TSEK06
Transistor-Level Design Report

Group 5

Editor: Johannes Klasson

Version P1B

Status

Reviewed	Johannes Klasson	2016-03-3
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	2016-02-19	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 high level design phase. The meaning of high level is that the every basic logic gate is implemented in VerilogA. The main reson for doing this is to be able to simulate all logic to make sure that everything works as intended. Block level diagrams can be found in section 2, simulation results in section 3 and a small risk analysis in section 4. In appendix C and **??** a time plan of the next phase and a time report of this phase can be found.

2 Block Level Description

This section contains block level descriptions of all parts of the system seen in figure 1.

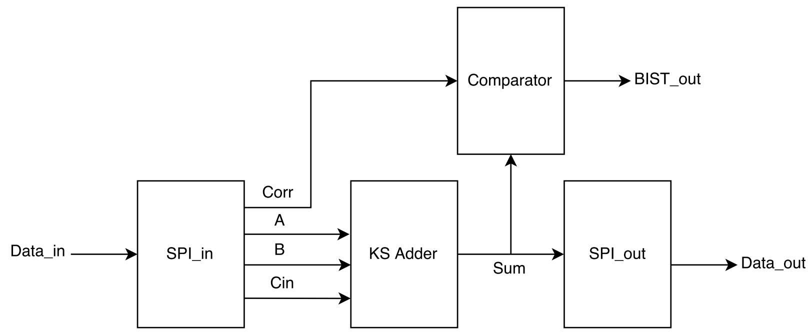


Figure 1 – Block diagram of the system.

2.1 SPI-in/PRBS

The SPI-in module consists of a lot of registers and some control logic moving data between these registers. Since the PRBS module share some registers with the SPI-in module, and the PRBS module is relatively small, we included it in the SPI-in module.
The first block in the SPI-in module is the SPI-receive block (see figure 2). It consists of 16 resettable D flip-flops (DFFSR), that is connected one after another. They are clocked on the SPI clock, and on each positive clock pulse, we get a new bit to shift in. After 16 pulses we have 16 bits stored, and a load signal is triggered so that each bit is moved to the correct PRBS-register.
The PRBS-registers (see figure 3) are register that consists of four DFFSR (one for each addition) and that can be run in two different modes. During the first mode, the normal mode, the registers are triggered with a load signal, and the data to the first DFFSR comes from the SPI-receive block. During the second mode, the PRBS mode, the registers and triggered with the system clock, and the data to the first DFFSR is the value of the third and forth DFFSR passed through a XOR. The mode is chosen by the SPI-enable signal. Of course, the correct sum does not need the PRBS mode, so it is only a normal register that only can run in normal mode.
The last block is the control block (see figure 4), which is the most complex block of the SPI-in module. The heart of the control block is a 6-bit counter. The first four bits signalizes if we have read a 16 bit word or not. The last two bits signalizes which of our four words we are currently reading. By combining these signals like in figure we are able to produce the load signals that trigger the PRBS-registers.

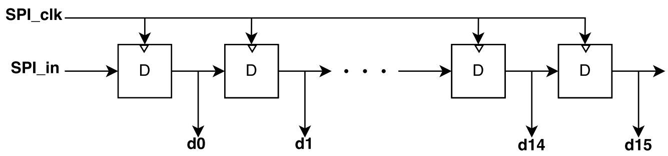


Figure 2 – Block diagram of the SPI-receive block.

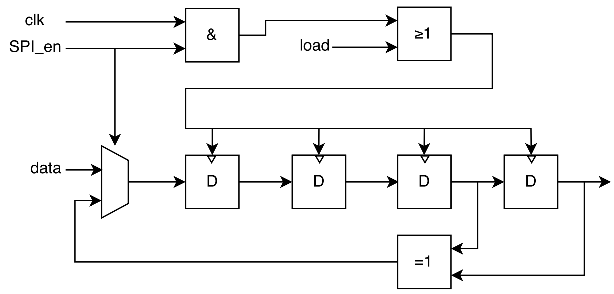


Figure 3 – Block diagram of the SPI-PRBS block.

