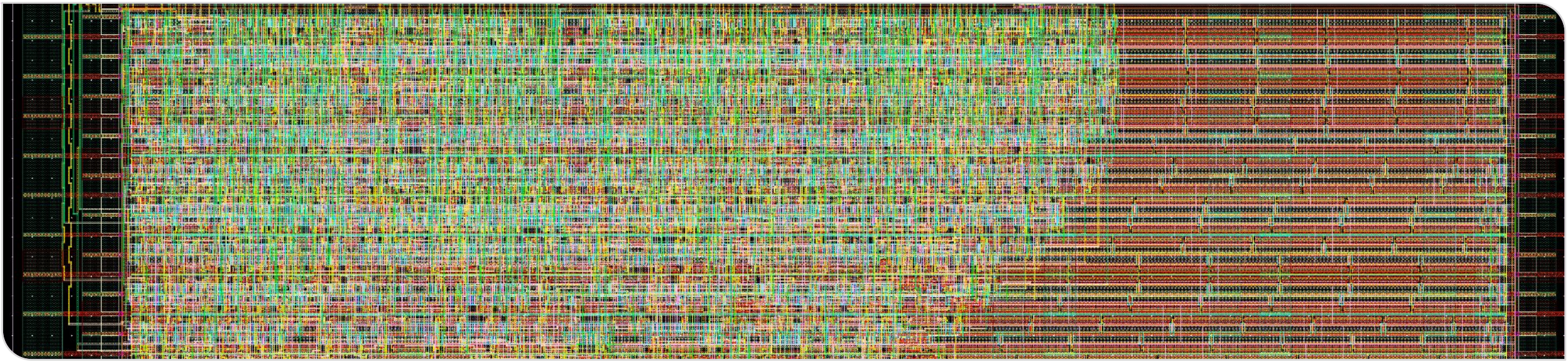
Karlsruhe Institute of Technology

Praktikum System-on-Chip

Final presentation of Group 6

Marius Haschka, Yang Hu, Manuel Philippin, Kishore Mangal Singh



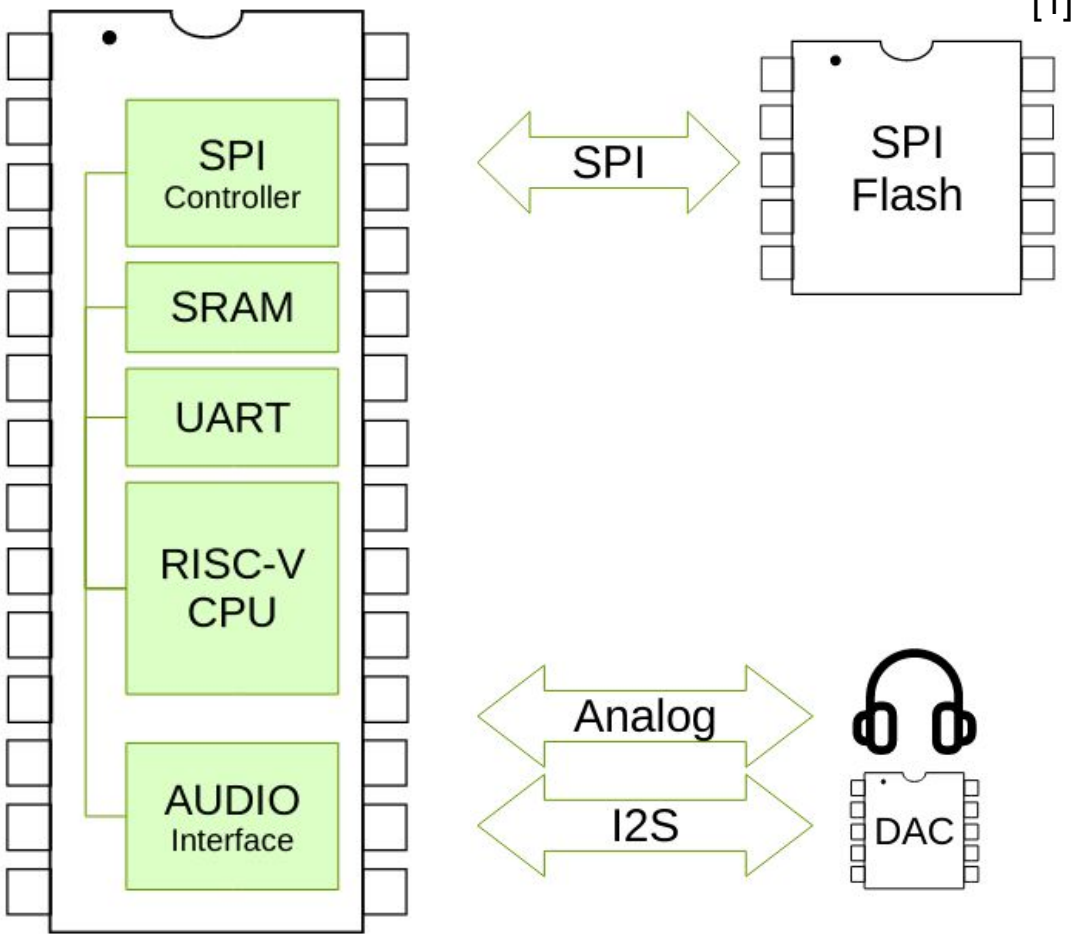
General Overview

Development of an audio player split into an **analog** and a **digital** part:

Analog and Mixed Signal design:
DAC + Amplifier + ASIC

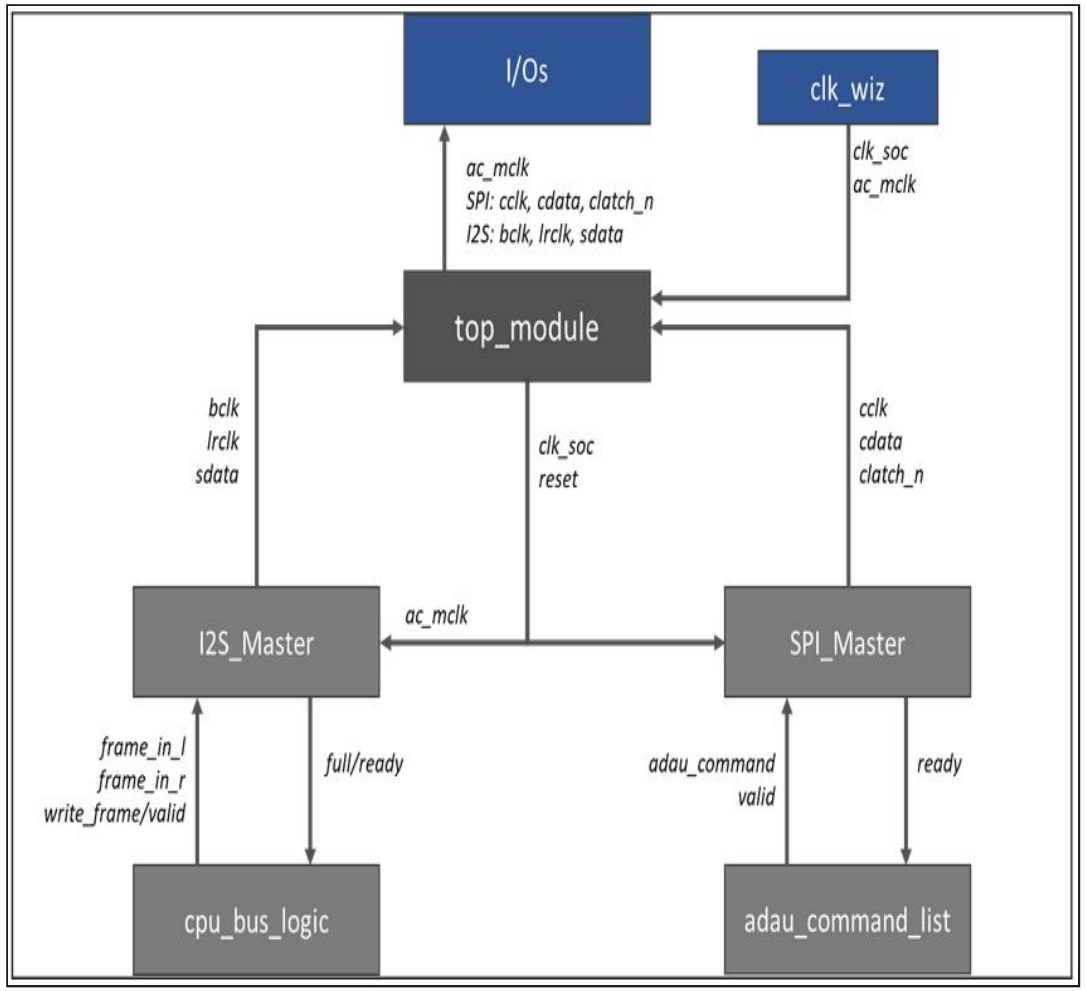
Digital Design
System Bus + FPGA Development

[1]



Source:
[1] PSOC, Slides 1 - Introduction, WS2122

FPGA Development Overview



- The prototyping system includes that runs on FPGA basically consists of two modules.
- Master functionality for the SPI (used for the control of the DAC).
- The I2S (used for transmitting the data) bus.
- adau_command_list: controls to be sent to DAC are configured.
- Sine_generator: used for simple testing of audio data. This module is then replaced with the RISC-V processor, which provides the real audio data.

