TSEK06
Transistor-Level Design Report

Group 5

Editor: Johannes Klasson

Version P1A

Status

Reviewed	Johannes Klasson	2016-03-10
Approved	Martin Nielsen-Lönn	-

PROJECT IDENTITY

VT, 2016, Group 5
Linköpings Tekniska Högskola, ISY

Group members

Name	Responsibility	Phone	E-mail
Johan Isaksson	Project Leader	070-2688785	johis024@student.liu.se
Johannes Klasson	Document Manager	073-8209003	johkl226@student.liu.se
Jonas Tarassu	VLSI Designer	070-5738583	jonta760@student.liu.se
Alexander Yngve	VLSI Designer	076-2749762	aleyn573@student.liu.se

Customer: ISY
Contact at customer: Martin Nielsen-Lönn
Course resposible: Atila Alvandpour
Consultant: Martin Nielsen-Lönn

Contents

Document history

Version	Date	Changes	Performed by
P1A	2016-03-10	First draft	Johan Isaksson

1 Introduction

This document describes the state of the 16-bit Kogge-Stone adder project in the course TSEK06 after finishing the transistor level design. The meaning of transistor level is that the every basic logic gate is implemented with 0.35 µm CMOS transistors. The main reason for doing this is to be able to simulate all logic to make sure that everything works as intended. Updated block diagrams can be found in section 2, simulation results in section 3 together with appendix B. A small risk analysis can be found in section 5. In appendix A the time plan for the next phase can be found.

2 Block Level Description

Much of the block level descriptions can be seen in the high level report, but the transistor view of the leaf-cells will be described in this chapter. To find good sizes for our gates we used a very simple sizing strategy. Start small, and if the signal is to weak to drive the components, we just size it up and if necessary, make a buffer for it. The transistor schematic of the basic blocks like AND, OR, DFF etc. are simple enough to leave out the description for them. In Fig. 1 an updated block diagram of the complete system can be seen.

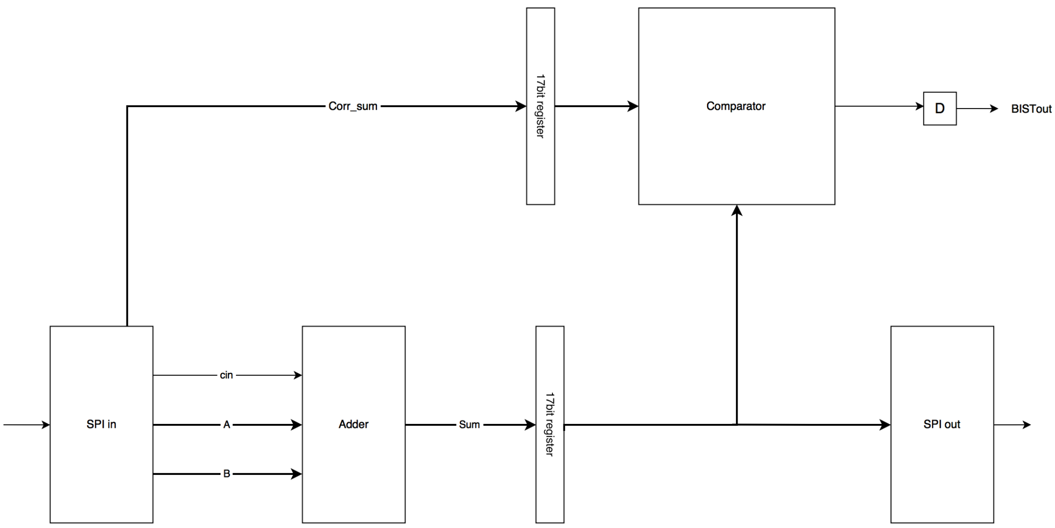

Figure 1 – Top level block diagram.

The updated diagram contains additional registers for synchronizing the signal before and after the comparator. This was done to provide more stable signals to the comparator and to have a more easily interpreted BISTout signal. The drawback is that the BISTout signal is available two clock cycles after the addition took place which isn't very much of a problem.

2.1 SPI-in/PRBS

This module only use basic leaf-cells at the transistor level. However, there are a few noteworthy things regarding the sizing of these basic blocks. Some of the signals are connected to a large amount of devices (60+), which means that the signal gets really weak. This was solved by just making the gates a bit bigger, or by using a buffer. SPI_en is using a 9x buffer, SPI_clk is using a 27x buffer and test_mode is using a 9x buffer. Clk_en is the result from an OR gate, and the NOR gate inside it is 6 times bigger and the inverter is 18 times bigger than our reference inverter.

2.2 SPI out

This chapter will describe the SPI output module.

2.2.1 Shift register

The output consist of a 68 bit shift register where each cell in the register contains one D flip-flop and one multiplexer. As we have transistor schematics for both the flip flop and the multiplexer nothing had to be changed to the individual cells.