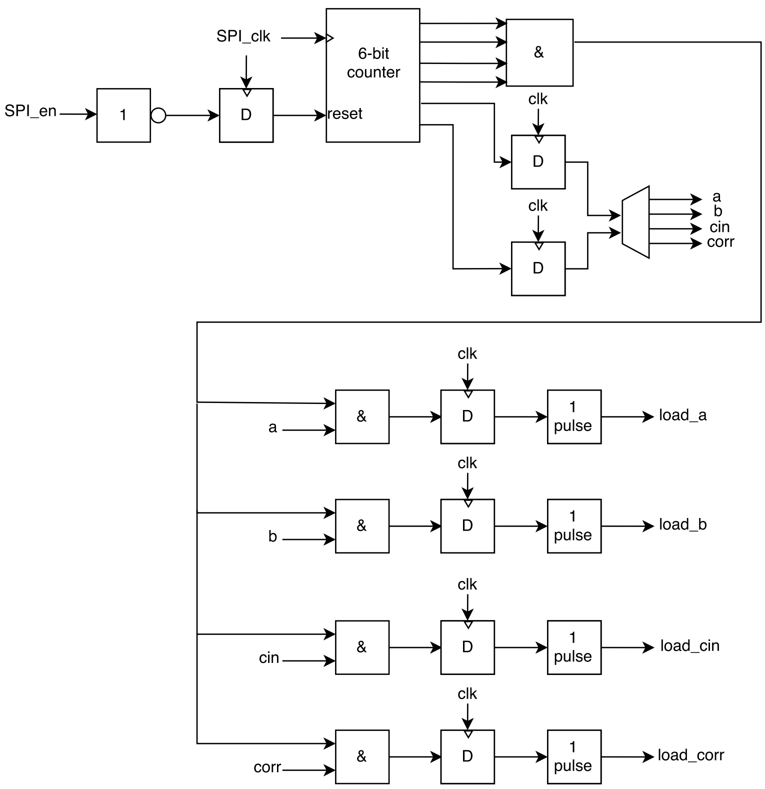


Figure 4 – Block diagram of the SPI-controller block.

2.2 16-bit Kogge-Stone Adder

The Kogge-Stone adder consists of four simple blocks connected in a complex way, as can be seen in A. These four blocks can be seen in figure 5-8. The red block constitute the initial stage which takes two binary numbers A and B as input. The corresponding truth table is found in table 2 in appendix B. The output signals P and G generated from this block are later used by other blocks in the adder. The G, also called the Generate signal, trickles down through the hierarchy of yellow, and yellow carry blocks to finally end up in the sum block. The truth table for this block can be found in table 5. Truth tables for the yellow and yellow carry blocks are found in table 3 and 4.

