Edwin Graca (r1015462), Javkhlantugs Altankhuyag (r0872231)

Complex Digital Design

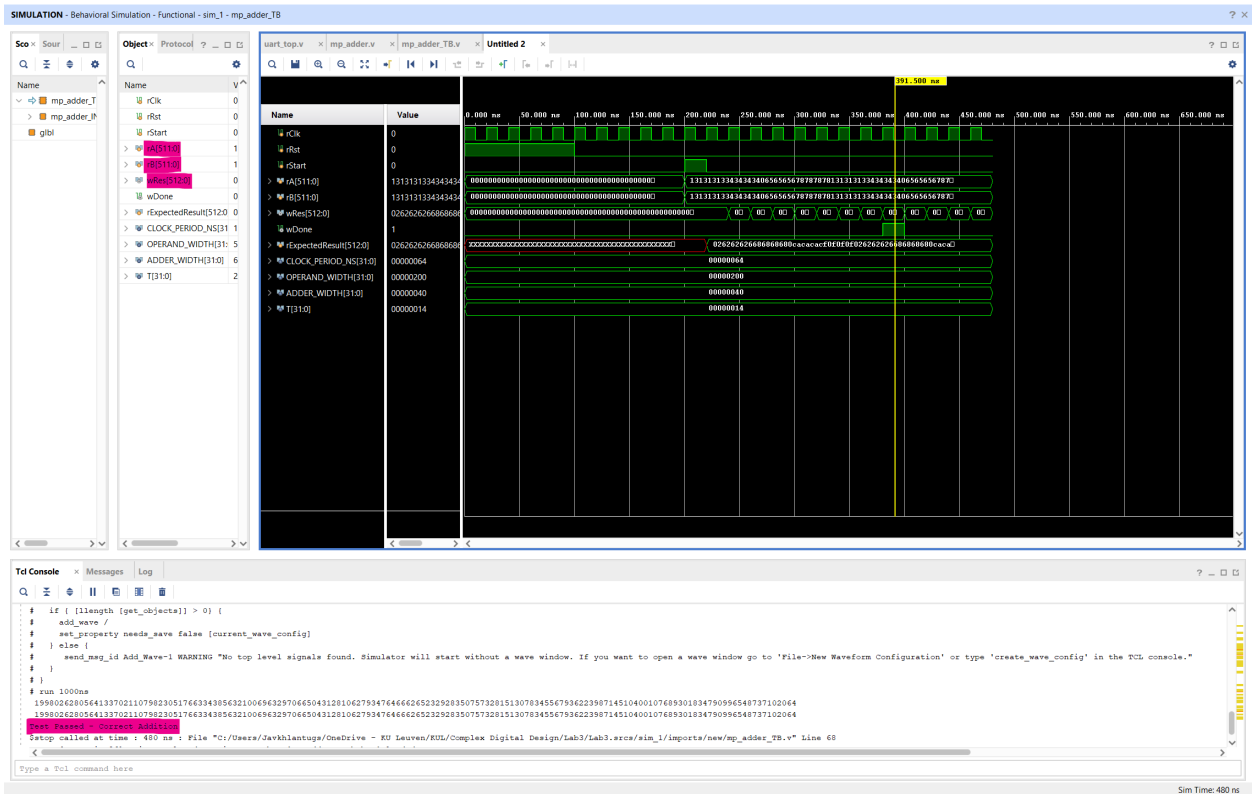
Final project

Summary

This project aims to build a combinational adder that takes two 512-bit long operands through the UART channel, adds them, and sends them back to the computer using the same UART channel. Minimum requirements were having ADDER_WIDTH of at least 32 bits long and needed to achieve all timing constraints. Fortunately, our adder design can handle up to 64-bit long ADDER_WIDTH.

We have chosen Carry-Select Adder as our improved combinational adder for the mp_adder module which was initially built with N-bit Ripple-Carry adders with Full Adders.