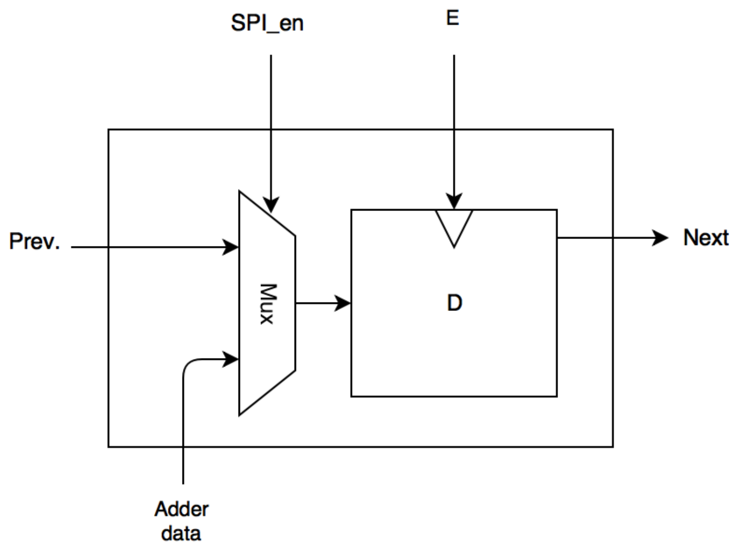

Figure 2 – Shift register cell

2.2.2 Control logic

As all verilogA code were replaced by transistor schematics, the problem with this design got exposed. Each of the four enable signal has a large fan out as they are connected to 17 cells in the shift register.
The solution was to size the the multiplexer generating the control signals. The last internal block that includes this multiplexer can be seen in Fig. 3.

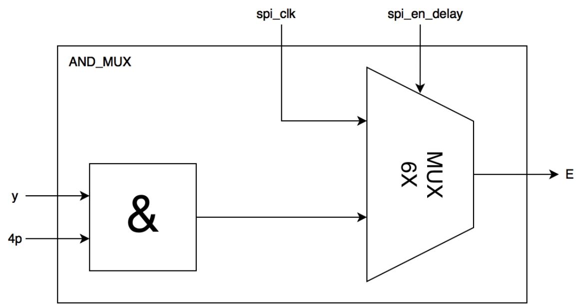

Figure 3 – Functional block containing an AND and a MUX.

By testing we found that a MUX that is approximately six times larger should be sufficient. In Fig. 4 the schematic of the sized multiplexer, implemented using NAND and an inverter is shown.

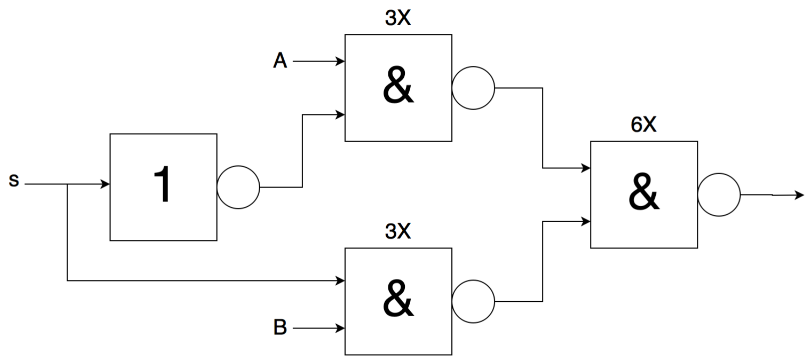

Figure 4 – Sized multiplexer.

2.2.3 Protocol

Due to a misunderstanding in the group regarding if the SPI clock should be kept high or low during the high time of the SPI enable signal, the data on the SPI output is now available for read already on the first falling edge of the SPI clock. This is possible because the the clock is kept low during high time of SPI enable so that the data can be written to the output at the first rising edge as usual but the first falling edge will come afterwards instead.

2.3 16-bit Kogge-Stone Adder

The Kogge-Stone adder consists of several simple blocks connected in a complex way. The adder has been significantly changed since the high-level design phase. The *yellow* block has been split into two blocks *yellow_inv_in* and *yellow_inv_out*, which can be seen in Fig. 5. The *yellow_inv_in* block takes inverted input signals and gives non-inverted output. The *yellow_inv_out* block takes non-inverted inputs and gives inverted output. This arrangement saves a lot of gates. The *yellow_carry* block has been split in the same way.

Because of the inverted signals from *yellow_inv_out* some *sum* blocks have been replaced with XNOR gates. A couple of inverters have also been added in some places to make sure the new blocks gets the correct input.
All transistors in the adder are minimum sized to achieve maximum performance when it comes to speed. This is possible since the gates have a low fan out.

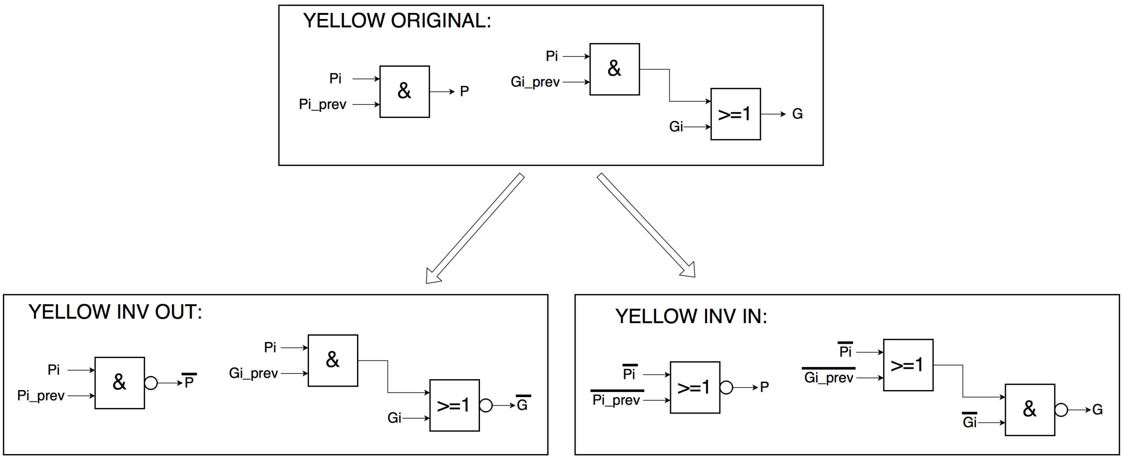

Figure 5 – The new yellow blocks.