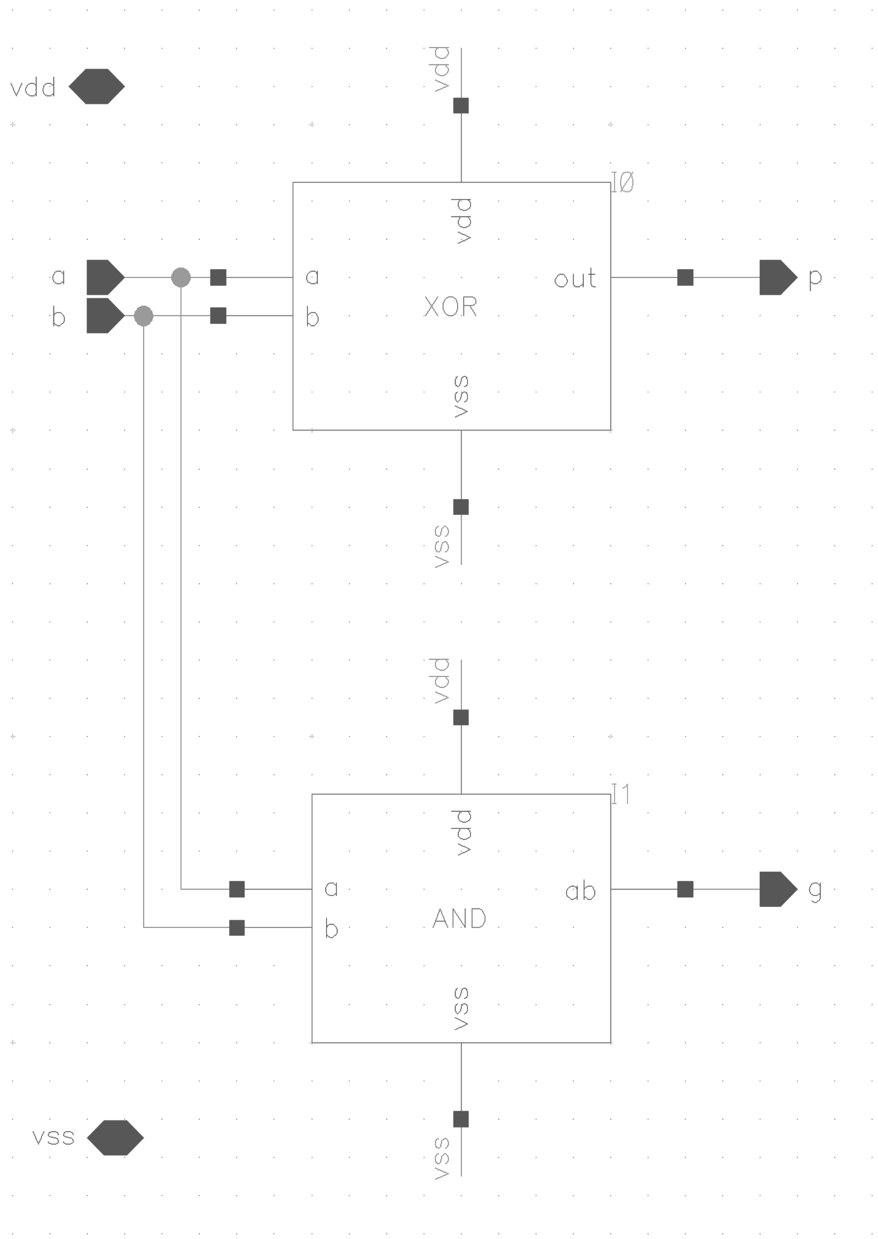


Figure 5 – Schematic view of the red block.

