TSEK06
High-Level Design Report

Group 5

Editor: Johannes Klasson

Version P1B

Status

Reviewed	Johannes Klasson	2016-02-15
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 Tarasso	Designer	070-5738583	jonta760@student.liu.se
Alexander Yngve	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	2016-02-15	First draft	Johan Isaksson

1 Introduction

This document describes the state of the project Kogge-Stone adder in the course TSEK06 by group 5 after finishing the high level design phase. Block level diagrams can be found in section 2, simulation results in section 3 and a small risk analysis in section 4. In appendix A and B 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.

Bild på top level

Figure 1 – Block diagram of the system.

2.1 SPI_in/PSBR

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_recieve block. 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 got 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 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_recieve 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 trhough a XOR. The mode is choosen by the SPI_enable signal.

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 2. These four blocks can be seen in figure 3-6. 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 C. 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.

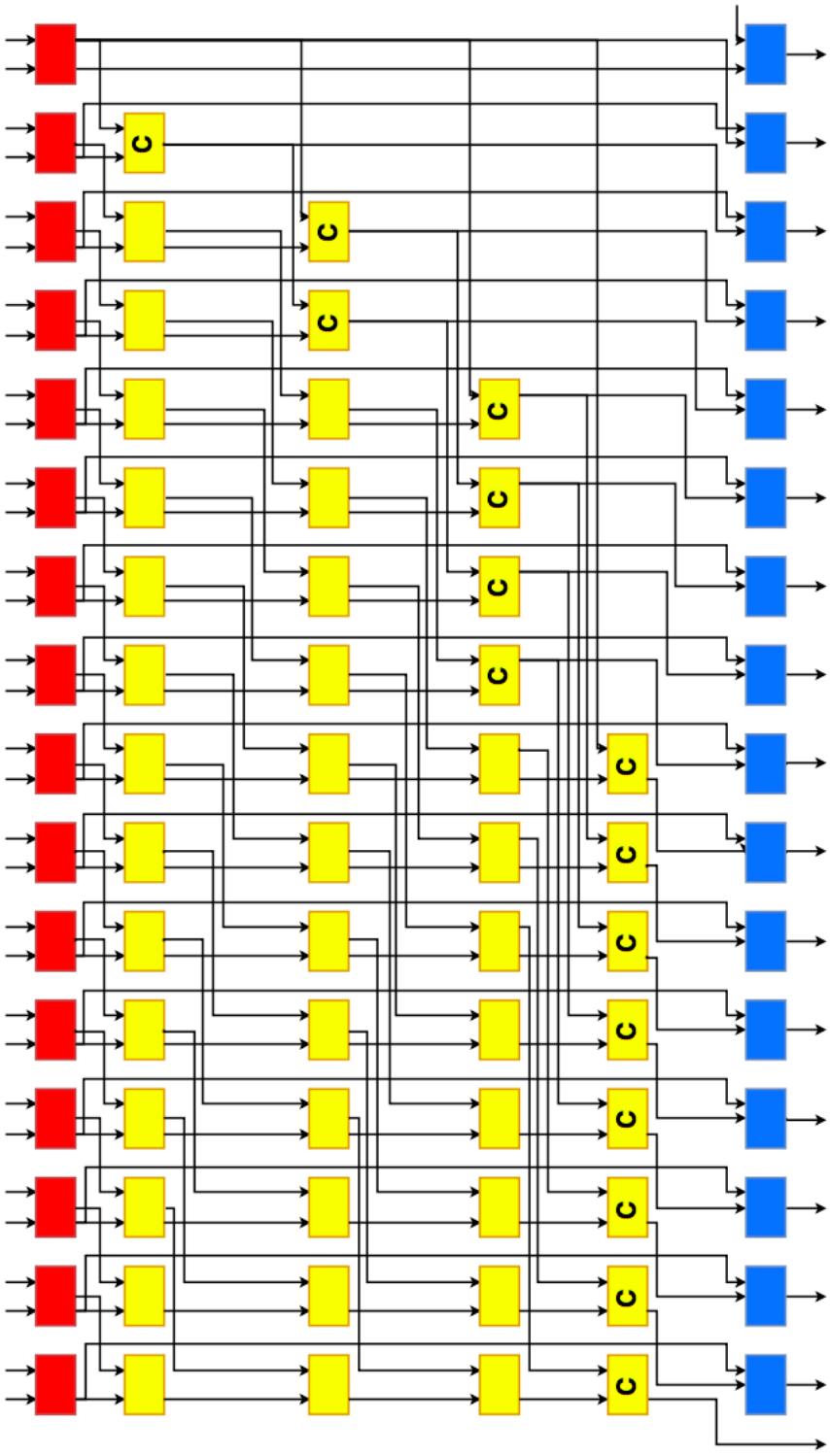

Figure 2 – Block diagram of the Kogge-Stone Adder.

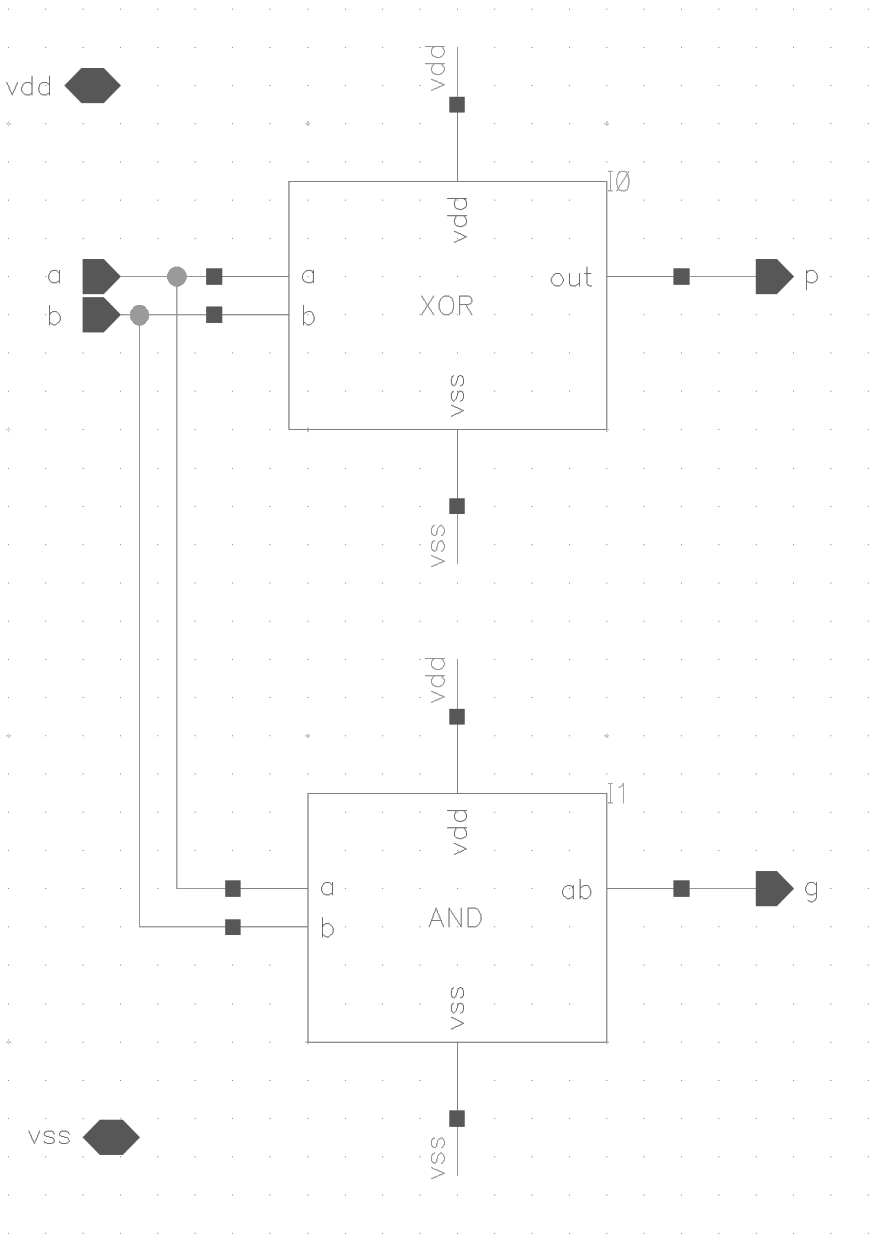

Figure 3 – Schematic view of the red block.