To save space, new switch nets were created for the generate calculation of the *yellow* blocks. The new nets can be seen in Fig. 6 and 12. By doing this the transistor count is cut in half.

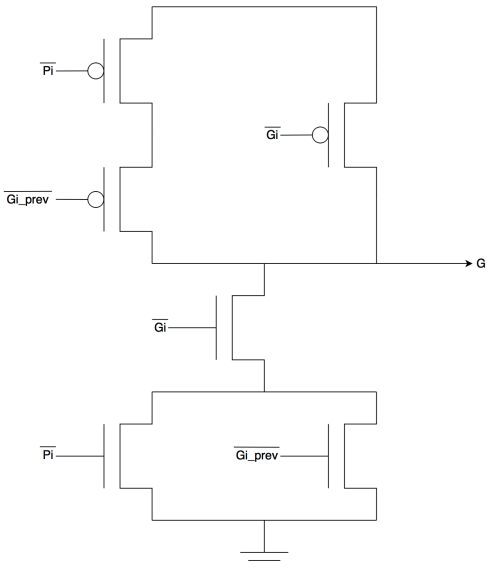

Figure 6 – Generate part of *yellow_inv_in*.

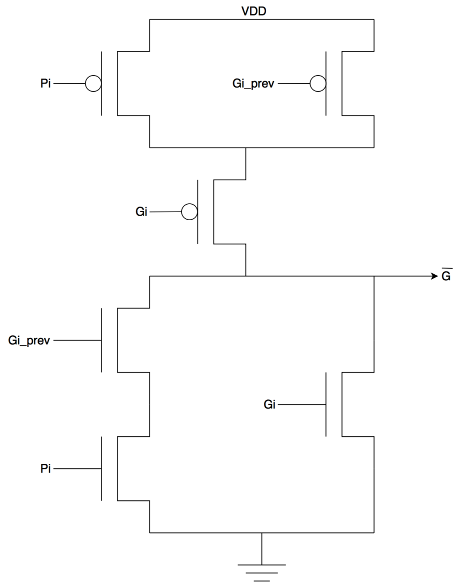

Figure 7 – Generate part of *yellow_inv_out*.

3 Simulation Results

This section describes the high level simulation results. All files referenced to in this section can be found in the attached zip-file.

3.1 SPI In

The first thing to test in the system is where it all begins, at the input. The basics of it can be seen in test_spi_receive. As can be seen, as soon as the SPI_enable signal goes low and the SPI_clk starts we start to receive one bit on every positive edge. The data is then shifted trough all of the 16 registers. As the 16th bit is shifted in, the first load signal is triggered. Every 16 bits after that another load signal is triggered, and this can be seen in test_spi_load. One can also see that once the last load signal is triggered, SPI_enable goes high again.
The last thing in the SPI_in module to test is how the data travels out of the PRBS registers. This can be seen in test_spi_prbs. One important thing to note is that as soon as SPI_enable goes high, the registers are triggered on the system clock. As one can see in the figure, the first bit is ready for a long time, and as soon as the last bit is ready, we start to add at full speed. And after the four bits are done the system continues to add the pseudo random numbers.

3.2 Kogge-Stone Adder

The simulation of the adder can be seen in Appendix B Fig. 13. The input sequence for A, B and cin is `0xFFFF`, `0x0000` and `1` respectively. This sequence should generate the worst case propagation delay which in our case, as can be seen in the simulation, is 1.36 ns. The reason to why the propagation delay is measured between the LSB bit of the A signal and cout is because cin is generated through an ideal voltage source which is set to high in this case. The A signal on the other hand is generated through a clocked test generation block written in VerilogA. The BISTout signal, which indicates if the addition was carried out successfully or not goes high two clock pulses after the data was fed into the adder, just as intended.

3.3 Comparator

The simulation of the comparator was left out since the comparator block is part of the test bench for the adder. All outputs from the adder goes into the comparator together with a Corr-sum signal. If these two signals matches, meaning the addition was carried out correctly in the adder, BISTout goes high.

3.4 SPI Out

The critical parts of this module are the events after a transition on the spi enable signal.
In the image spi_out_control a simulation of the behaviour when the SPI enable goes high. The simulation shows that the buffering of the signals successfully achieved satisfying rise and fall times For the enable signals.
When spi enable goes low, the four enable signals are the same as the spi clock which can be seen in the image spi_out_control2.

3.5 Top Level

One can get a overall picture of the behaviour of the system by looking at at the simulations in the order spi_receive, spi_prbs and last the spi_out. This is possible because all this simulations are part of a bigger one. The Kogge-Stone adder was left out here for simplicity.

4 Pad Assignment and Early Test Plan

Table 1 shows the pin assignments for the chip.

Table 1 – Pin assignments

Name	Direction	Type	Description
Vdd1	INOUT	Analog	Will provide most of the system with power and will be a steady 3.3 V.
Vdd2	INOUT	Analog	Will provide the adder with power and it might vary from 3.3 V downto threshold-voltage.
GND	INOUT	Analog	Ground.
Clk	IN	Digital	This is the clock for the adder, some registers and control logic. Should have a frequency of at least 200 MHz at 3.3 V. Will be lower as we decrease the voltage of Vdd2.
SPI_clk	IN	Digital	This clock is used by the input and output unit and should be at least five times slower than the system clock. Should also be low if SPI_en is inactive.
SPI_en	IN	Digital	Active low. Should go high on the first negative flank of SPI_clk after the last value is read.
SPI_in	IN	Digital	Updates it's value as soon as SPI_en goes low, and should have it's value ready on the first positive flank of SPI_clk, since this is when we read the value. The value of SPI_in should then be updated on every negative flank of SPI_clk.
SPI_out	OUT	Digital	The data is available for read on the first falling edge of SPI_clk after SPI_en has gone active.
BISTout	OUT	Digital	If the IN-data is correct, BISTout should be constant high after the first addition is done until the the PRBS-bit is set.
Cin	IN	Digital	Used to measure propagation delay.
Cout	OUT	Analog	Used to measure propagation delay.
Sum15	OUT	Analog	Used to measure propagation delay.