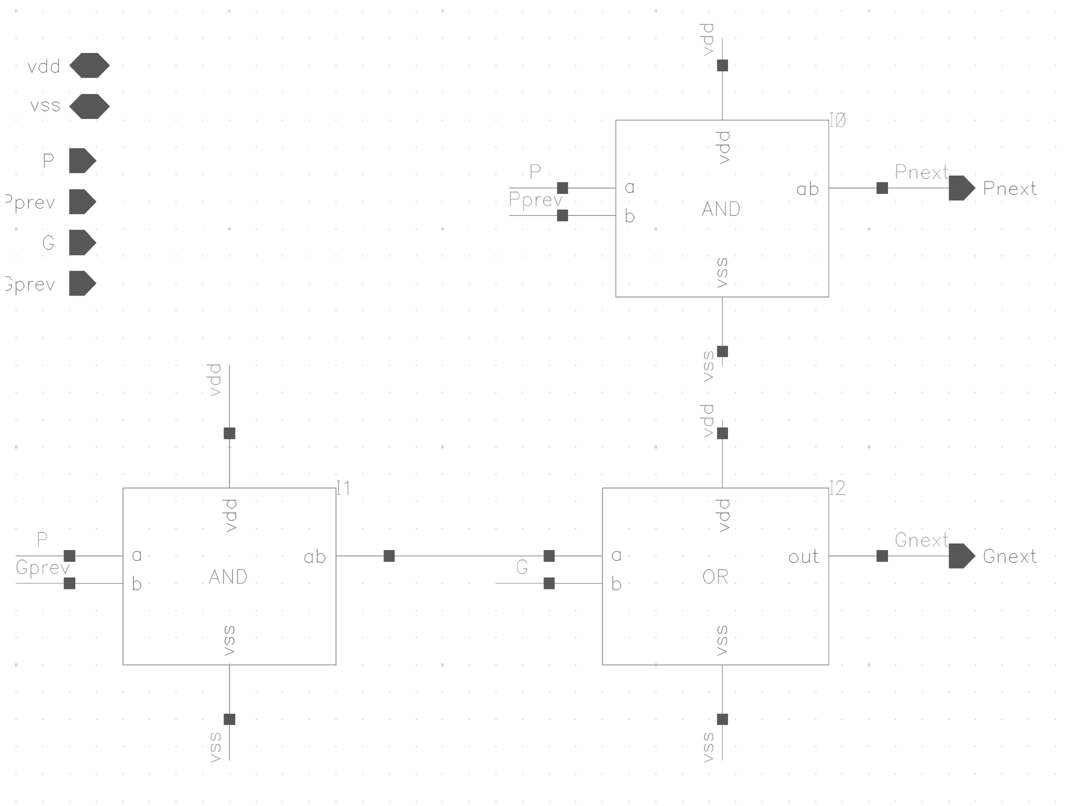


Figure 6 – Schematic view of the yellow block.

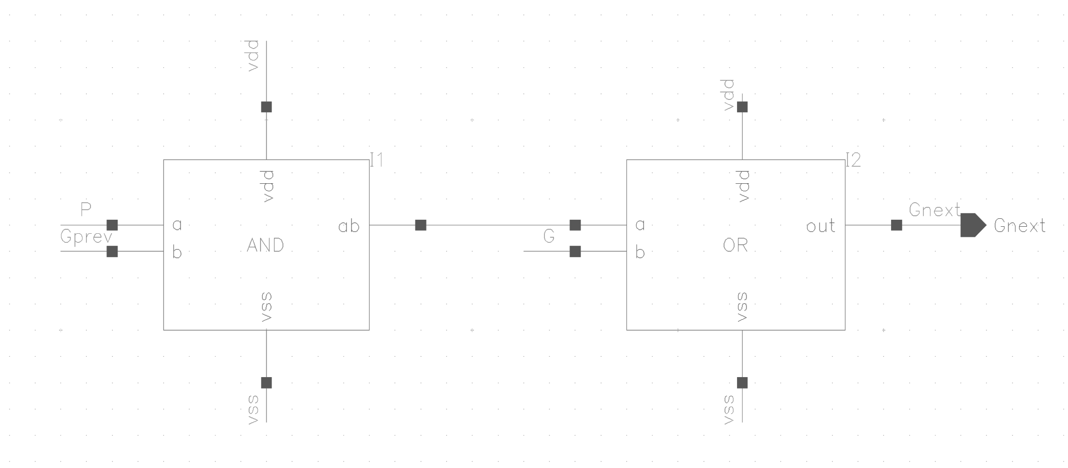


Figure 7 – Schematic view of the yellow carry block.

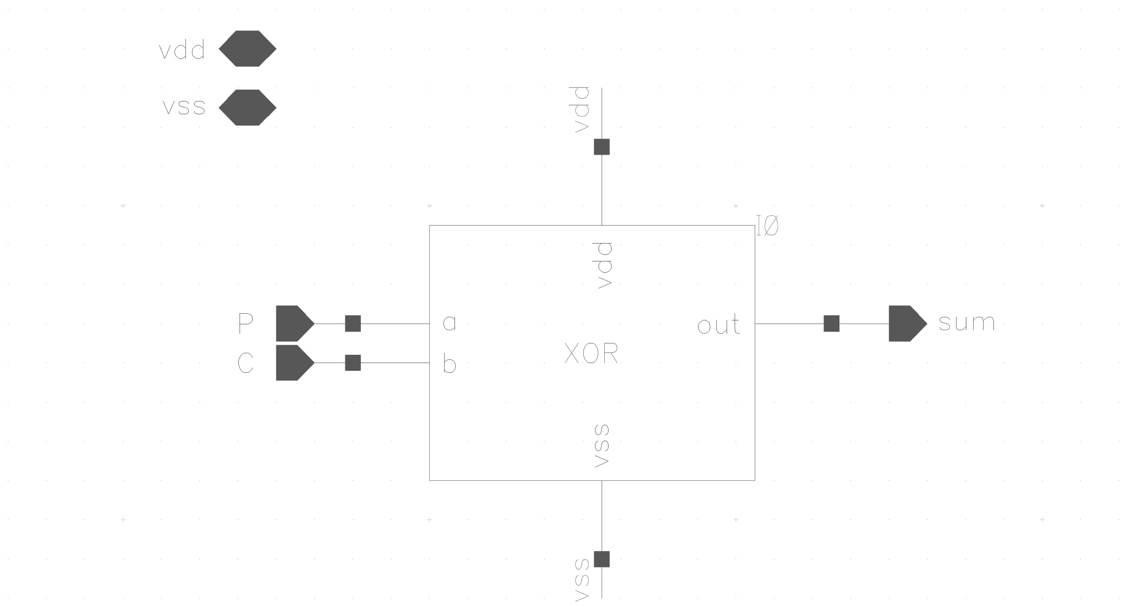


Figure 8 – Schematic view of the sum block.

