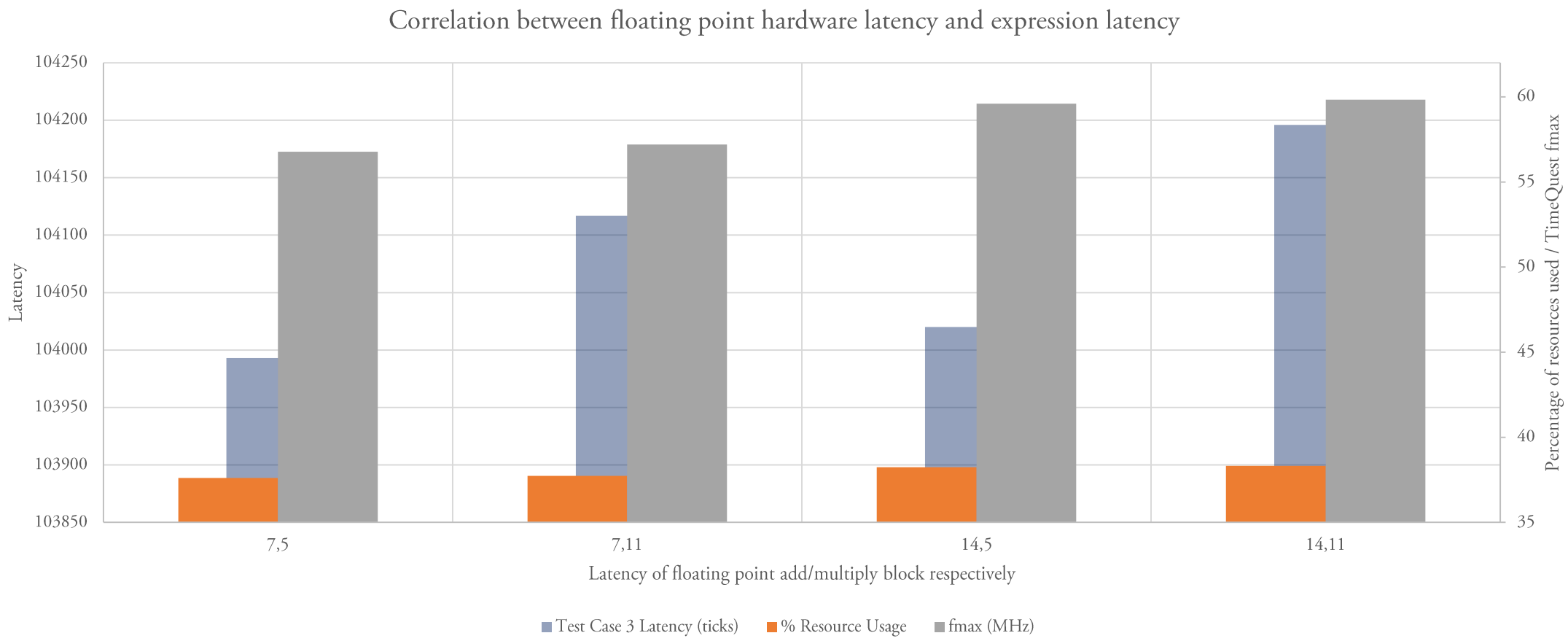
Group 18

Jeremy Chan*
Department of Electrical and Electronic Engineering
Imperial College
London, United Kingdom
jc4913@ic.ac.uk | 00818433

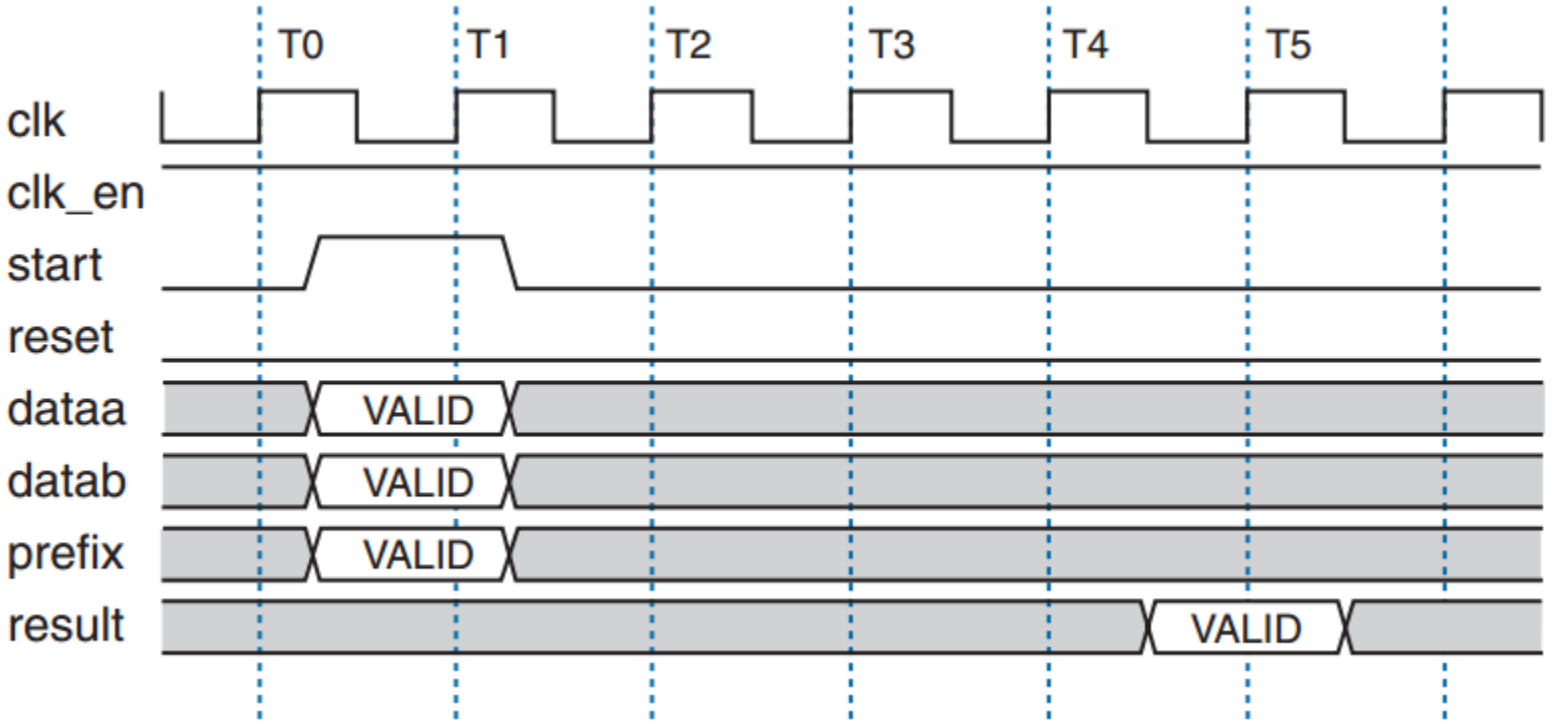
Chak Yeung Dominic Kwok*
Department of Electrical and Electronic Engineering
Imperial College
London, United Kingdom
cyk113@ic.ac.uk | 00827832