5 Risks and Delays

During the transistor level design there hasn't been any considerable delays. The only thing that is an issue is that all members in the group has quite different schedules which sometimes can hinder cooperation. However the group has solved this by using good tools for project tracking (Trello) and communication (Slack). This has helped considerably with the delegation of tasks and keeping track of what needs to be done.
During the next phase of the project, cooperation will be more important since the tasks will be harder. We will need to plan ahead and schedule occasions where we all can meet and work together.

A Time Plan

Planning

Project: 16 bit Kogge-Stone ad

Project group: 5	**Date: 160310**	**Reviewed:**
Customer: Martin Nielsen-Lön	**Version: P3A**	Johannes Klasson
Course: TSEK06	**Author: JI**	

ACTIVITIES		TIME	WHO	TIME PLAN (week number)																		
no	Description	hours	Initials	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
1	Define structure of the SPI unit	10	JI, JK		9,5			2,														−2
2	Implement counters in Verilog-A	10	JI, JK		4																	6
3	Implement control logic in Verilog-A	10	JI, JK			10																0
4	Implement 1:4 decoder in Verilog-A	10	JI, JK			5,																4,5
5	Integrate to high level design of SPI	10	JI, JK			17																−7
6	Simulation and test of high level design (SPI)	5	JI, JK				13	6														−1
7	Implement transistor level design of the SPI unit	30	JI, JK					6	1,													22,
8	Simulation and test of transistor design (SPI)	20	JI, JK						12	0												7,5
9	Implement layout level design of SPI unit	30	JI, JK										25	25	25	25						−7
10	Simulation and test of layout (SPI)	20	JI, JK														20					0
19	Define structure of the adder	10	JT, AY		3																	7
20	Implement Generate calculation logic in Verilog-A	10	JT, AY		1																	9
21	Implement Propagate calculation logic in Verilog-A	10	JT, AY		1																	9
22	Implement Sum calculation logic in Verilog-A	10	JT, AY		1																	9
23	Integrate to high level design of adder	20	JT, AY				17															3
24	Simulation and test of high level design (adder)	20	JT, AY				3,			32												−1
25	Implement transistor level design of the adder unit	40	JT, AY					12	7													21
26	Simulation and test of the transistor design (adder)	20	JT, AY					3	12	0												5
27	Implement layout level design of adder unit	40	JT, AY										25	20	20	20						−4
28	Simulation and test of layout (adder)	20	JT, AY														20					0
29	Define structure of the comparator	5	JT		0																	5
30	Implement bit comparator in Verilog-A	5	JI			0																5
31	Integrate to high level design of the comparator	10	JI			2																8
32	Simulation and test of the high level design (comparator)	5	JI				2															3
33	Implement transistor level design of the comparator unit	20	JT, AY					0	0													20
34	Simulation and test of the transistor design (comparator)	10	JT, AY							0												10
35	Implement layout level design of comparator unit	20	JT, AY										20	20								−2
36	Simulation and test of layout (comparator)	10	JT, AY												10							0
37	Off-chip hardware interface	30	JI, JH, JT, AY															15	15			0
38	Documentation and presentation	60	JI, JH, JT, AY	28			35	14	2	19									10	20		−6
39	Meetings	60	JI, JH, JT, AY		11	0	0	0	5,	4,5			4	4	4	4	4	4	4	4	4	3
40	Buffer time	80	JI, JH, JT, AY					6					5	5	5	5	5	5	5	10	0	29
41	High level integration (System)	15	JI, JH, JT, AY				14															1
42	Transistor level integration	10																				10
43	Layout level integration	15	JI, JH, JT, AY															30				−1
44	Implementation of test bench for SPI	5	JH			0																5
45	Implementation of test bench for generator	5	JI		0																	5
46	Implementation of testbench for adder	10	AY		0			5														5
47	Implementation of test bench for comparator	5	JT		0																	5
48	Implementation of test bench for the complete system	20	JI, JH, JT, AY				15		7													−2
	Sum, number of hours	755			30,5	34	99	55	47	55,	0	0	79	74	64	54	49	54	34	34	4	0