2.3 Comparator

The comparator consists of 17 2-input XNOR gates where one bit of each number is fed into each gate. The output from the XNOR gates are fed into a couple of AND gates which generates the final output. The comparator is 17 bits wide since it compares two 16 bit numbers plus their carry bits. The logic table of the XNOR gates is shown in table 1.

Table 1 – Logic table of XNOR block.

A_i	B_i	$Y = \overline{(A_i \oplus B_i)}$
0	0	1
0	1	0
1	0	0
1	1	1

2.4 SPI out

This chapter will discuss the changes to the SPI output module.

2.4.1 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, just change the configuration files to use schematics instead of the verilog code.

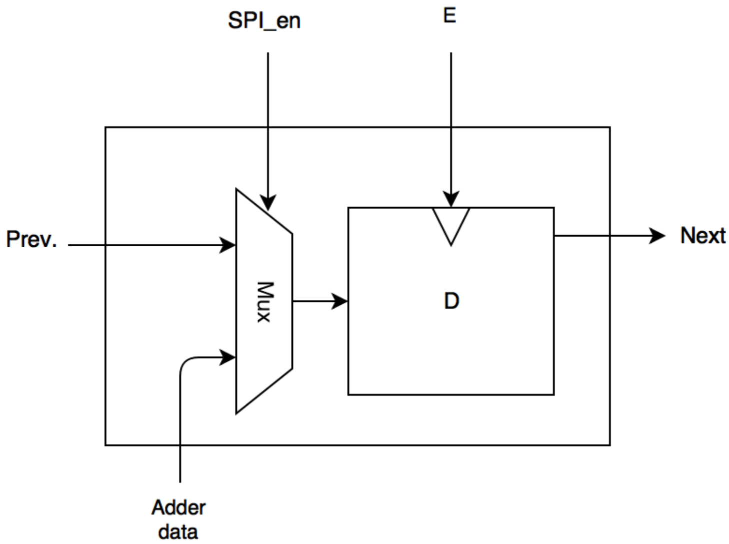


Figure 9 – Shift register cell

2.4.2 Control logic

As all verilog 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. 10.

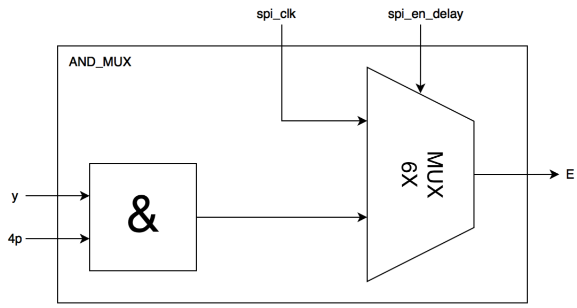


Figure 10 – Functional block containing an AND and a MUX.

By testing we found that a MUX that is approximately six times larger should be sufficient. In Fig. 11 the schematic of the sized multiplexer, implemented using NAND and an inverter, is shown.

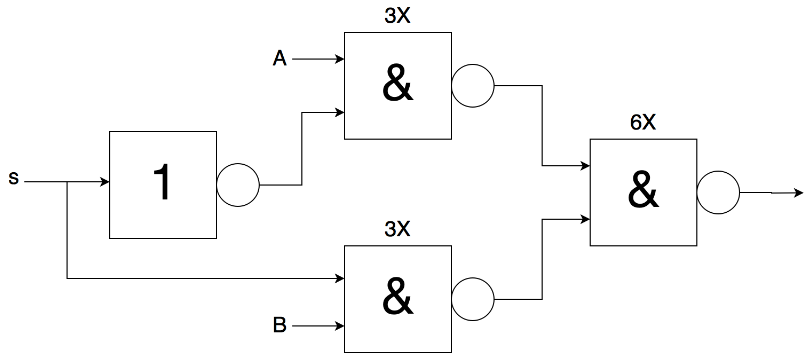


Figure 11 – Sized multiplexer.