*These authors contributed equally to this work

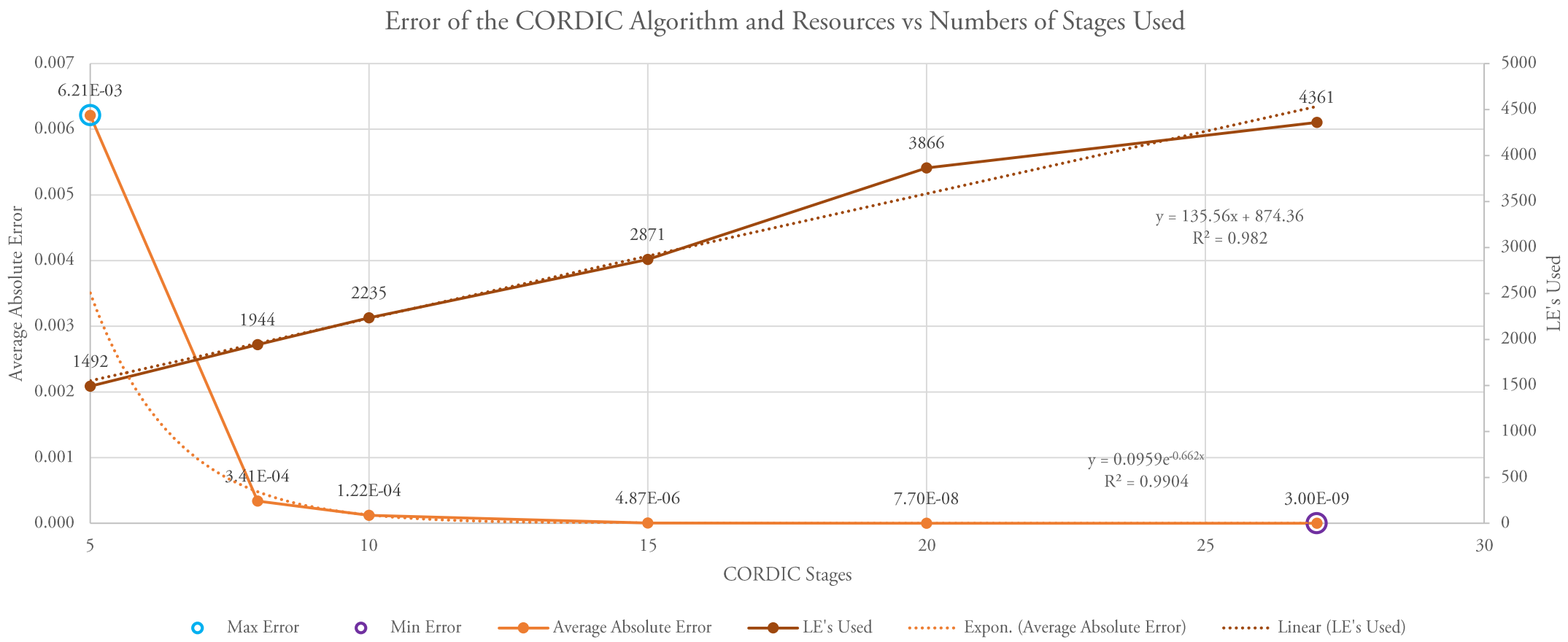
Pipeline Stages



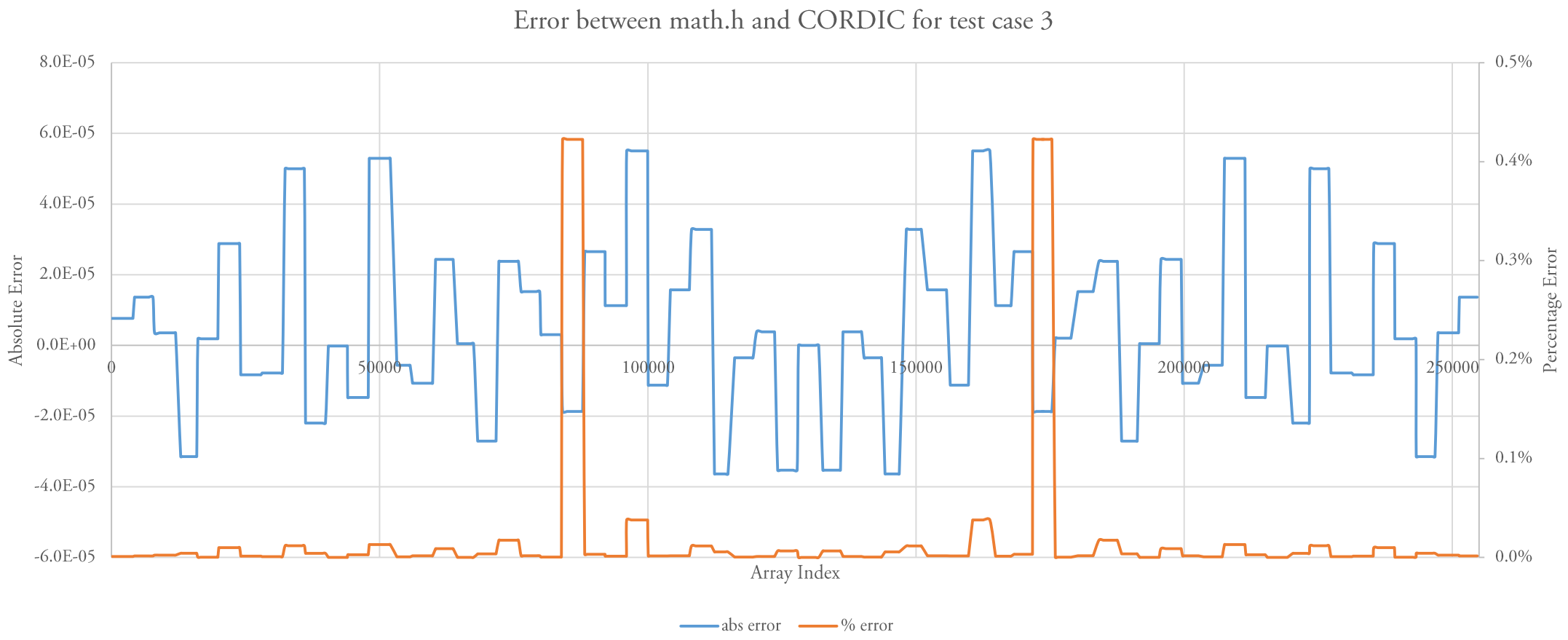
NIOS II Handshaking



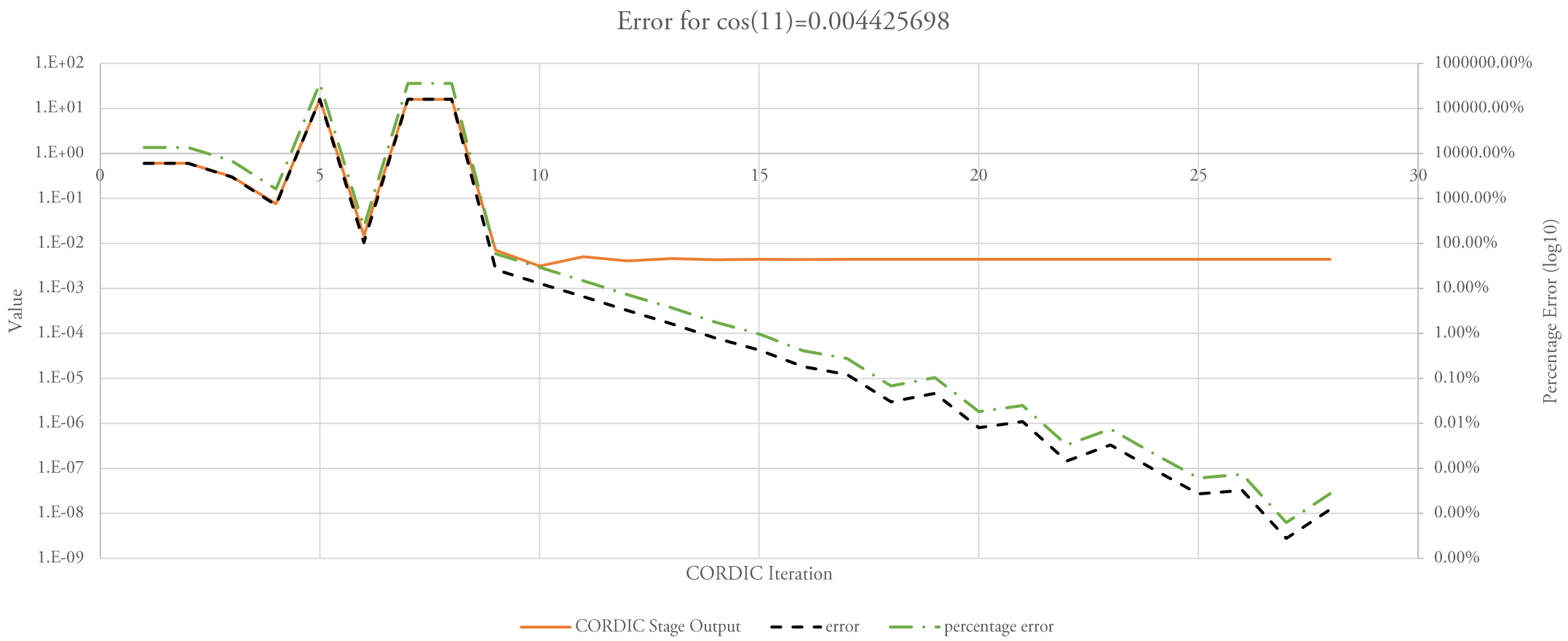
CORDIC Stage's Justification



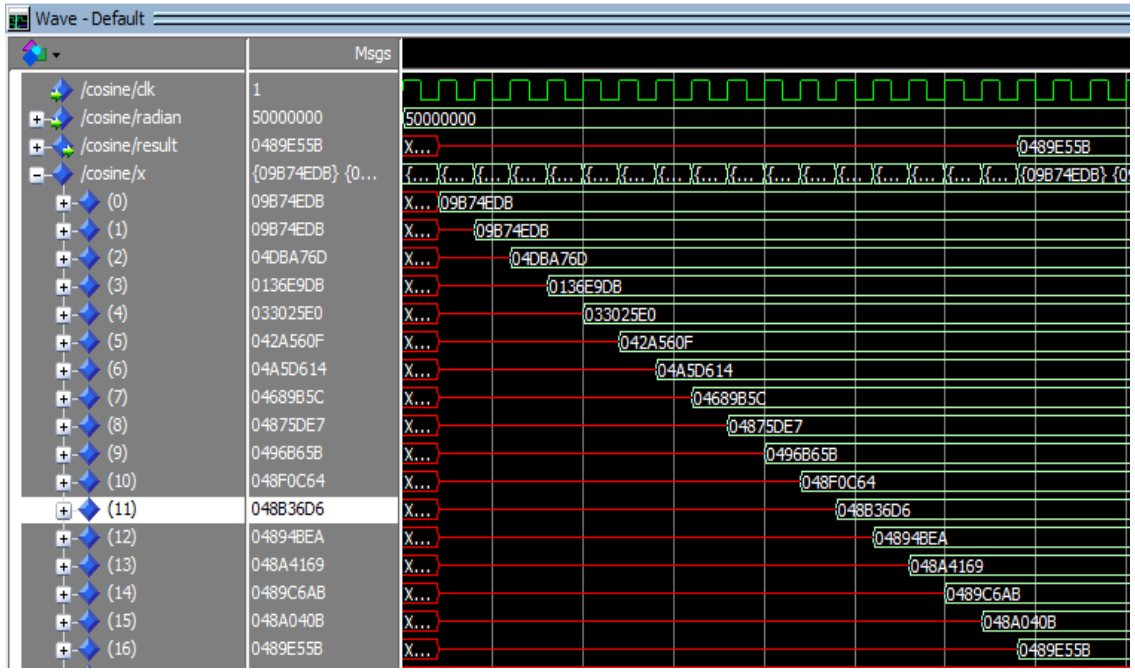
CORDIC Accuracy

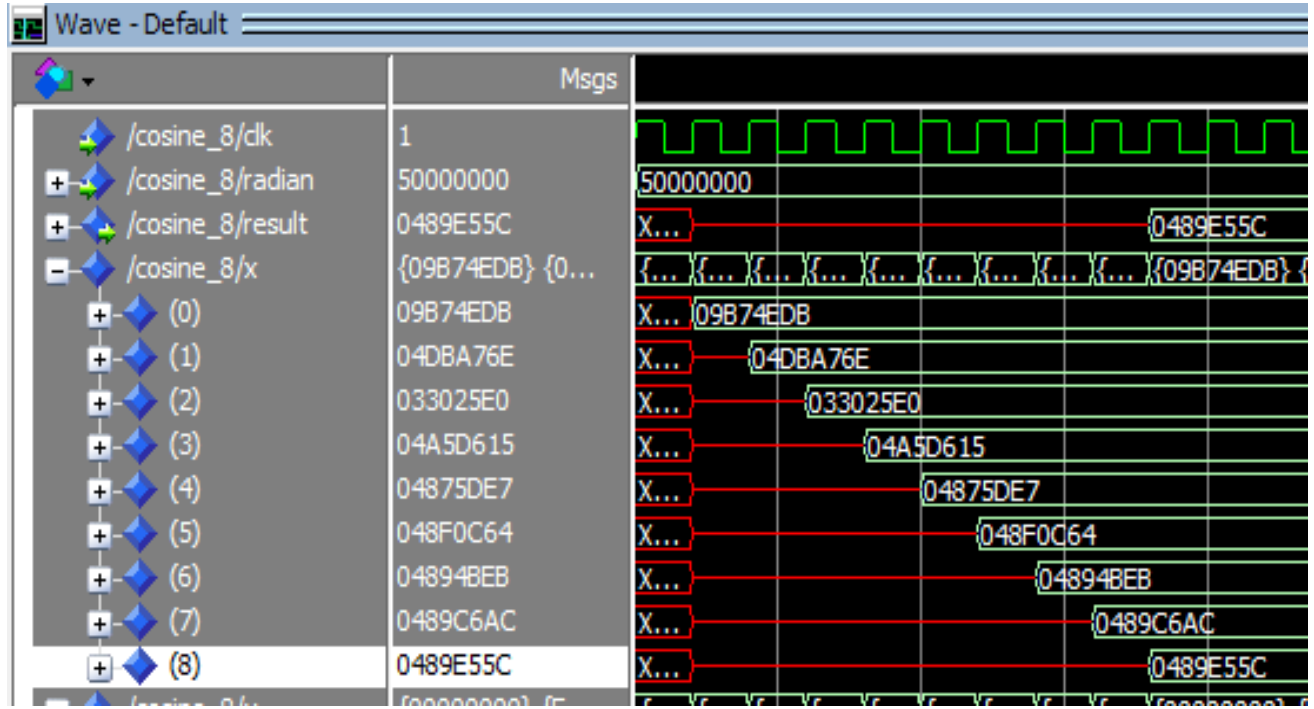


cos(11) Corner Case

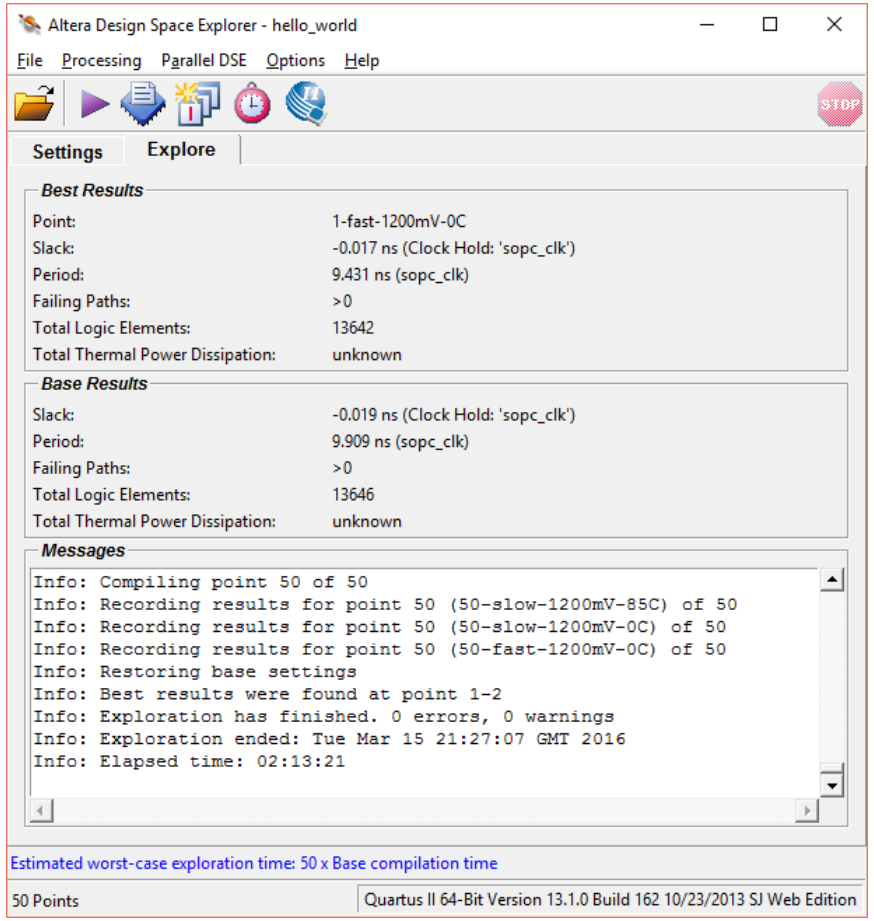


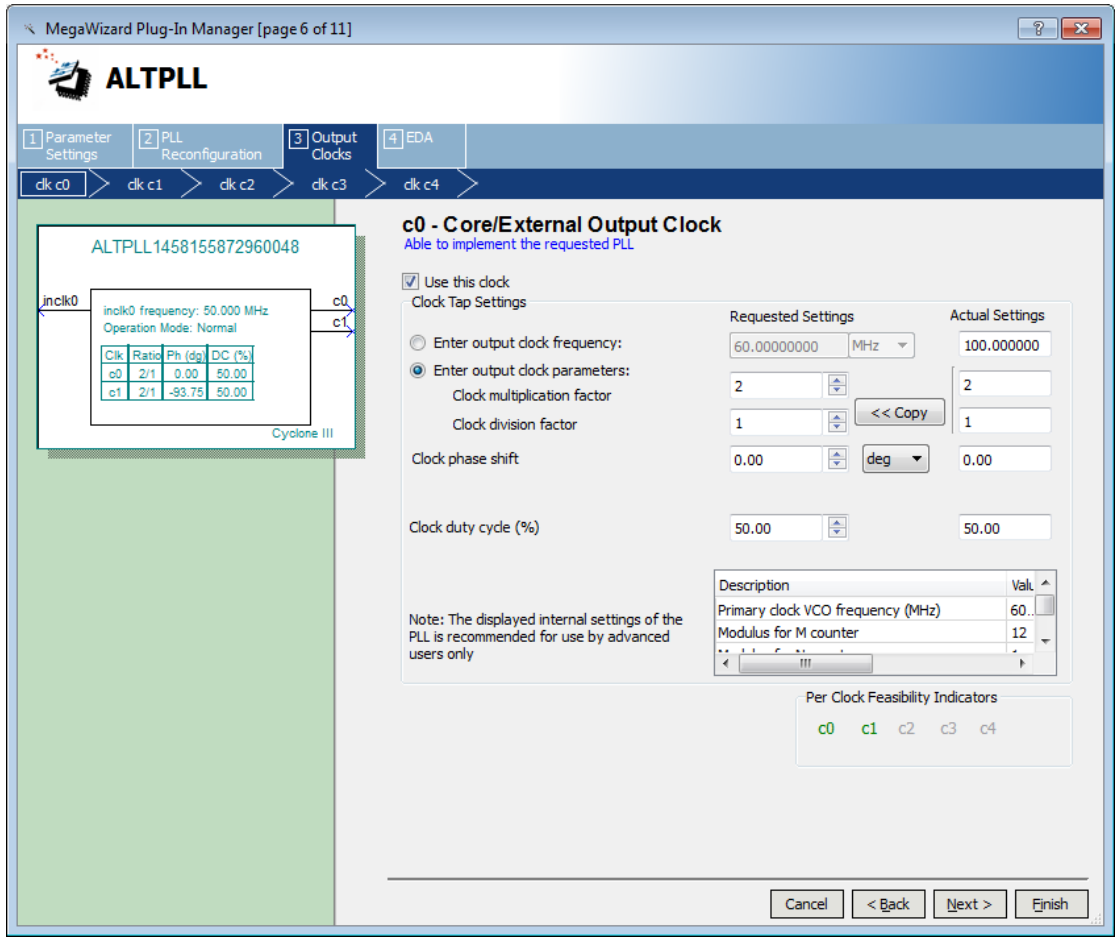
Unrolling the CORDIC





Seed Sweep


Altera Design Space Explorer - hello_world
File Processing Parallel DSE Options Help
Settings Explore
Best Results
Point: 1-fast-1200mV-0C
Slack: -0.017 ns (Clock Hold: 'sopc_clk')
Period: 9.431 ns (sopc_clk)
Failing Paths: >0
Total Logic Elements: 13642
Total Thermal Power Dissipation: unknown
Base Results
Slack: -0.019 ns (Clock Hold: 'sopc_clk')