Our adder requires 9 clock cycles to calculate 512-bit long operand additions (see figure below) and meets all the timing constraints.



```
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL console."
#   }
# }
# run 1000ns
 1998026280564133702110798230517663343856321006963297066504312810627934764666265232928350757328151307834556793622398714510400107689301834790996548737102064
 1998026280564133702110798230517663343856321006963297066504312810627934764666265232928350757328151307834556793622398714510400107689301834790996548737102064
Test Passed - Correct Addition
$stop called at time : 480 ns : File "C:/Users/Javkhlantugs/OneDrive - KU Leuven/KUL/Complex Digital Design/Lab3/Lab3.srcs/sim_1/imports/new/mp_adder_TB.v" Line 68
```

Figure 1: mp_adder simulation.

Edwin Graca (r1015462), Javkhlantugs Altankhuyag (r0872231)

Technical description

We took the Uniform-Sized Carry-Select adder as our reference design.

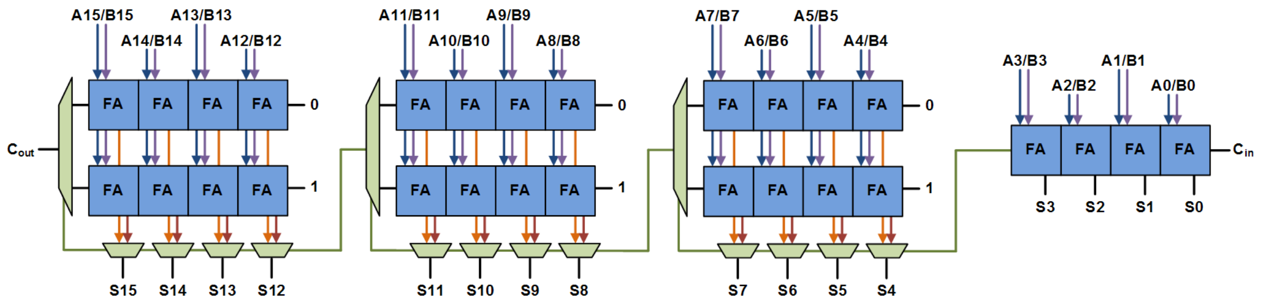


Figure 2: Uniform-sized Carry-Select adder.

UART (uart_top) Module:

The uart_top is responsible for coordinating UART communication, including receiving data (iRx) from an external device and transmitting data (oTx) to the external device. It interfaces with both the UART receiver (uart_rx) and UART transmitter (uart_tx) modules to facilitate communication. It instantiates the mp_adder module to perform the addition of 2 operands.

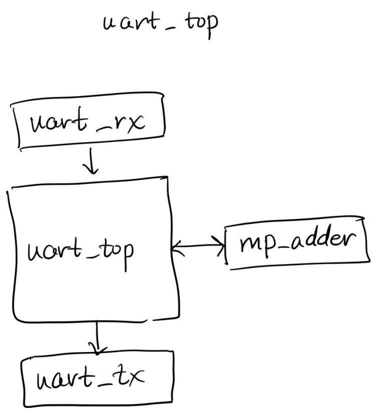


Figure 3: The uart_top high-view diagram.

Multi-Precision adder (mp_adder) Module:

The mp_adder module takes iOpA and iOpB of width OPERAND_WIDTH = 512 each and adds them together to produce the result oRes. Once mp_adder receives both operands through the UART channel, iStart is asserted to be true and the addition process will start. It uses FSM to control the addition process. The module generates the oDone signal when the addition process is completed.

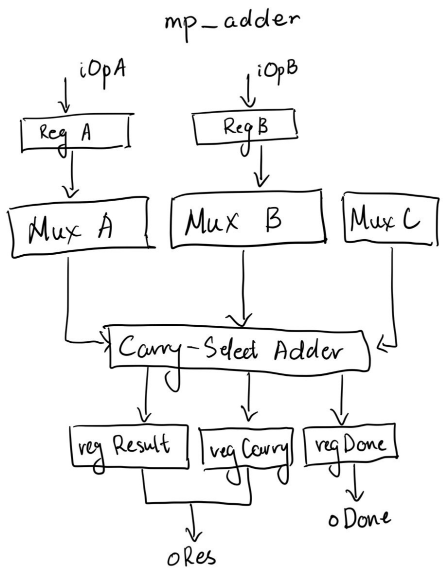


Figure 4: The mp_adder high-view diagram.