2.4.3 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 as the group decided to keep the clock 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.

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 test_koggeadder and the input sequence is the same that were fed into the SPI. The most relevant part of the simulation is precisely after the topmost signal goes high. When this happens the data is already fed into the register in front of the adder and begins to shift into the adder on positive clock edge. The out, or to makes things more clear, the BISTout signal clearly shows that the two first addition yields the correct result but the thirds makes the same signal go low. This is a construction of the input from our side to test if the comparator can detect errors. After this the BISTout are low for a while before it goes high again which means that the fourth addition was successful.

3.3 Comparator

The simulation of the comparator is seen in test_corr. The same reasoning as in the section above applies here. The simulation is quite striped down due to readability but one can clearly see that out, which is BISTout, is high if and only if the corr-signals and sum signals match.

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_unbuffered a simulation of the behaviour when the SPI enable goes high. As can be seen, the four enable pulses are created correctly but they are very weak. This simulation shows the system before buffering the enable signals.
In the image spi_out_control_buffered a simulation of the system after buffering the signals can be seen.
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

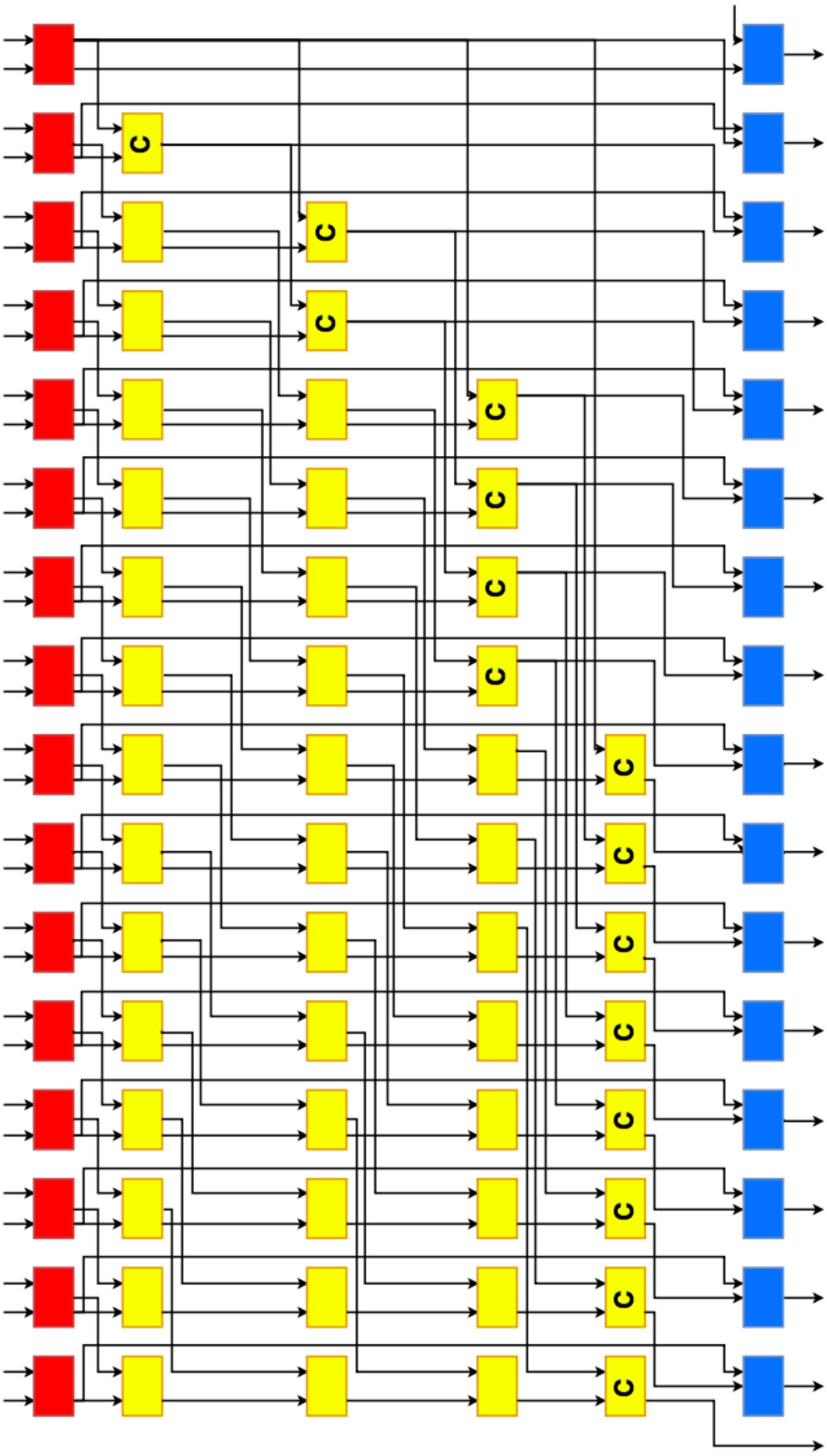
One can get a overall picture of the behaviour of the system by looking at at the simulations in the order spi_receive, spi_prbs, koggeadder, corr and last the spi_out. This is possible because all this simulations are part of a bigger one.