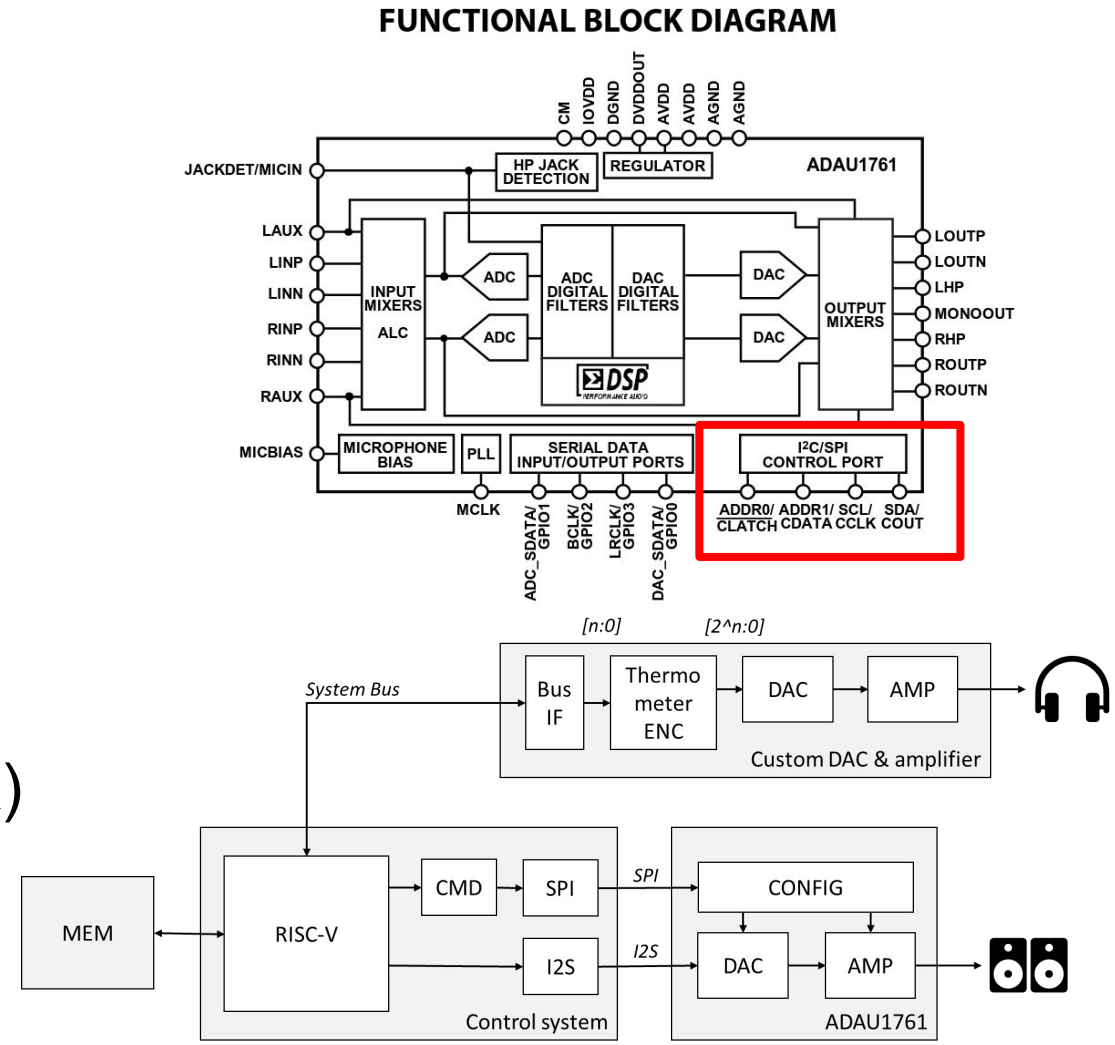
Source: [1]

FPGA Development: SPI Master

- Control Port slave mode on ADAU side
- Receive Configuration Data from command list
- Transfer command to ADAU

- 4 Inputs reset, clk, data_in, command_valid
- 4 Outputs ¬clatch, cclk, cdata, spi_ready