B Appendix B

B.1 SPI In

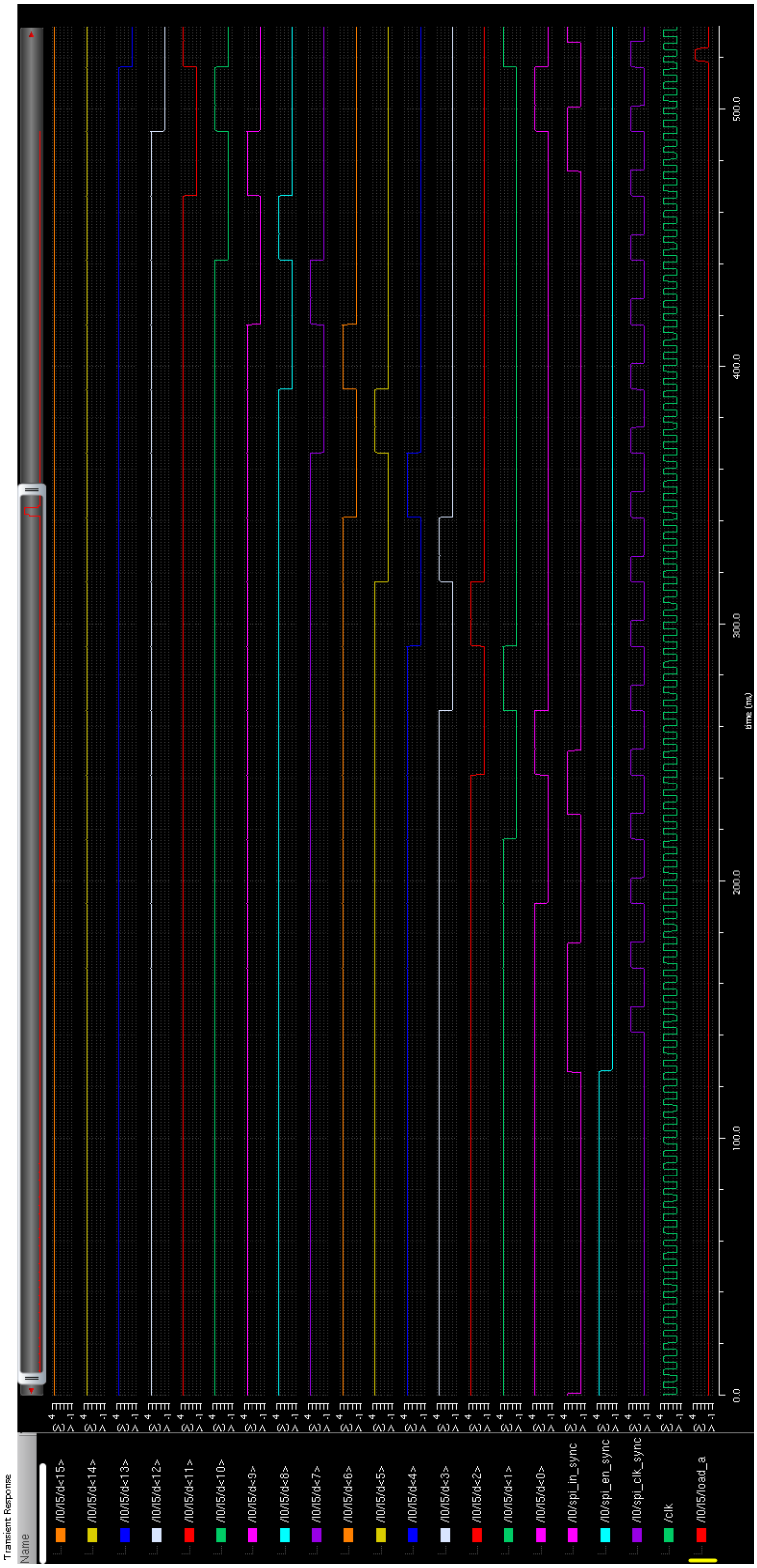

Figure 8 – SPI receiver

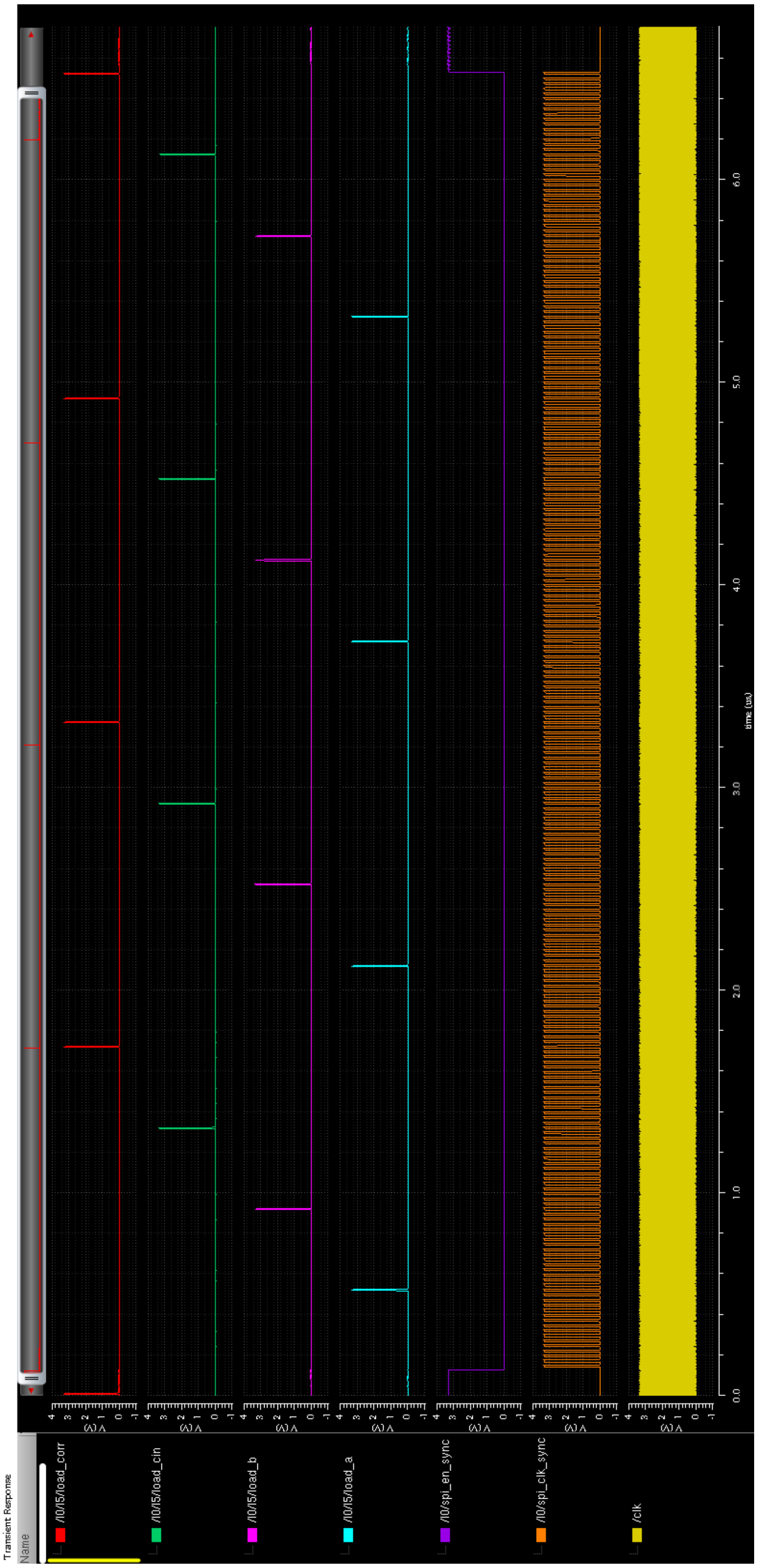