4 Risks and Delays

During the high level design phase there hasn't been any condiderable risks or 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 phases 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 Block diagram of the Kogge-Stone Adder



B Truth Tables for the Kogge-Stone Adder

Table 2 – Logic table of red block.

A_i	B_i	$P = A_i \oplus B_i$	$G = A_i \wedge B_i$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 3 – Logic table of yellow block.

G_i	$G_{i,prev}$	P_i	$P_{i,prev}$	$P = P_i \wedge P_{i,prev}$	$G = (P_i \wedge G_{i,prev}) \vee G_i$
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Table 4 – Logic table of yellow with carry block.

P_i	G_i	$G_{i,prev}$	$G = (P_i \wedge G_{i,prev}) \vee G_i$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 5 – Logic table of sum block.

P_i	C_{i-1}	$S_i = P_i \oplus C_{i-1}$
0	0	0
0	1	1
1	0	1
1	1	0

C Time Plan

Planning

Project: 16 bit Kogge-Stone ad

Project group: 5	**Date: 160204**	**Reviewed:**
Customer: Martin Nielsen-Lön	**Version: P1B**	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																	0,5
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															−8
7	Implement transistor level design of the SPI unit	30	JI, JK					15	15													0
8	Simulation and test of transistor design (SPI)	20	JI, JK							20												0
9	Implement layout level design of SPI unit	30											25	25								−2
10	Simulation and test of layout (SPI)	20													20							0
11	Define structure of the generator	10	AY		0																	10
12	Implement linear feedback shift registers in Verilog-A	10	AY			0																10
13	Integrate to high level design of the generator	10	AY			0																10
14	Simulation and test of the high level design (generator)	5	AY				0															5
15	Implement transistor level design of the generator unit	15	JI, JK					10	5													0
16	Simulation and test of the transistor design (generator)	10	JI, JK							10												0
17	Implement layout level design of generator unit	15											10	5								0
18	Simulation and test of layout (generator)	10													10							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,															16,
25	Implement transistor level design of the adder unit	40	JT, AY					20	20													0
26	Simulation and test of the transistor design (adder)	20	JT, AY						10	10												0
27	Implement layout level design of adder unit	40											10	13	15	15						−1
28	Simulation and test of layout (adder)	20															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					5	5													10
34	Simulation and test of the transistor design (comparator)	10	JT, AY							10												0
35	Implement layout level design of comparator unit	20											15	15								−1
36	Simulation and test of layout (comparator)	10													10							0
37	Off-chip hardware interface	30																15	15			0
38	Documentation and presentation	60		28			35		20	20									10	20		−7
39	Meetings	60			11	0	0	4	4	4			4	4	4	4	4	4	4	4	4	1
40	Buffer time	80						15	10	10			5	5	5	5	5	5	5	10	0	0
41	High level integration	15	JI, JH, JT, AY				14															1
42	Transistor level integration	10								10												0
43	Layout level integration	15														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																	10
47	Implementation of test bench for comparator	5	JT		0																	5
48	Implementation of test bench for the complete system	20					15															5
	Sum, number of hours	755			30,5	34	99	69	89	94	0	0	69	67	64	54	29	24	34	34	4	0