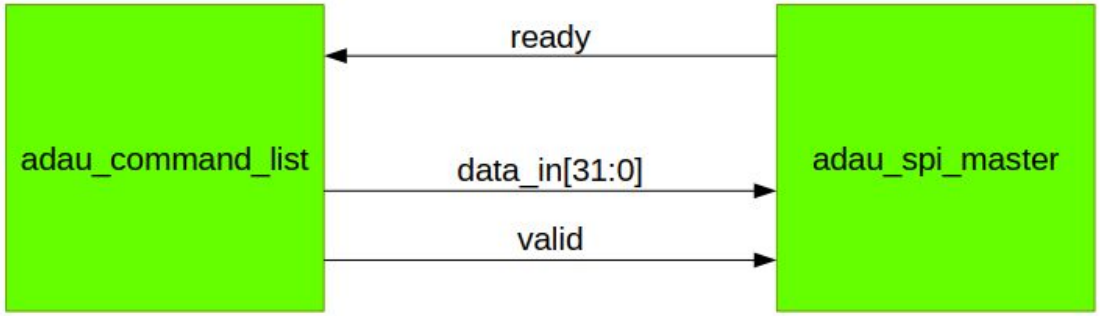
- During data transmission
  - cclk 5 MHz (Max. 10 MHz)
  - 120 MHz (clk_soc) / 24 = 5 MHz (cclk)
  - ¬CLATCH set low
  - MSB first

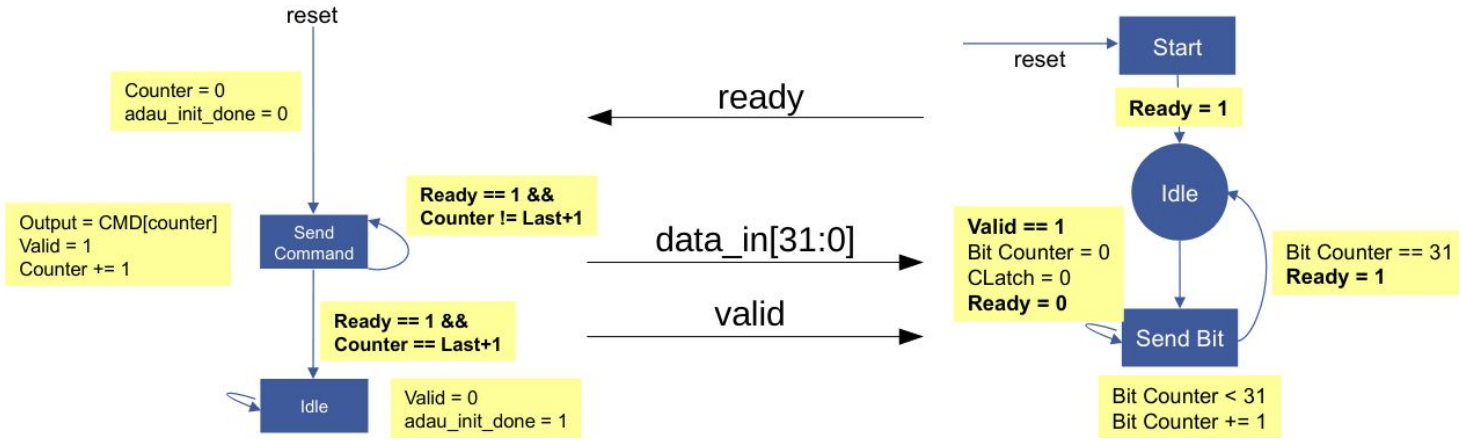


Source: [ADAU1761 Datasheet, Analog Devices]

FPGA Development: SPI Command List

- Command list give the ADAU the configuration data through SPI
- Commands sent through “hand-shake” with the SPI Master





8-Bit Header (0x00), 16-Bit Address, 8-Bit Data

Register (Address)	Value
SPI on (0) [3x]	0x00
R0: Clock Control (0x4000)	0x01
R15: Serial Port 0 (0x4015)	0x00
R16: Serial Port 1 (0x4016)	0x40
R22: Playback Mixer Left 0 (0x401C)	0x21
R24: Play Mixer Right 0 (0x401E)	0x41

Source: [PSoC Slide 3, P19, P21]

FPGA: Constraint & Debugging

Constraint File:

- Correct physical pins and signal connection
- "# Note that the bank voltage for IO Bank 13 is fixed to 3.3V on ZedBoard."
- *set_property IOSTANDARD LVCMOS33 [get_ports {signal}];*