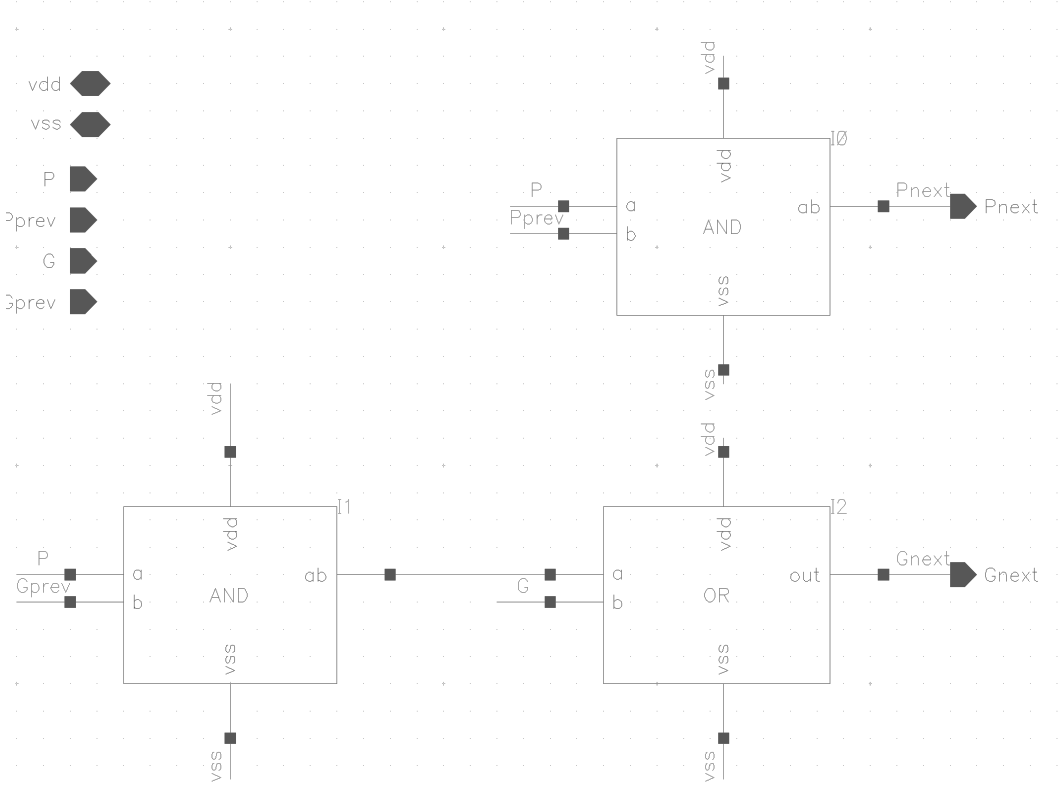

Figure 4 – Schematic view of the yellow block.

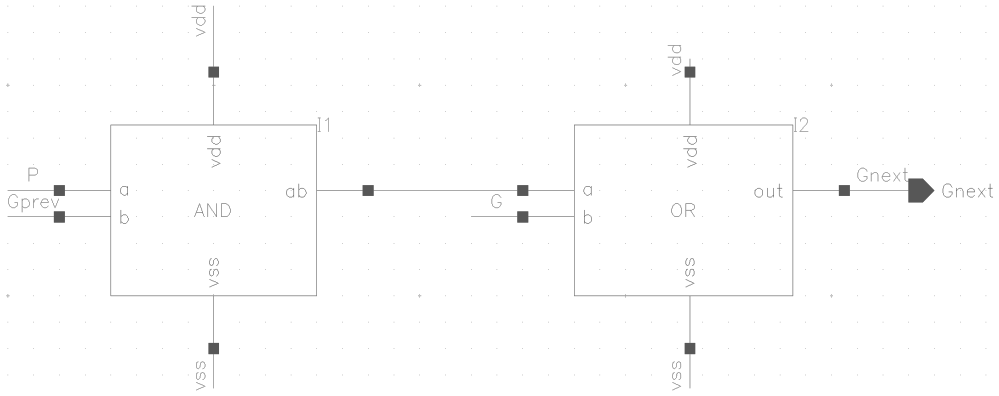

Figure 5 – Schematic view of the yellow carry block.