Figure 9 – Load signals

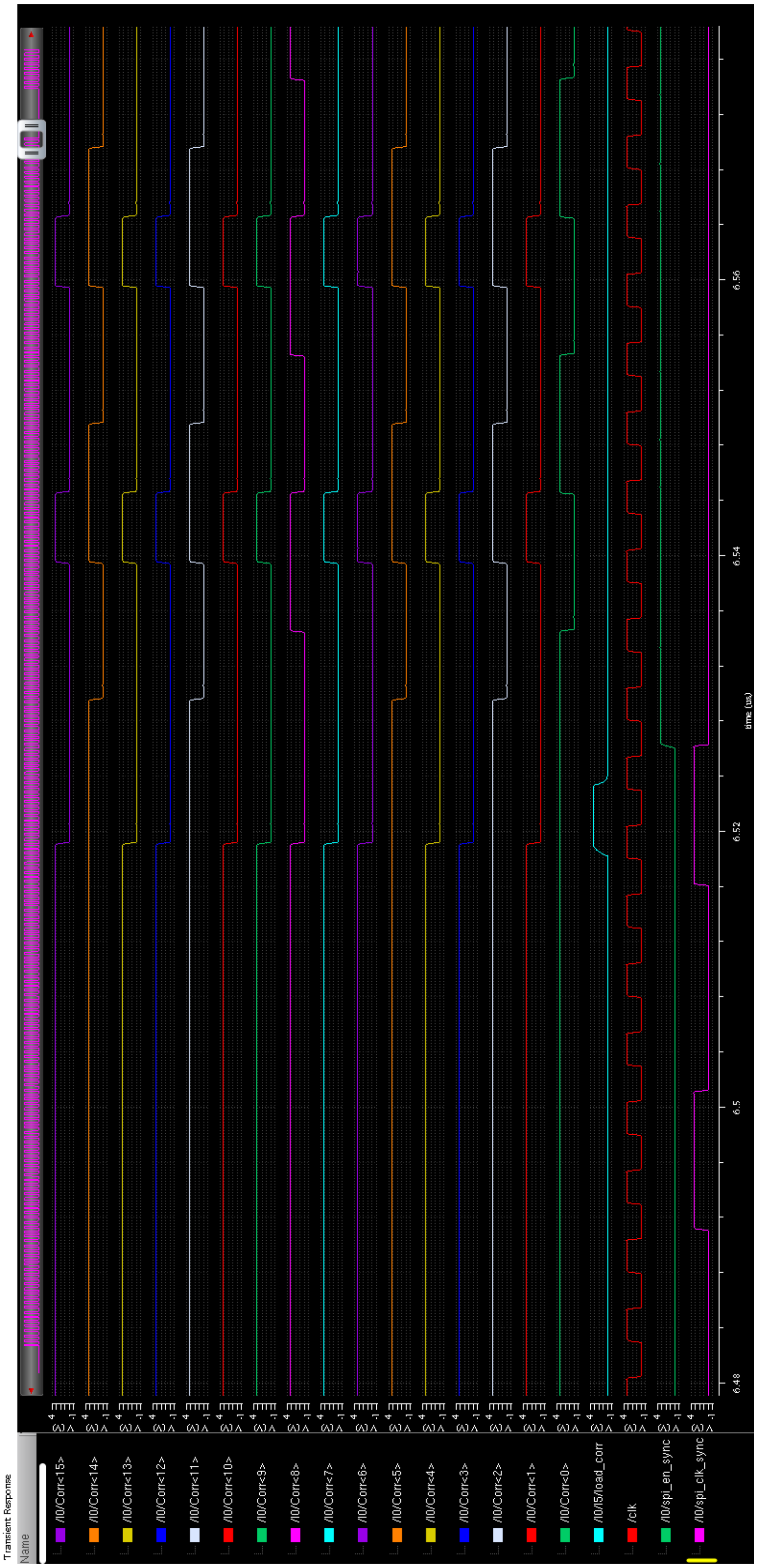

Figure 10 – PRBS registers

B.2 SPI Out