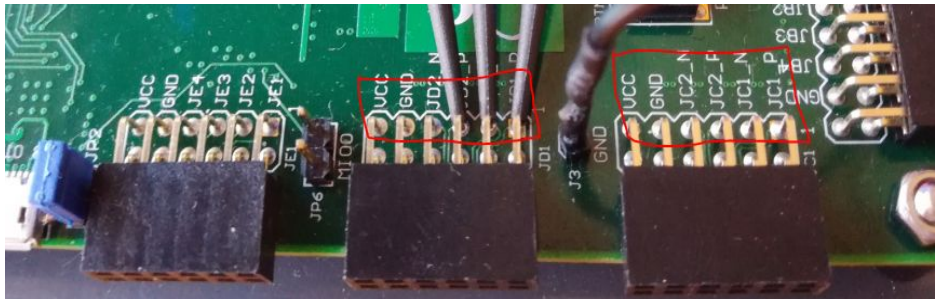
Select the physical pins

- Find the outer pin for better grip from logic analyzer
- Assign the digital pins to the selected physical pins

Table 9 - CODEC Connections

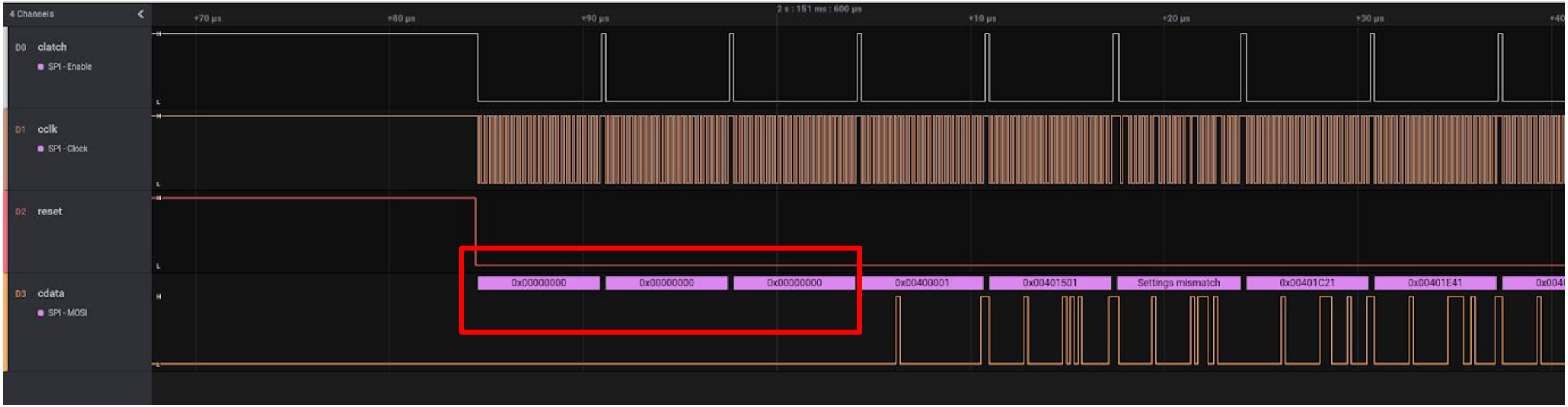
Signal Name	Description	Zynq pin
AC-ADR0	I2C Address Bit 0/SPI Latch Signal	AB1
AC-ADR1	I2C Address Bit 1/SPI Data Input	Y5
AC-MCLK	Master Clock Input	AB2
AC-GPIO2	Digital Audio Bit Clock Input/Output	AA6
AC-GPIO3	Digital Audio Left-Right Clock Input/Output	Y6
AC-GPIO0	Digital Audio Serial-Data DAC Input	Y8
AC-GPIO1	Digital Audio Serial Data ADC Output	AA7
AC-SDA	I2C Serial Data interface	AB5
AC-SCK	I2C Serial Data interface	AB4

https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zedboard/



Manuel Philippin, Marius Haschka, Kishore Mangal Singh, Yang Hu
Source:
[PSoC Lab Exercises 3]

FPGA: Constraint & Debugging



- Use logic analyzer to read the signal and datas

Manuel Philippin, Marius Haschka, Kishore Mangal Singh, Yang Hu
Source:
[PSoC Lab Exercises 3]

FPGA: I2S Master

I2S (“Inter-IC Sound”): unidirectional interface for audio devices

CLK = Bit clock (“bclk”)

WS = Word select, used for left and right channel (“lrclk”)

SD = Serial data (“sdata”)