Period: 9.909 ns (sopc_clk)
Failing Paths: >0
Total Logic Elements: 13646
Total Thermal Power Dissipation: unknown
Messages
Info: Compiling point 50 of 50
Info: Recording results for point 50 (50-slow-1200mV-85C) of 50
Info: Recording results for point 50 (50-slow-1200mV-0C) of 50
Info: Recording results for point 50 (50-fast-1200mV-0C) of 50
Info: Restoring base settings
Info: Best results were found at point 1-2
Info: Exploration has finished. 0 errors, 0 warnings
Info: Exploration ended: Tue Mar 15 21:27:07 GMT 2016
Info: Elapsed time: 02:13:21
Estimated worst-case exploration time: 50 x Base compilation time
50 Points
Quartus II 64-Bit Version 13.1.0 Build 162 10/23/2013 SJ Web Edition

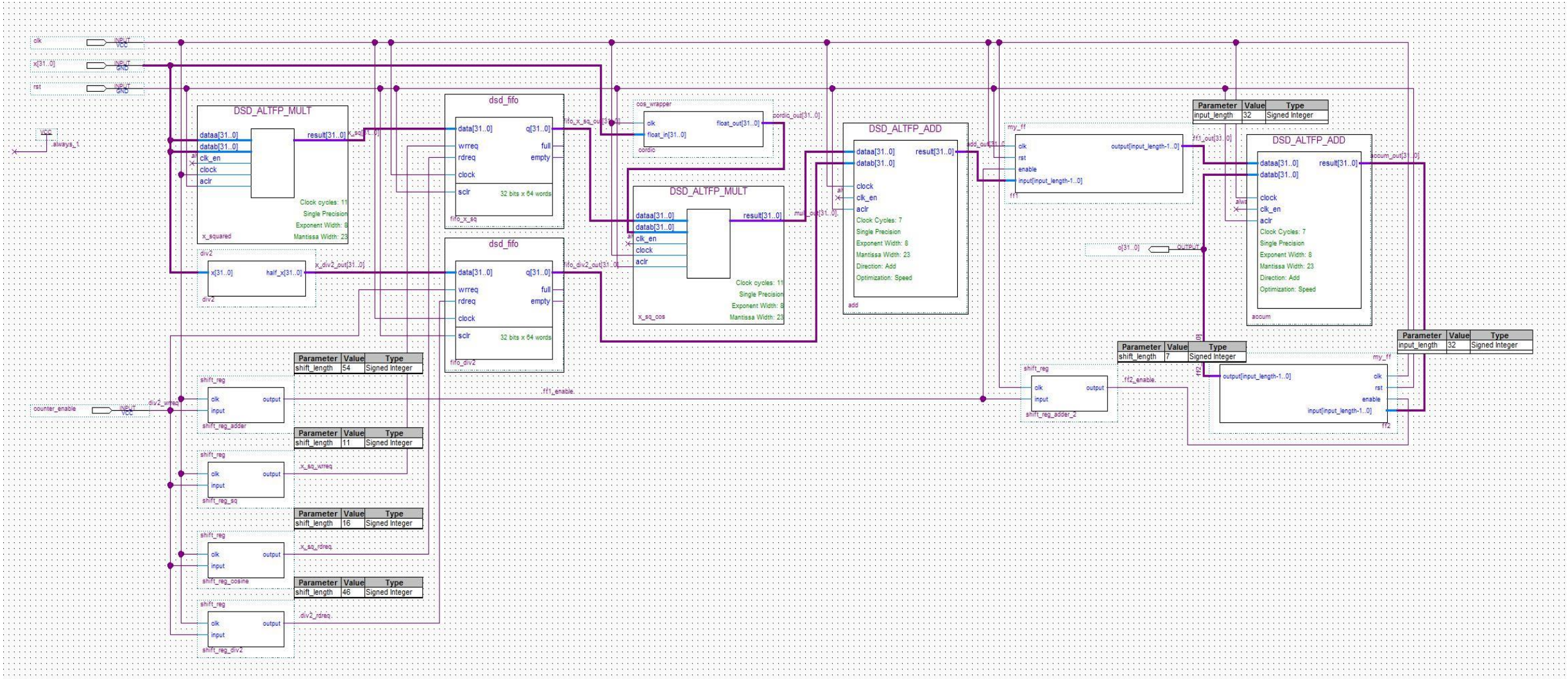

MegaWizard Plug-In Manager [page 6 of 11]
ALTPLL
1 Parameter Settings
2 PLL Reconfiguration
3 Output Clocks
4 EDA
clk c0
clk c1
clk c2
clk c3
clk c4
ALTPLL1458155872960048
inclk0
c0
c1
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk Ratio Ph (dg) DC (%)
c0 2/1 0.00 50.00
c1 2/1 -93.75 50.00
Cyclone III
c0 - Core/External Output Clock
Able to implement the requested PLL
Use this clock
Clock Tap Settings
Requested Settings
Actual Settings
Enter output clock frequency:
60.00000000 MHz
100.000000
Enter output clock parameters:
Clock multiplication factor
2
2
Clock division factor
1
1
<< Copy
Clock phase shift
0.00
deg
0.00
Clock duty cycle (%)
50.00
50.00
Note: The displayed internal settings of the PLL is recommended for use by advanced users only
Description
Primary clock VCO frequency (MHz)
Modulus for M counter
12
Per Clock Feasibility Indicators
c0 c1 c2 c3 c4
Cancel
< Back
Next >
Finish