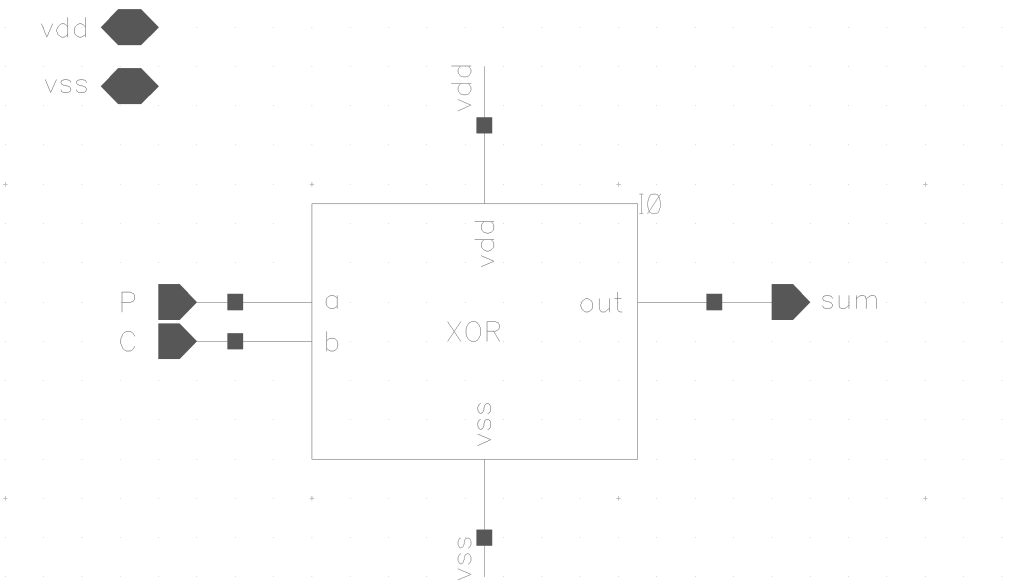

Figure 6 – Schematic view of the sum block.

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 module is responsible for correct output of the data from the adder.

2.4.1 Shift register

The output consist of four sums, each containing 16 sum bits plus one carry output bit. This makes a total of 68 bits. To implement this we use a 68 bit shift register where each cell in the register contains one D flip-flop and one multiplexer. The multiplexer is used to switch between shifting and parallel load of the D flip-flop. In figure 7 the schematic of a cell can be seen.