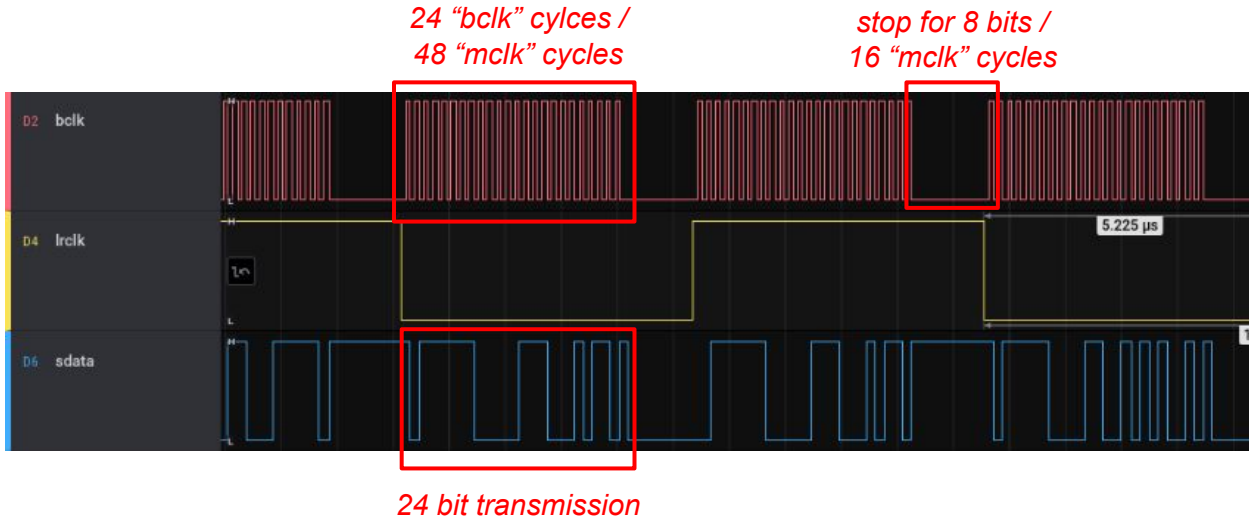
System requirements:

Audio sampled with 48 kHz from the DAC

24 bit data per channel

Utilisation of MCLK (12,288 MHz)

→ Statemachine is controlled through *clk_counter* and *bit_counter*



clk_counter

	63 -> 17	bit_counter 23 -> 0
	16 -> 3	bclk <= 0, bit_counter <= 23
Right channel	2	rd_en <= 1
Right channel	1	read data from FIFO, set data_av_flag
Right channel	0	toggle lrclk

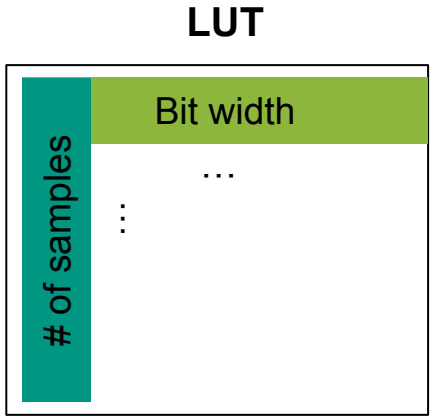
FPGA: Sine generator

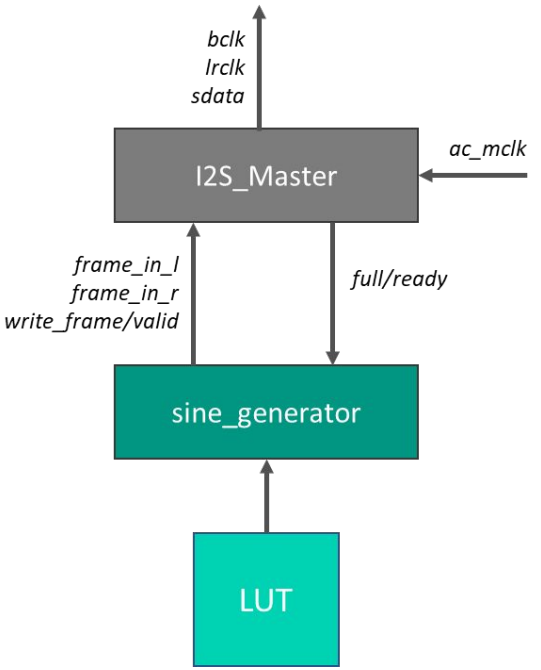
- A simple sine wave can be represented by a look up table (LUT)

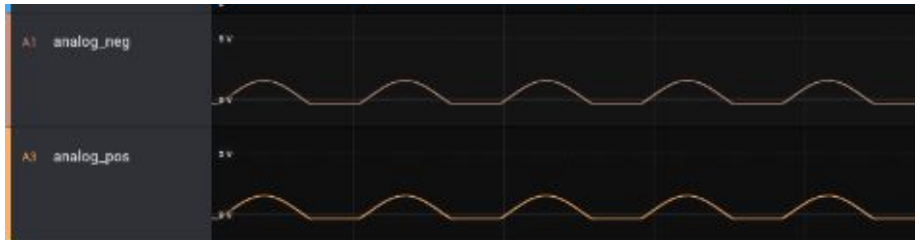
→ specified by (bit-) width and height (# of samples)