Software Optimization Level

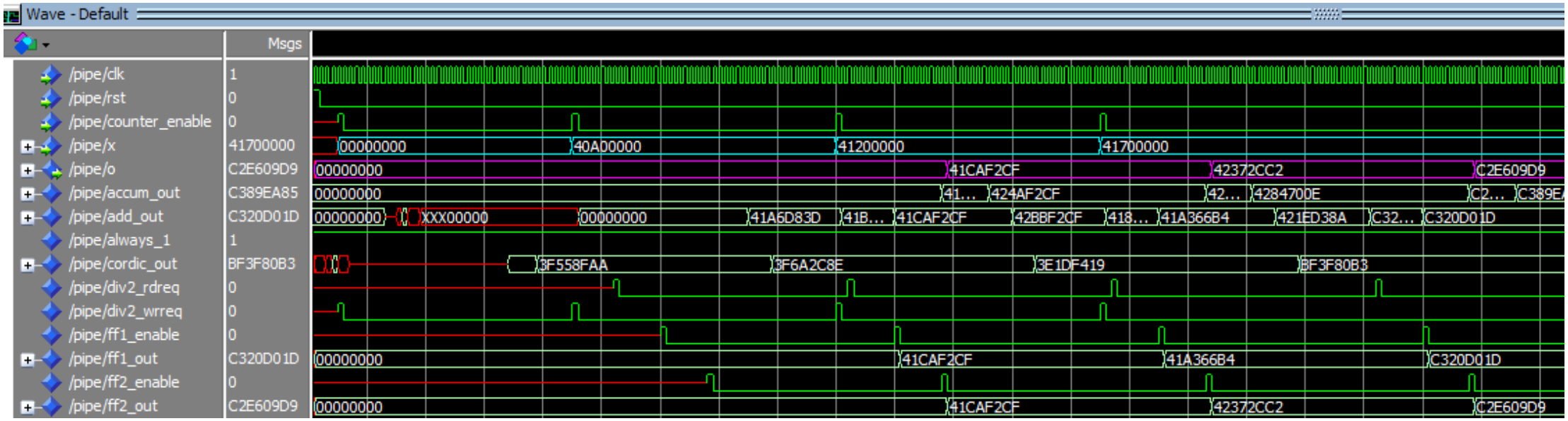
```
/cygdrive/h/DSD/T8_combined_func2/software/t8comb2
Pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 73KB in 1.0s (73.0KB/s)
Verified OK
Starting processor at address 0x008001B4
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Hi, v1.7
Running algorithm for task: 4+
Dividing result by 1024 is: OFF
gen vector 1, sum vector 1
Time = 0; Result = 57881.226562
gen vector 2, sum vector 2
Time = 1; Result = -76943.906250
gen vector 3, sum vector 3
Time = 154; Result = 37025504.000000


nios2-terminal: exiting due to ^C on host

jc4913@eews506a-022 /cygdrive/h/DSD/T8_combined_func2/software/t8comb2
$
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity combined_func2 is
port(
clk    : in  std_logic;
x      : in  std_logic_vector(31 downto 0);
last_result: std_logic_vector(31 downto 0);
result : out std_logic_vector(31 downto 0);
clk_en : in std_logic
);
end entity combined_func2;

architecture main of combined_func2 is
signal exponent : std_logic_vector(7 downto 0);
signal half_x   : std_logic_vector(31 downto 0);
signal cos_out  : std_logic_vector(31 downto 0);
signal sq_out   : std_logic_vector(31 downto 0);
signal mult_out : std_logic_vector(31 downto 0);
signal add_out : std_logic_vector(31 downto 0);
begin
sub_exponent : process(x)
begin
exponent <= std_logic_vector(signed(x(30 downto 23)) - 1);
end process sub_exponent;

div2 : process(exponent, x)
begin
if to_integer(signed(x)) = 0 then
half_x <= (others => '0');
else
half_x <= x(31) & exponent & x(22 downto 0);
end if;
end process div2;

cosine_blk : entity cos_wrapper_8 port map(
clk,
x,
cos_out,
clk_en
);

sq_block : ENTITY fpmul PORT map(
clk_en,
clk,
x,
x,
sq_out
);

mult_block : ENTITY fpmul PORT map(
clk_en,
clk,
sq_out,
cos_out,
mult_out
);

adder_block1 : ENTITY fpadd PORT map(
clk_en,
clk,
mult_out,
half_x,
add_out
);
adder_block2 : ENTITY fpadd PORT map(
clk_en,
clk,
add_out,
last_result,
result
);

end architecture main;
```

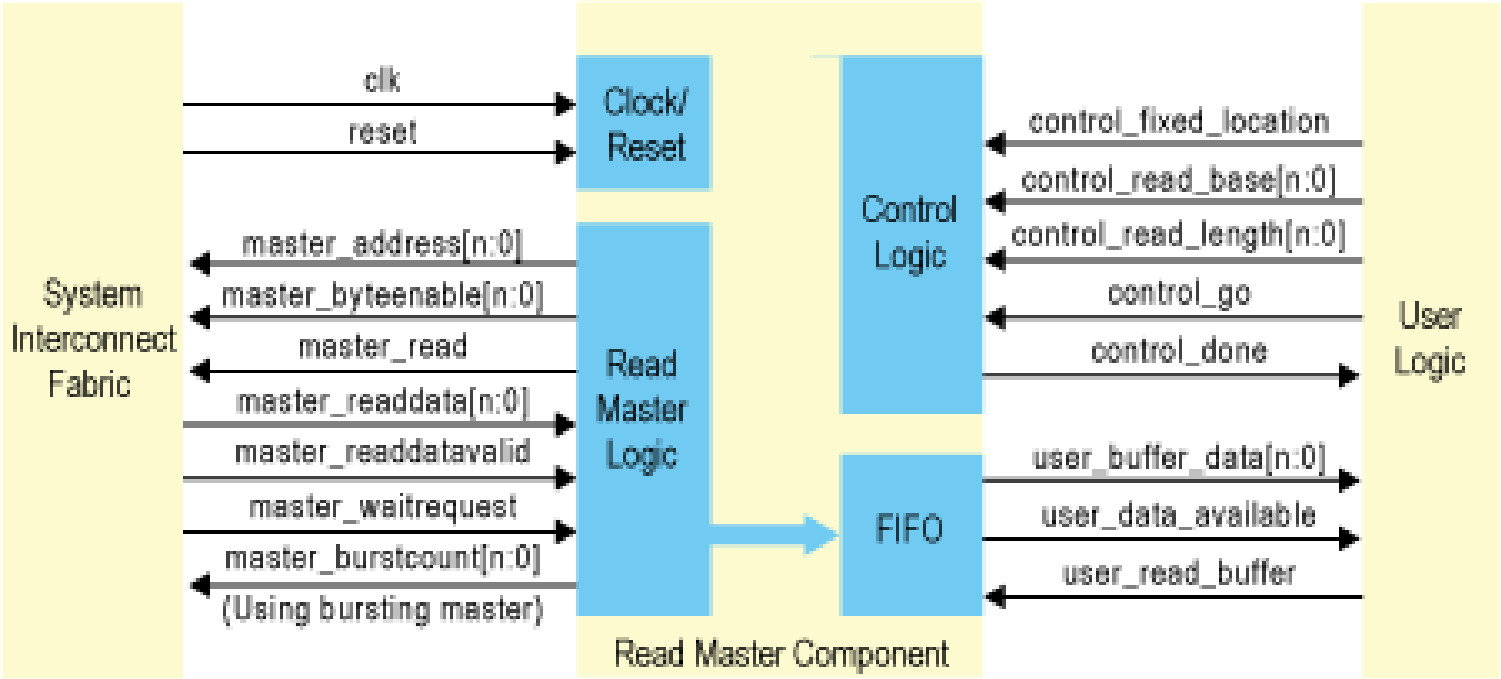
NIOS II ‘Hack’