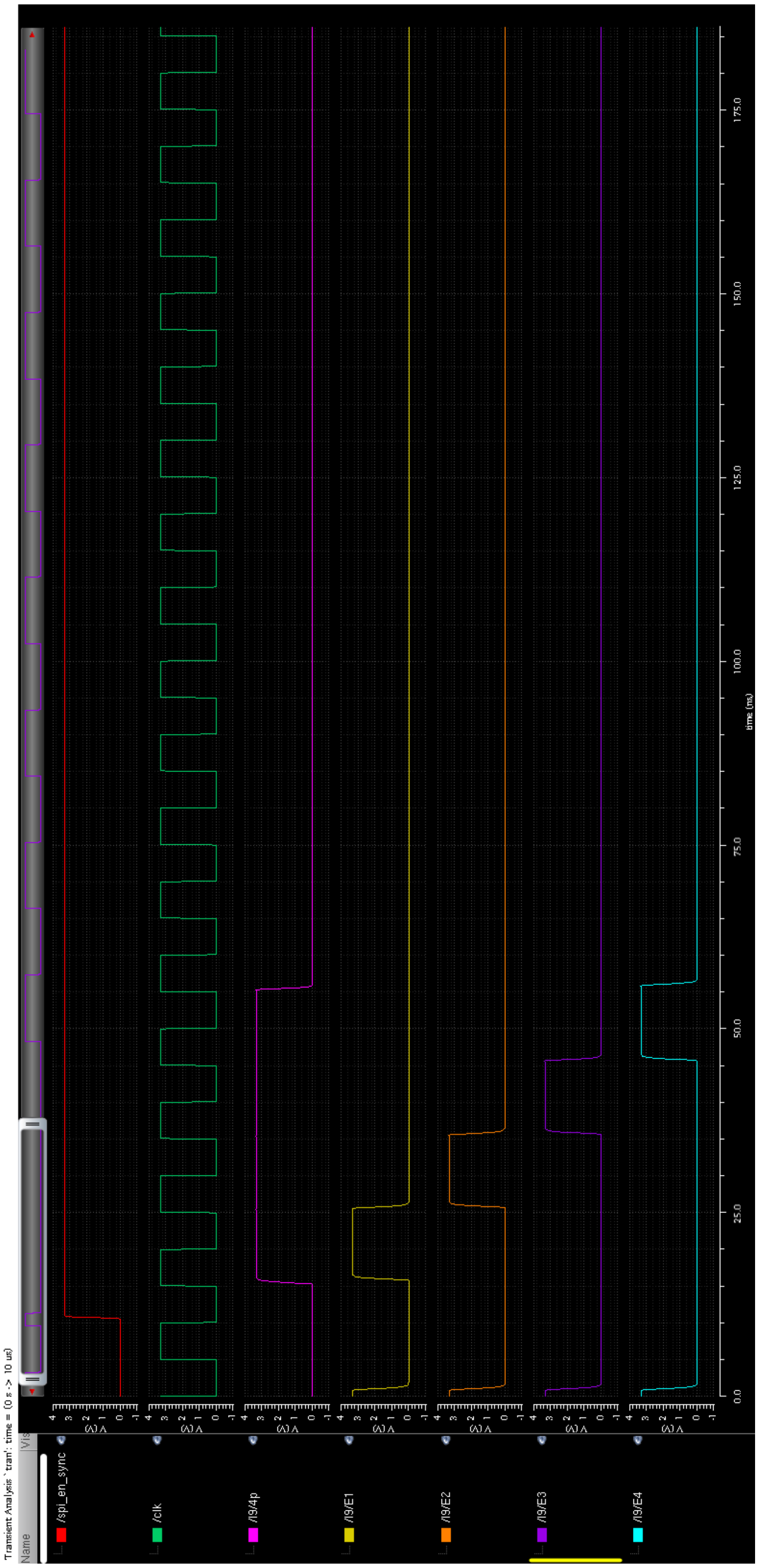

Figure 11 – Enable high

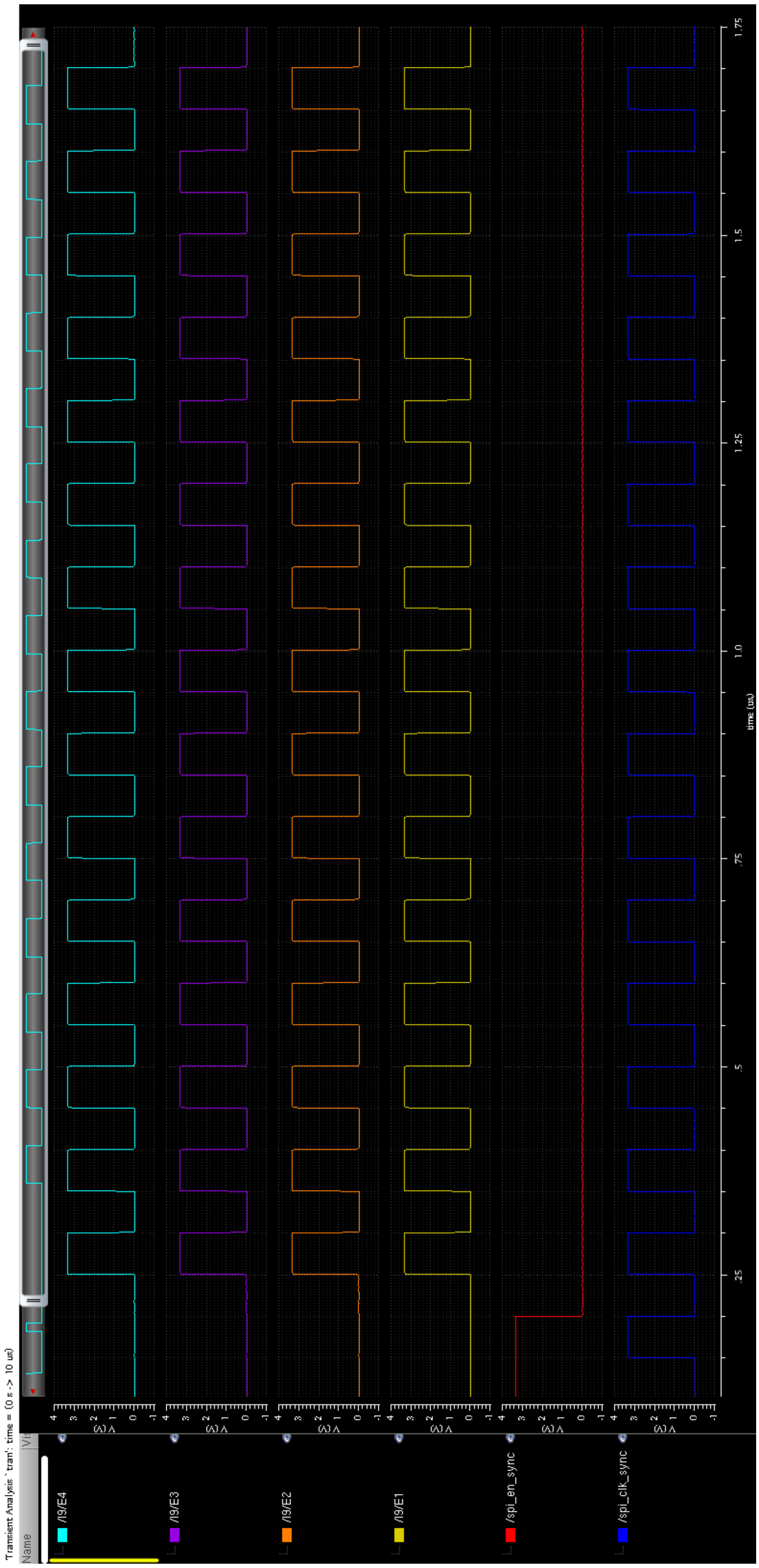

Figure 12 – Enable low