- To test the system a LUT is used that is read by the sine generator

- Reading out the LUT sample by sample through a counter





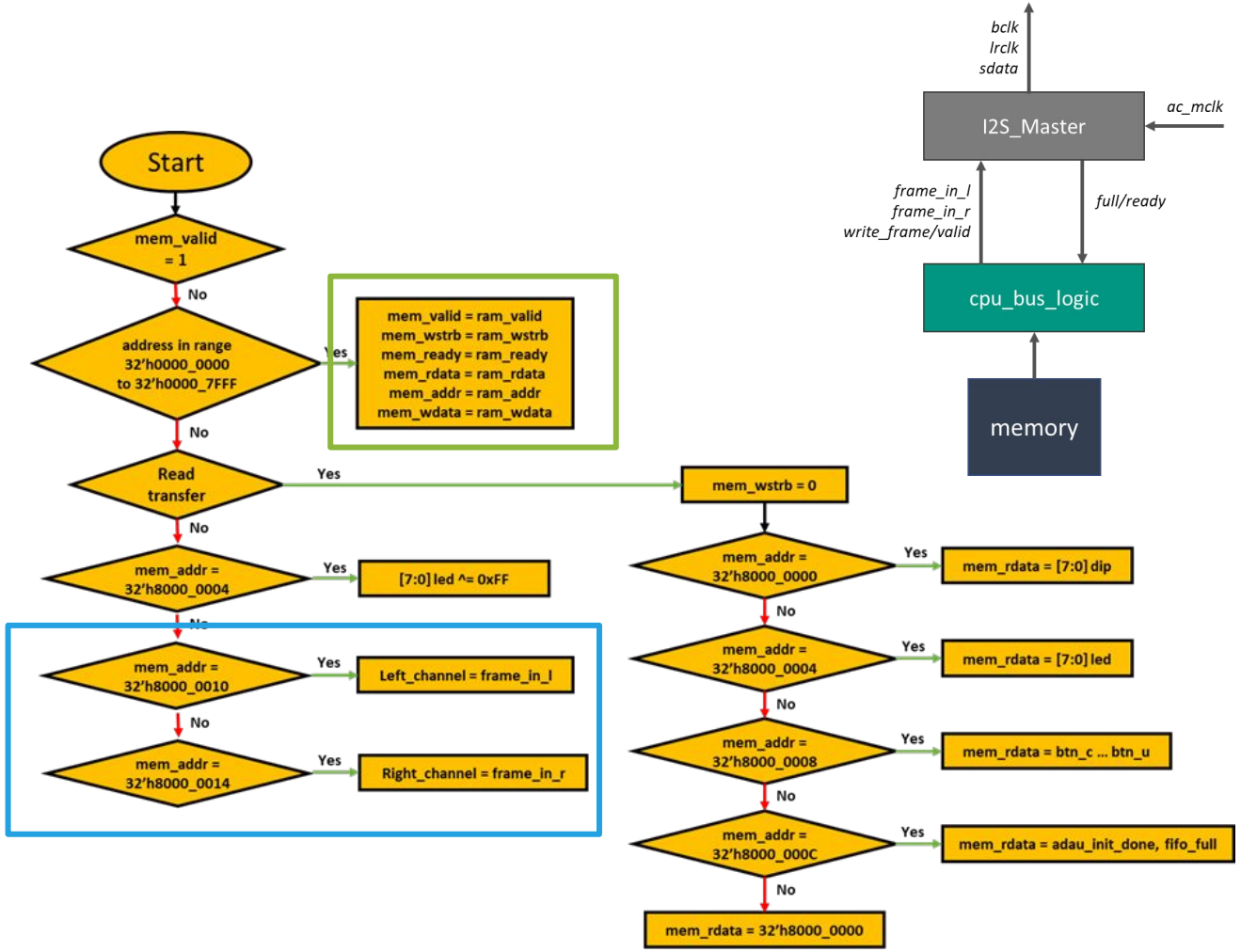


FPGA: RISC-V Core

Playing arbitrary audio data is realized by a RISC-V *PicoRV32* processor

cpu_bus_logic handles fetching data from the RAM and saving it into the FIFO