clk
x[31..0]
rst
counter_enable
DSD_ALTFP_MULT
dataa[31..0]
datab[31..0]
clk_en
clock
aclr
result[31..0]
Clock cycles: 11
Single Precision
Exponent Width: 8
Mantissa Width: 23
x_squared
div2
x[31..0]
half_x[31..0]
dsd_fifo
data[31..0]
q[31..0]
wrreq
rdreq
full
empty
sclr
32 bits x 64 words
fifo_x_sq
fifo_div2
cos_wrapper
float_in[31..0]
float_out[31..0]
cordic
DSD_ALTFP_ADD
Clock Cycles: 7
Direction: Add
Optimization: Speed
add
x_sq_cos
my_ff
output[input_length-1..0]
input[input_length-1..0]
ff1
accum
OUTPUT
Parameter | Value | Type
input_length | 32 | Signed Integer
shift_length | 54 | Signed Integer
shift_length | 11 | Signed Integer
shift_length | 16 | Signed Integer
shift_length | 46 | Signed Integer
shift_length | 7 | Signed Integer
shift_reg
output
shift_reg_adder
shift_reg_sq
shift_reg_cosine
shift_reg_div2
shift_reg_adder_2
ff2
ff1_enable
ff2_enable
x_sq_wrreq
x_sq_rdreq
div2_rdreq