Carry-Select adder (carry_select_adder) Module:

The carry_select_adder takes the width of ADDER_WIDTH up to 64 bits before failing to meet the timing constraints. Our carry_select_adder has a uniform block size of 8. Since carry-in is known at the beginning of computation, a carry select block is not needed for the first eight bits.

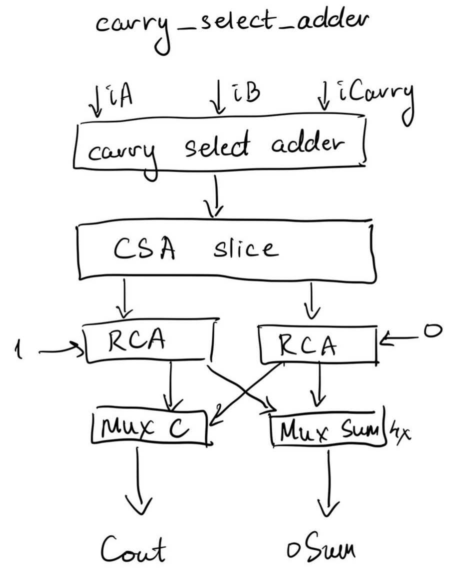
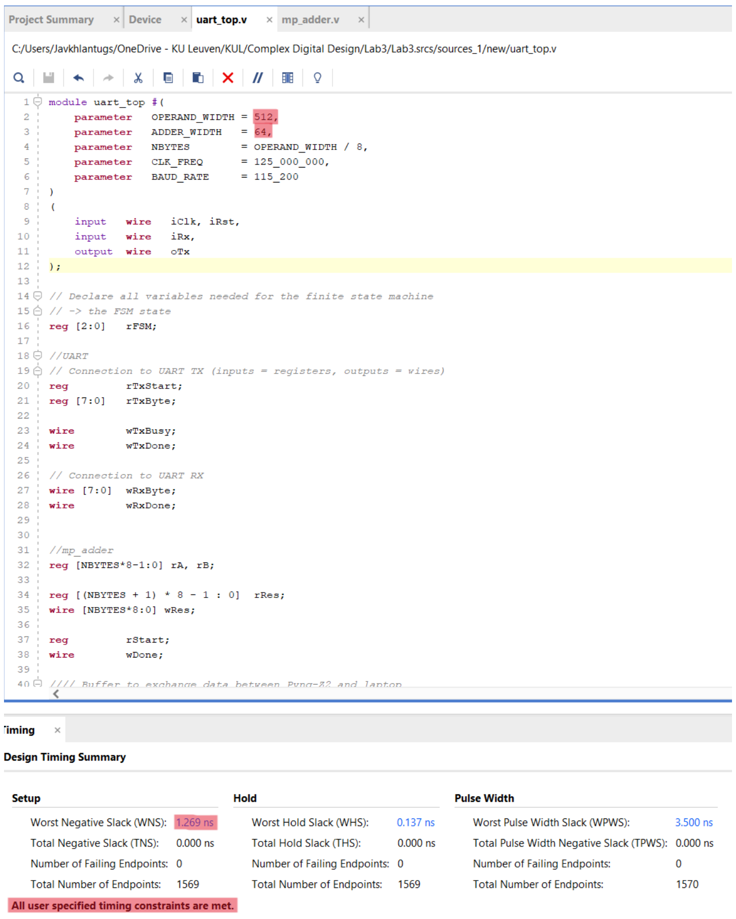


Figure 5: The carry_select_adder module high-view diagram.