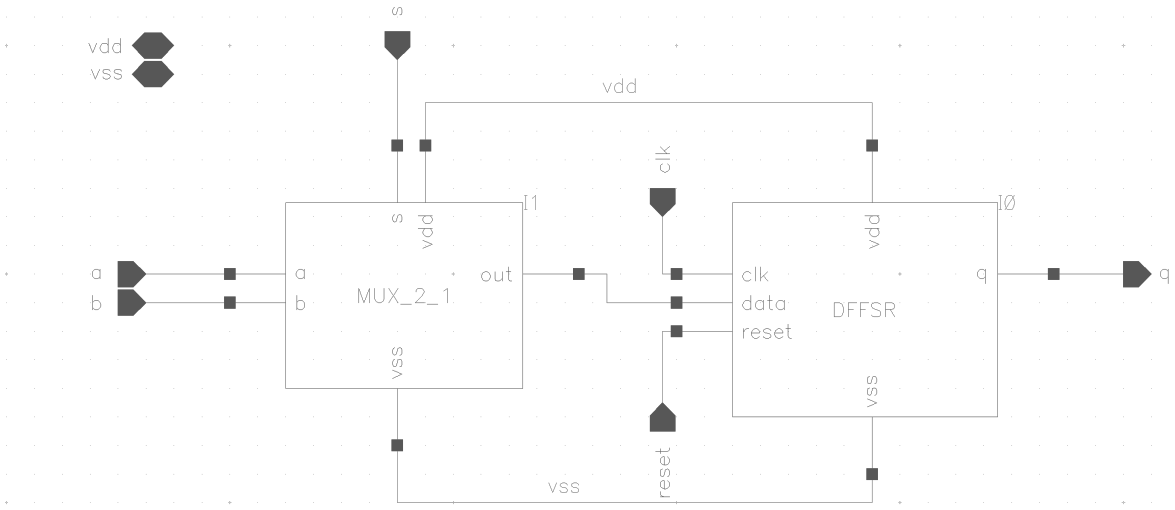

Figure 7 – Shift register cell

2.4.2 Control logic

The control logic is needed to load the long shift register with the correct data.
As the adder is built in such a way that it executes all the four additions as fast as it can after the spi enable signal goes high, the output from one addition will only be available for one system clock cycle. This means that we need to quickly grab the data and put it in the correct place.
Our implementation uses a pulse generator, a 2-bit counter, a decoder to select what 17-bit part of the shift register to load, and some multiplexers to select if to clock the register on the spi clock or on the enable signals from the decoder. The pulse generator is triggered by the spi enable signal and creates a pulse that is four system clock cycles long. This to only allow the counter to increment four times, and also to clip the pulse from the last decoder output. As can be seen in 8 the last output of the decoder is also feeded back to the enable of the counter through an inverter. This prevents a possible glitch that is generated when the counter starts over. The glitch would reload the section of the shift register containing the first sum.