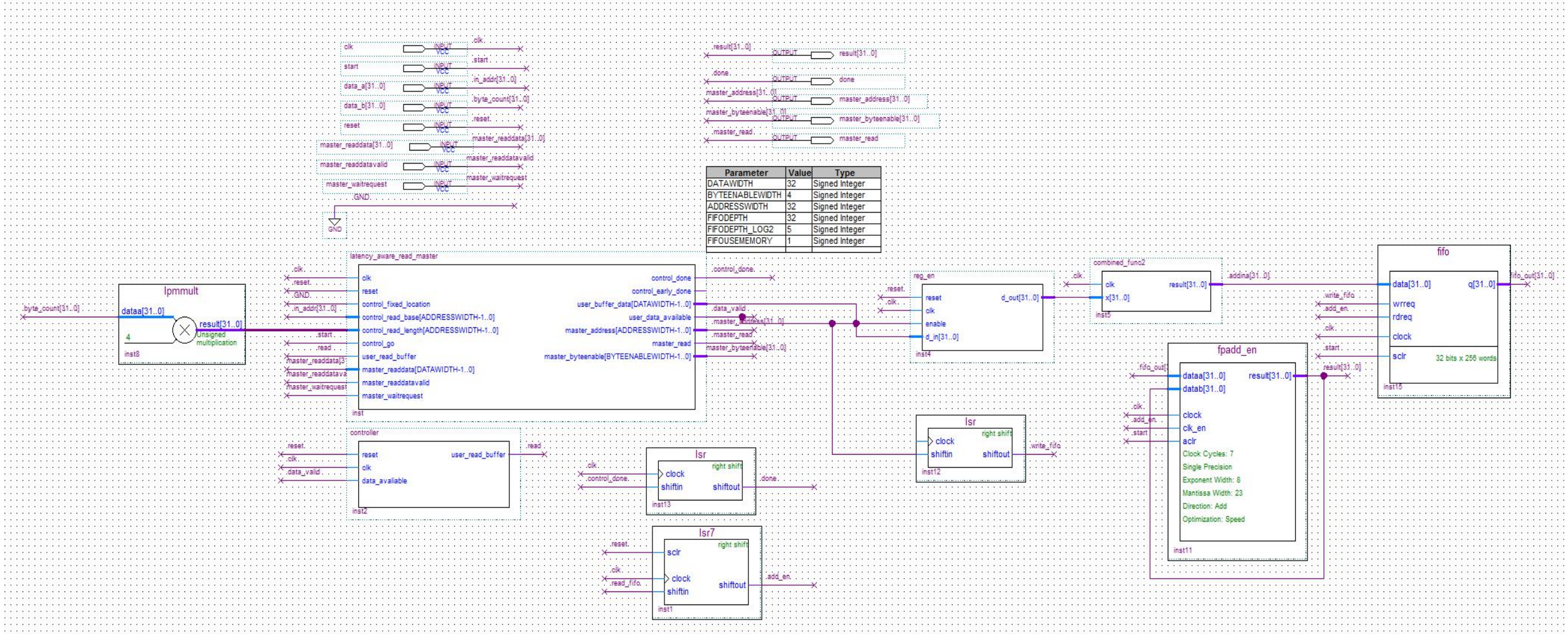
FIFO Pipe Simulation



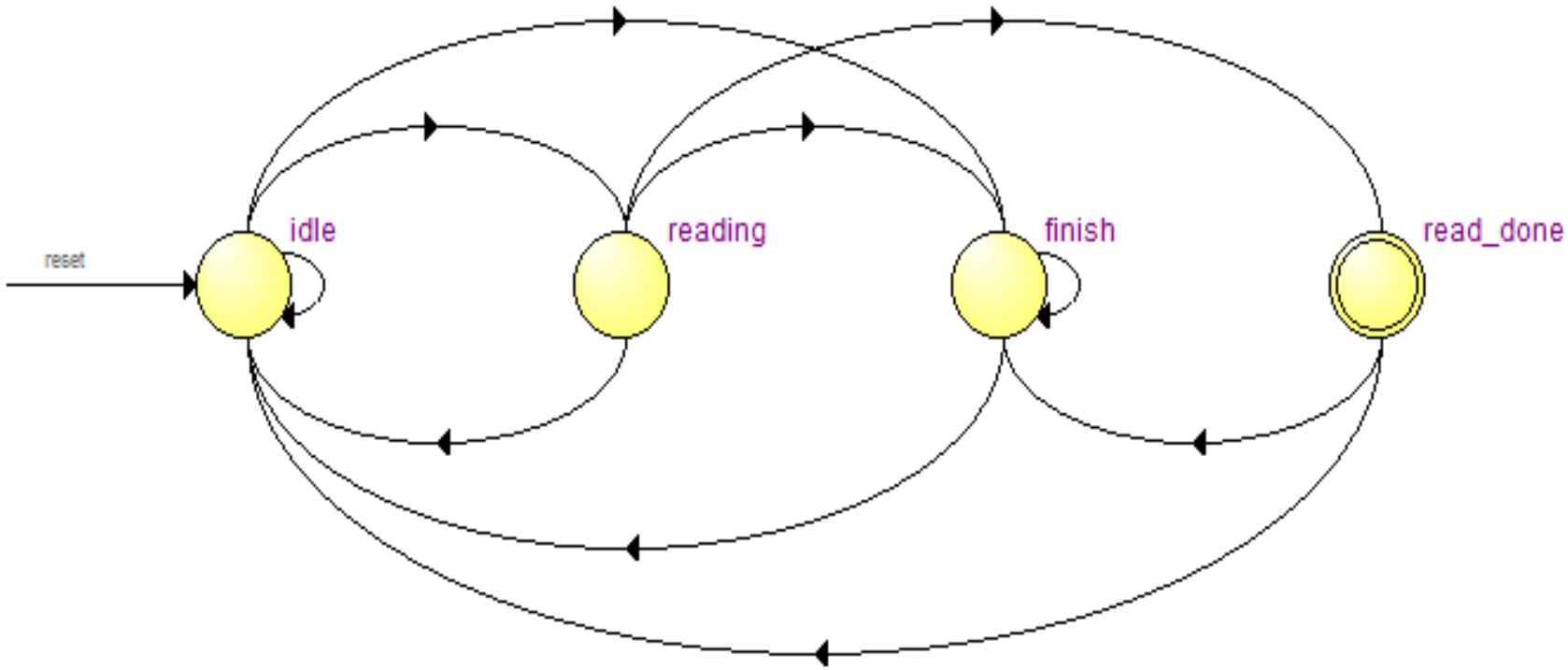
DMA Template



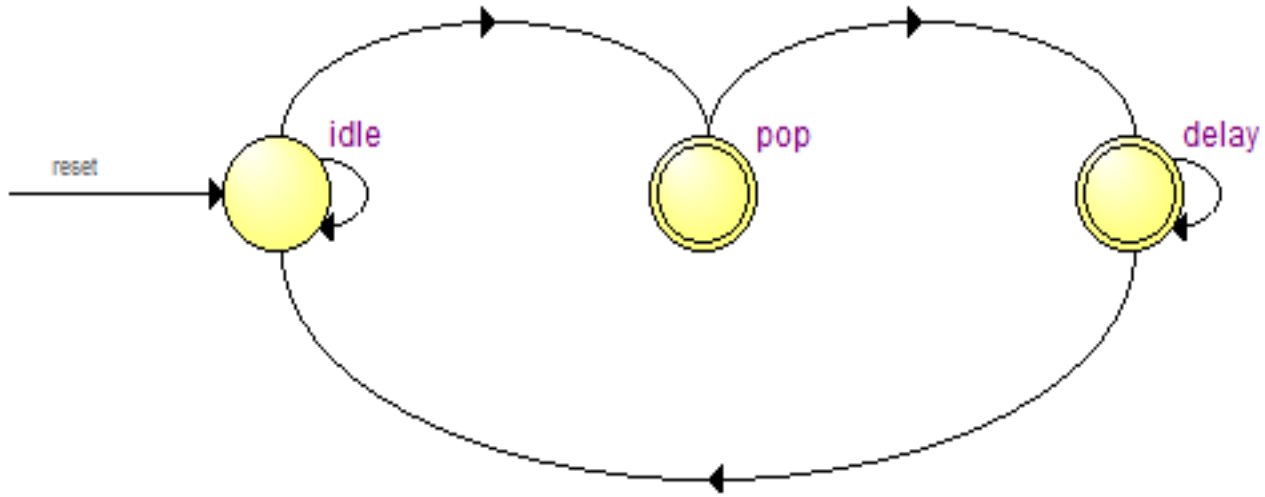
DMA BDF


clk
start
data_a[31..0]
data_b[31..0]
reset
master_readdata[31..0]
master_readdatavalid
master_waitrequest
GND
.clk.
.start.
.in_addr[31..0]
.byte_count[31..0]
.reset.
.master_readdata[31..0]
.master_readdatavalid
.master_waitrequest
.result[31..0]
.done.
.master_address[31..0]
.master_byteenable[31..0]
.master_read.
OUTPUT
result[31..0]
done
master_address[31..0]
master_byteenable[31..0]
master_read
Parameter | Value | Type
DATAWIDTH | 32 | Signed Integer
BYTEENABLEWIDTH | 4 | Signed Integer
ADDRESSWIDTH | 32 | Signed Integer
FIFODEPTH | 32 | Signed Integer
FIFODEPTH_LOG2 | 5 | Signed Integer
FIFOUSEMEMORY | 1 | Signed Integer
lpmmult
dataa[31..0]
result[31..0]
Unsigned multiplication
inst8
latency_aware_read_master
control_done
control_early_done
user_buffer_data[DATAWIDTH-1..0]
user_data_available
master_address[ADDRESSWIDTH-1..0]
master_read
master_byteenable[BYTEENABLEWIDTH-1..0]
control_fixed_location
control_read_base[ADDRESSWIDTH-1..0]
control_read_length[ADDRESSWIDTH-1..0]
control_go
user_read_buffer
master_readdata[DATAWIDTH-1..0]
inst
controller
data_avaliable
inst2
lsr
right shift
shiftin
shiftout
inst13
lsr7
sclr
inst1
reg_en
d_out[31..0]
enable
d_in[31..0]
inst4
combined_func2
x[31..0]
inst5
fpadd_en
dataa[31..0]
databa[31..0]
clock_en
aclr
Clock Cycles: 7
Single Precision
Exponent Width: 8
Mantissa Width: 23
Direction: Add
Optimization: Speed
inst11
fifo
data[31..0]
q[31..0]
wrreq
rdreq
32 bits x 256 words
inst15

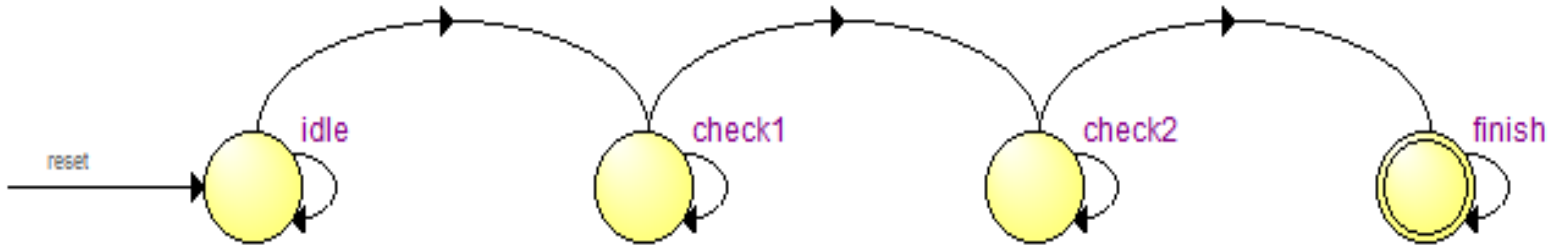
DMA Module FSM



Accumulator FSM



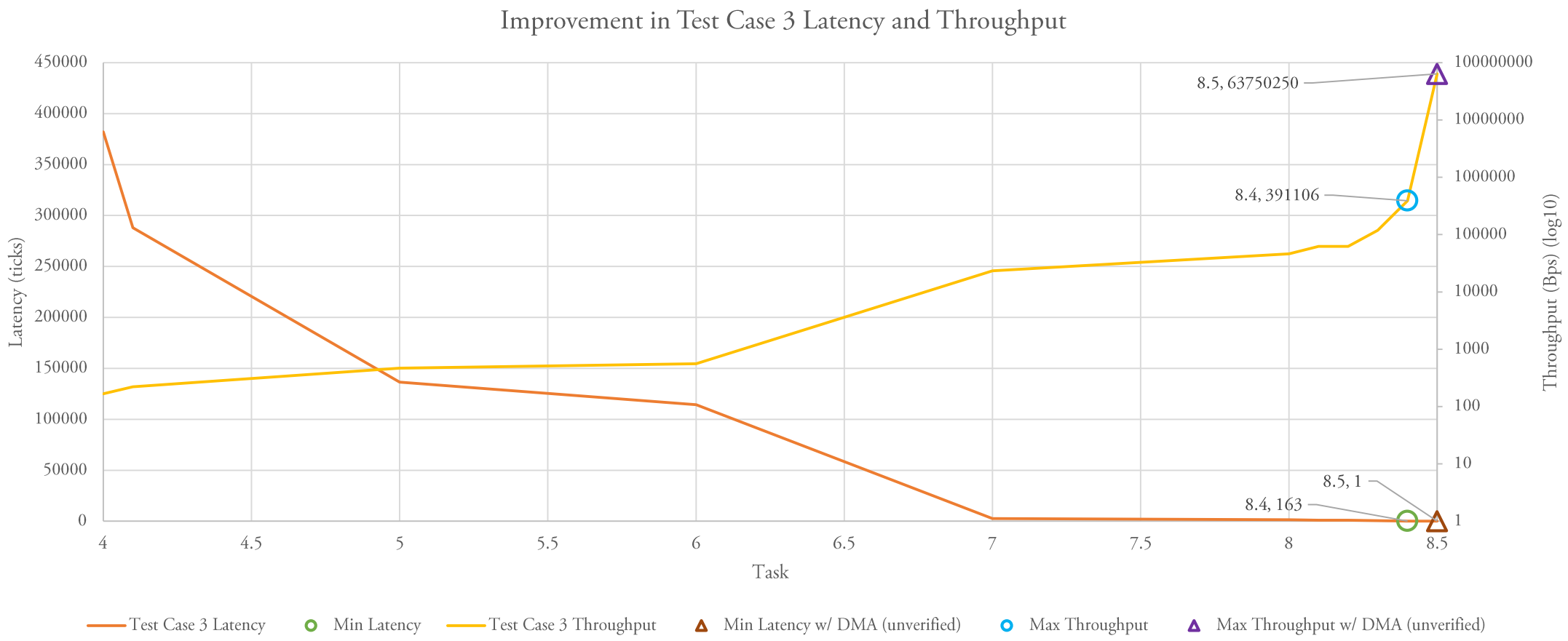
Overall FSM



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.all;