B.3 Adder

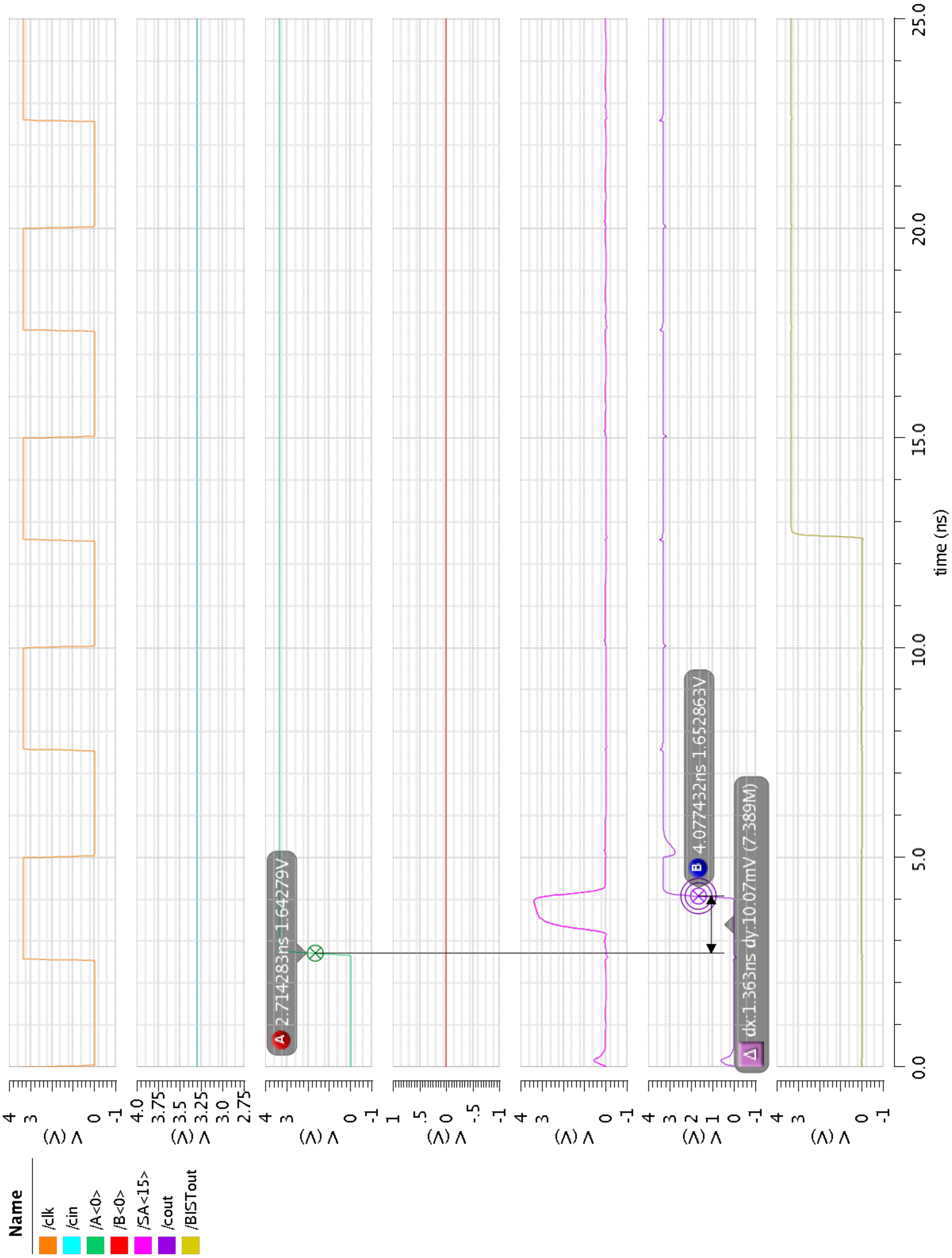

Figure 13 – Worst case propagation delay Kogge-Stone Adder