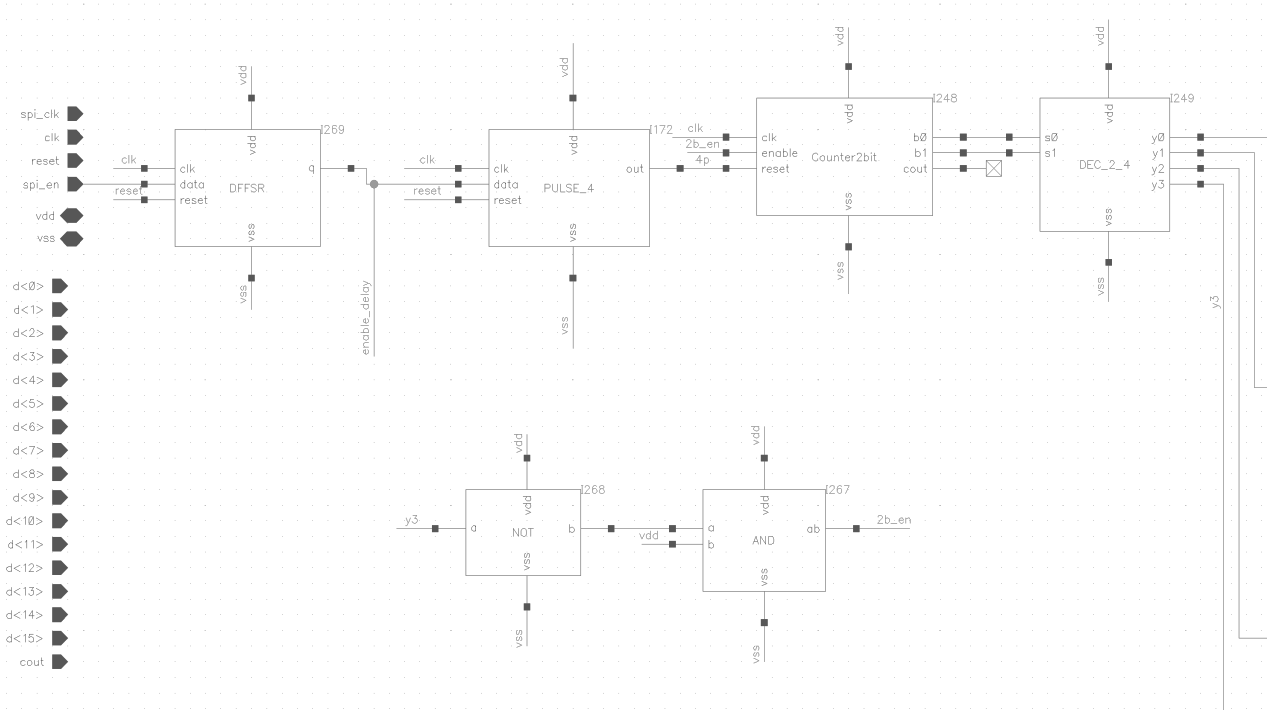

Figure 8 – Control logic for spi output, first half

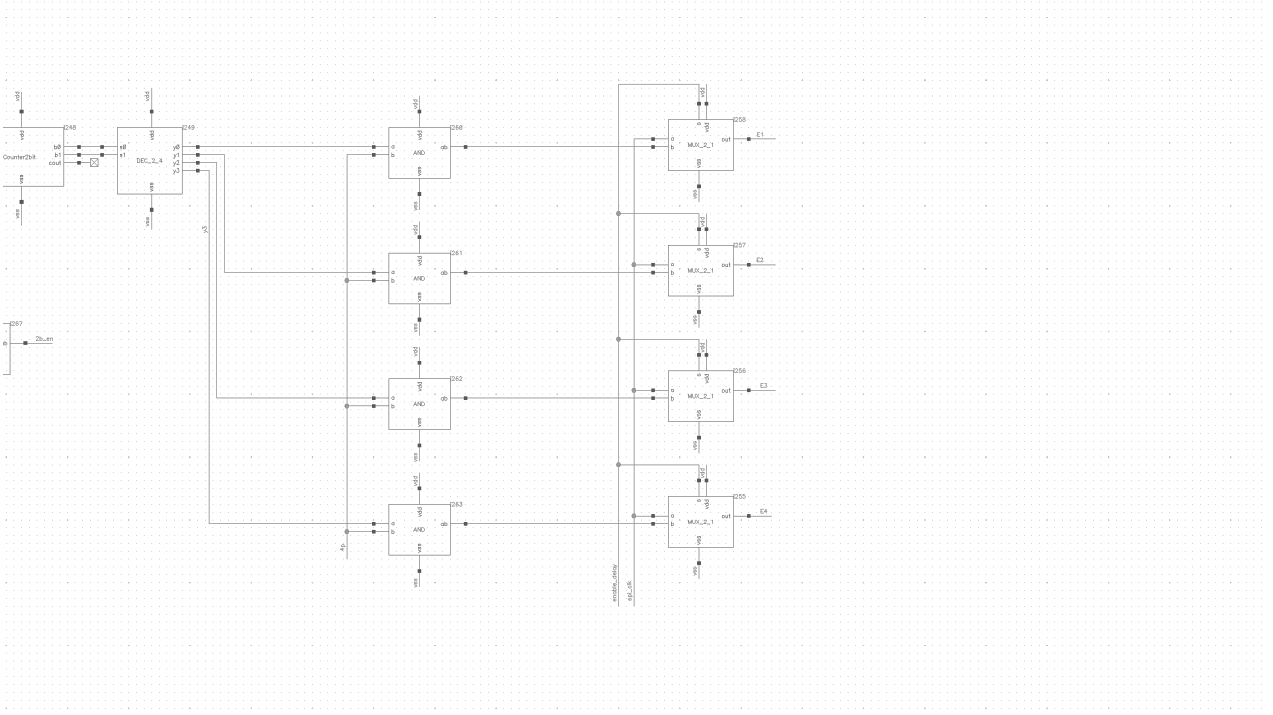

Figure 9 – Control logic for spi output, second half

2.4.3 Protocol

This unit outputs the data with most significant bit first, in this case the carry output. To not use

3 Simulation Results

3.1 SPI In

HANKEN

3.1.1 Recieve

3.1.2 Hanken 2

3.1.3 Hanken 3

3.2 Kogge-Stone Adder

JONAS

3.3 Comparator

JONAS

3.4 SPI Out

HANKEN

3.5 Top Level

JONAS

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 Time Plan

JOHAN

B Time Report

JOHAN

C 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