entity pipeline is
port(
--custom instruction interface
clk                  : out std_logic;
reset                : out std_logic;
start                : in  std_logic;
done                 : out std_logic;
clk_en               : in  std_logic;
in_addr              : in  std_logic_vector(31 downto 0);
n_byte               : in  std_logic_vector(31 downto 0);
result               : out std_logic_vector(31 downto 0);
--mm interface
clk_mm               : in  std_logic;
reset_mm             : in  std_logic;
master_address       : out std_logic_vector(31 downto 0);
master_read          : out std_logic;
master_byteenable    : out std_logic_vector(3 downto 0);
master_readdata      : in  std_logic_vector(31 downto 0);
master_readdatavalid : in  std_logic;
master_waitrequest   : in  std_logic
);
end entity pipeline;
architecture main of pipeline is
signal transfer_done  : std_logic;
signal ram_out        : std_logic_vector(31 downto 0);
signal full           : std_logic;
signal load_en        : std_logic;
signal cordic_in      : std_logic_vector(31 downto 0);
signal cordic_out     : std_logic_vector(31 downto 0);
signal fifo_out       : std_logic_vector(31 downto 0);
signal rdreq          : std_logic;
signal fsm_wrreq      : std_logic;
signal empty          : std_logic;
signal fifo_wrreq     : std_logic;
signal control_done   : std_logic;
signal data_avaliable : std_logic;
signal delay          : std_logic;
signal tmp_delay      : std_logic;
constant tmp_count    : std_logic_vector(31 downto 0) :=
std_logic_vector(to_unsigned(208, 32));
```

```
begin

--count_byte: entity counter
--port map(n_byte,clk_mm,start,rdreq,delay);
control_finish : entity fsm_finish port map(clk_mm, start, empty,
control_done, delay);
tmp_delay <= '0';

DMA : entity MemRead port map(clk, reset, start, transfer_done, clk_en,
in_addr, n_byte, ram_out,
                      clk_mm, reset_mm, master_address, master_read,
master_byteenable, master_readdata, master_readdatavalid, master_waitrequest,
                      full, load_en, fsm_wrreq, control_done, data_avaliable
);

data_register : process
begin
wait until clk_mm'event and clk_mm = '1';
if load_en = '1' then
cordic_in <= ram_out;
end if;
end process data_register;

fifo : entity fifo_cordic port map(
start, clk_mm, cordic_out, rdreq, fifo_wrreq, empty, full, fifo_out
);

accum : entity accumulator port map(
clk_mm, fifo_out, start, empty, result, rdreq
);

count_finish : entity lsr34 port map(
start, clk_mm, clk_en, delay, done
);

delay_wrrqt : entity lsr26 port map(
start, clk_mm, clk_en, fsm_wrreq, fifo_wrreq
);

cordic_blk : entity combined_func2 port map(clk_mm, cordic_in, cordic_out,
clk_en);

end architecture main;
```

Latency Improvement



Task 8 Latency Improvement

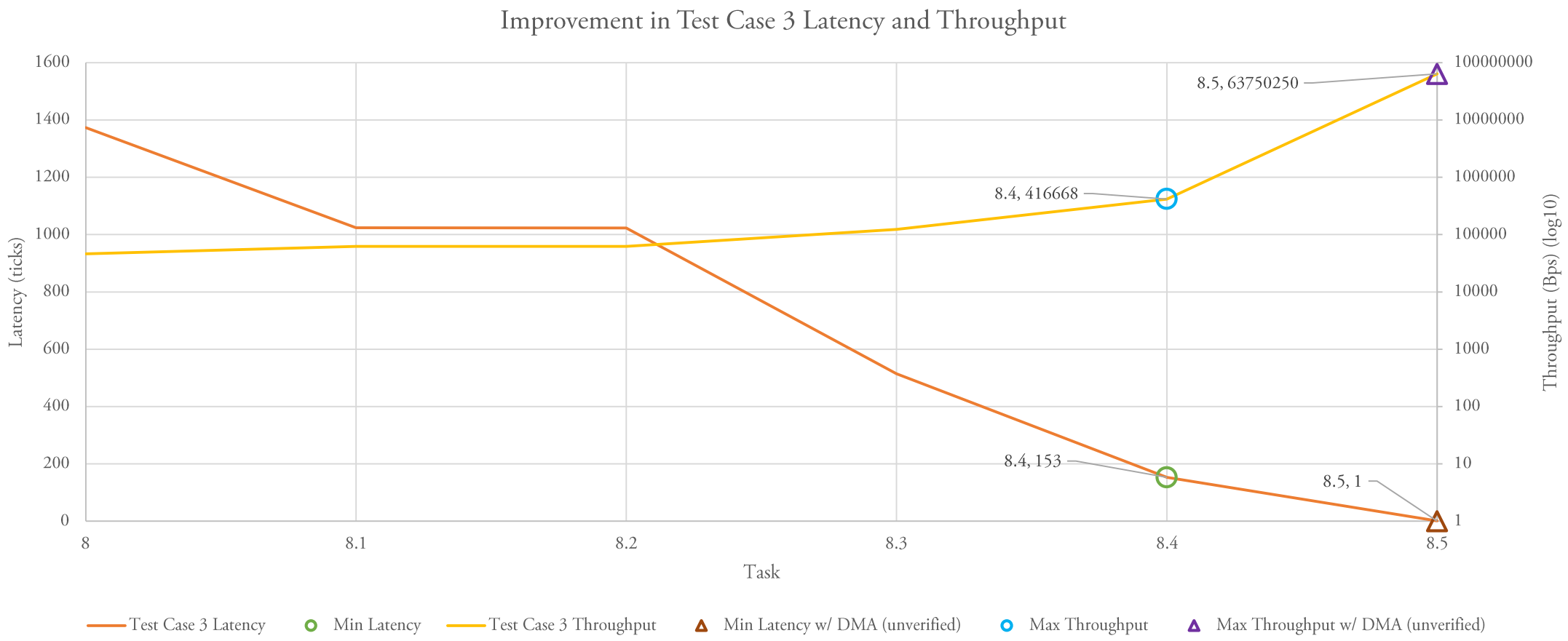


Table of Modifications

Task	Modification	Test Case 3 Latency	Test Case 3 Throughput	Latency Improvement from Reference (%)	Throughput Improvement from Design Reference (%)
4	Reference	382038	166.8688717	N/A	N/A
4.1	2KB->32KB Cache	288024	221.33659	24.6085%	32.6410%
5	Embedded Multipliers	136496	467.0484849	64.2716%	179.8895%
6	Add Floating Point Add/Mult	114332	557.588864	70.0731%	234.1479%
7	Change cos() to hardware CORDIC	2718	23454.83812	99.2886%	13955.8499%
8	Unroll CORDIC to 8*2 stages	1373	46431.3547	99.6406%	27725.0546%
8.1	Convert all to block diagram hardware	1024	62256.10352	99.7320%	37208.3984%
8.2	Use datab to reduce nested add	1023	62316.95992	99.7322%	37244.8680%
8.3	Overclocking to 100MHz	515	123786.8932	99.8652%	74082.1359%
8.4	Level 3 Optimization	153	416668.3007	99.9600%	249598.0392%
8.5	DMA	1	63750250	99.9997%	38203700.0000%

```
float sumVector_cos_custom(float x[], unsigned int M)
{
    float result=0.0;
    int i=0;
    for (i=0; i<M; i++){
        result=ALT_CI_COMPUTE_0(x[i],result);

    }
    return result;
}
```

```
printf("gen vector 3, ");
generateVector(x3, N3, S3);
printf("sum vector 3\n");
float y3=0.0;
if (task) {t5 = times(NULL); y3 = sumVector_cos_custom(x3, N3); t6 = times(NULL);}
else {t5 = times(NULL); y3 = sumVector(x3, N3); t6 = times(NULL);}
gcvt(t6-t5, 10, c);
alt_putstr("Time = "); alt_putstr(c); alt_putstr("; ");
printf("Result = %f\n", ((div_1024)?y3/1024:y3));
```