Edwin Graca (r1015462), Javkhlantugs Altankhuyag (r0872231)

In our design, with OPERAND_WIDTH = 512 and ADDER_WIDTH = 64, it instantiates 512/64 = 8 carry_select_adder modules. The CSA_slice module is to depict 1 block of 2 Ripple carry adders, 2 carry inputs of 0 and 1, mux switches for each full adder sum and 1 mux switch for carry out.

Performance evaluation



C:/Users/Javkhlantugs/OneDrive - KU Leuven/KUL/Complex Digital Design/Lab3/Lab3.srcs/sources_1/new/uart_top.v

```verilog
module uart_top #(
    parameter   OPERAND_WIDTH = 512,
    parameter   ADDER_WIDTH   = 64,
    parameter   NBYTES        = OPERAND_WIDTH / 8,
    parameter   CLK_FREQ      = 125_000_000,
    parameter   BAUD_RATE     = 115_200
)
(
    input   wire   iClk, iRst,
    input   wire   iRx,
    output  wire   oTx
);

// Declare all variables needed for the finite state machine
// -> the FSM state
reg [2:0]   rFSM;

//UART
// Connection to UART TX (inputs = registers, outputs = wires)
reg         rTxStart;
reg [7:0]   rTxByte;

wire        wTxBusy;
wire        wTxDone;

// Connection to UART RX
wire [7:0]  wRxByte;
wire        wRxDone;


//mp_adder
reg [NBYTES*8-1:0] rA, rB;

reg [(NBYTES + 1) * 8 - 1 : 0]  rRes;
wire [NBYTES*8:0] wRes;

reg         rStart;
wire        wDone;

//// Buffer to exchange data between Pynq-Z2 and laptop
```

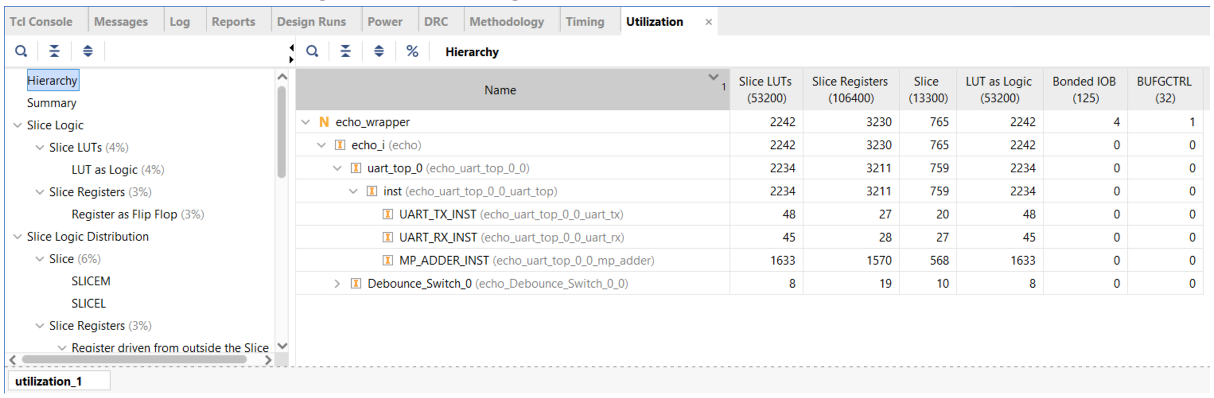
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.269 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1569	Total Number of Endpoints:	1569	Total Number of Endpoints:	1570

All user specified timing constraints are met.

Figure 6: Timing summary report.

As shown in Figure 6, all timing constraints are met with 64 ADDER_WIDTH.



Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (125)	BUFGCTRL (32)
echo_wrapper	2242	3230	765	2242	4	1
echo_i (echo)	2242	3230	765	2242	0	0
uart_top_0 (echo_uart_top_0_0)	2234	3211	759	2234	0	0
inst (echo_uart_top_0_0_uart_top)	2234	3211	759	2234	0	0
UART_TX_INST (echo_uart_top_0_0_uart_tx)	48	27	20	48	0	0
UART_RX_INST (echo_uart_top_0_0_uart_rx)	45	28	27	45	0	0
MP_ADDER_INST (echo_uart_top_0_0_mp_adder)	1633	1570	568	1633	0	0
Debounce_Switch_0 (echo_Debounce_Switch_0_0)	8	19	10	8	0	0

Figure 7: Utilization report.

As shown in Figure 7, the utilization of slice LUTs, registers, and slices is moderate, with 1633 slice LUTs, 1570 slice registers, and 568 slices used.

Comparison

We have used the reference case shown in cdd.balasch.be :

Worst-case Delay: 5.331ns (ADDER_WIDTH = 16)

Area Costs: 1073 LUTs and 1573 FFs.

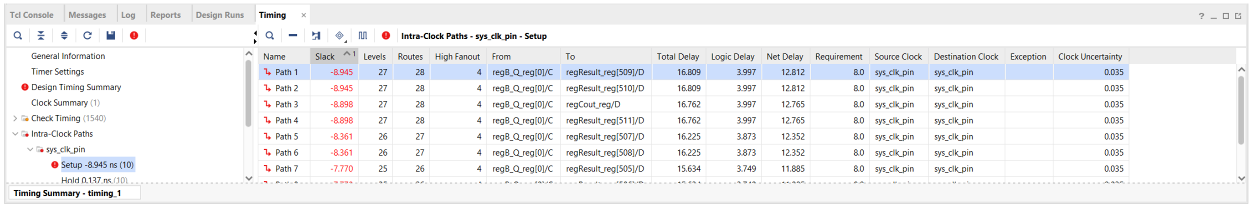
Our Design:

Worst-case Delay: 6.595ns (ADDER_WIDTH = 64)

Area Costs: 1633 LUTs and 1570 FFs.

In the reference case with the Ripple Carry Adder, the worst-case delay was 5.331ns, whereas in the current case with the Carry Select Adder, the worst-case delay is significantly reduced to 6.595ns.

This speed decreases due to the ADDER_WIDTH difference (4 times longer ADDER_WIDTH).



Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	-8.945	27	28	4	regB_Q_reg[0]/C	regResult_reg[509]/D	16.809	3.997	12.812	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 2	-8.945	27	28	4	regB_Q_reg[0]/C	regResult_reg[510]/D	16.809	3.997	12.812	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 3	-8.898	27	28	4	regB_Q_reg[0]/C	regCout_reg/D	16.762	3.997	12.765	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 4	-8.898	27	28	4	regB_Q_reg[0]/C	regResult_reg[511]/D	16.762	3.997	12.765	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 5	-8.361	26	27	4	regB_Q_reg[0]/C	regResult_reg[507]/D	16.225	3.873	12.352	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 6	-8.361	26	27	4	regB_Q_reg[0]/C	regResult_reg[508]/D	16.225	3.873	12.352	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 7	-7.770	25	26	4	regB_Q_reg[0]/C	regResult_reg[505]/D	15.634	3.749	11.885	8.0	sys_clk_pin	sys_clk_pin		0.035

Figure 8: Post-synthesis timing report.

A better speed comparison would be when two adders have the same ADDER_WIDTHs. In Figure 8, a reference design with a 64-bit ADDER_WIDTH timing report is displayed showing 16.809ns. Our design offers 2.54 times faster addition than the original reference design.

In the reference case with the Ripple Carry Adder, the area costs were 1073 LUTs and 1573 FFs.

In the current case with the Carry Select Adder, the area costs are 1633 LUTs and 1570 FFs.

While the number of LUTs has increased ~55% in the current case, the number of FFs remains similar. An increase in the number of LUTs is expected as the Carry Select Adder may require more logic resources compared to the traditional Ripple Carry Adder. However, the overall area increase is within acceptable limits, considering the significant speed improvement achieved.