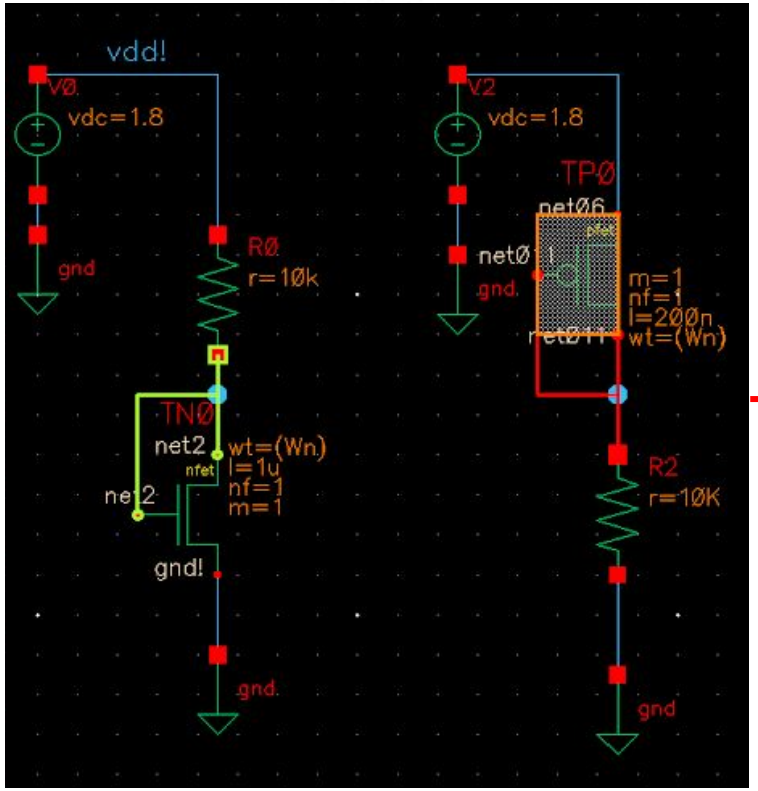
There are additional features for debugging and checking the ADAU status

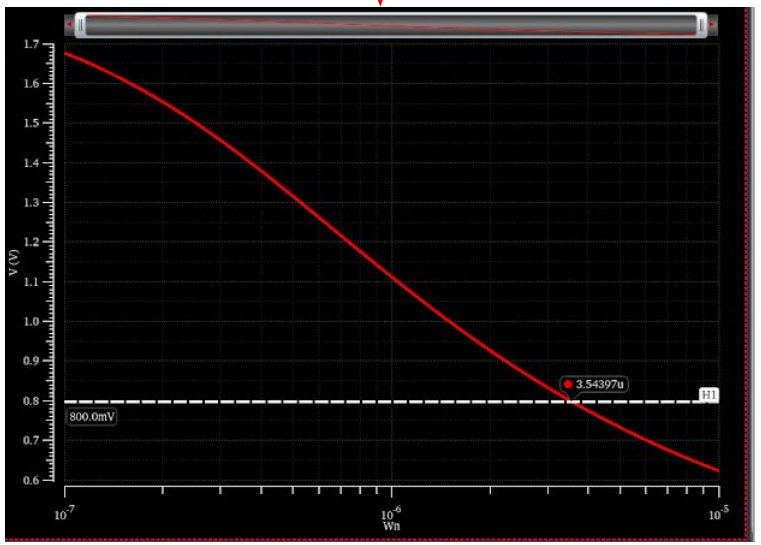


ASIC Design: Two stage differential Amplifier (1)

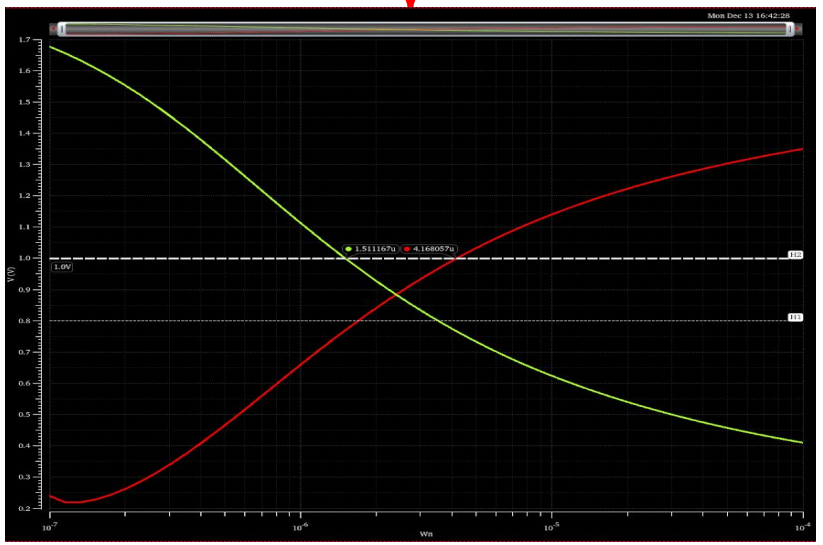
Karlsruhe Institute of Technology

Width of NMOS/PMOS Transistors:





NMOS width: 3.54 um

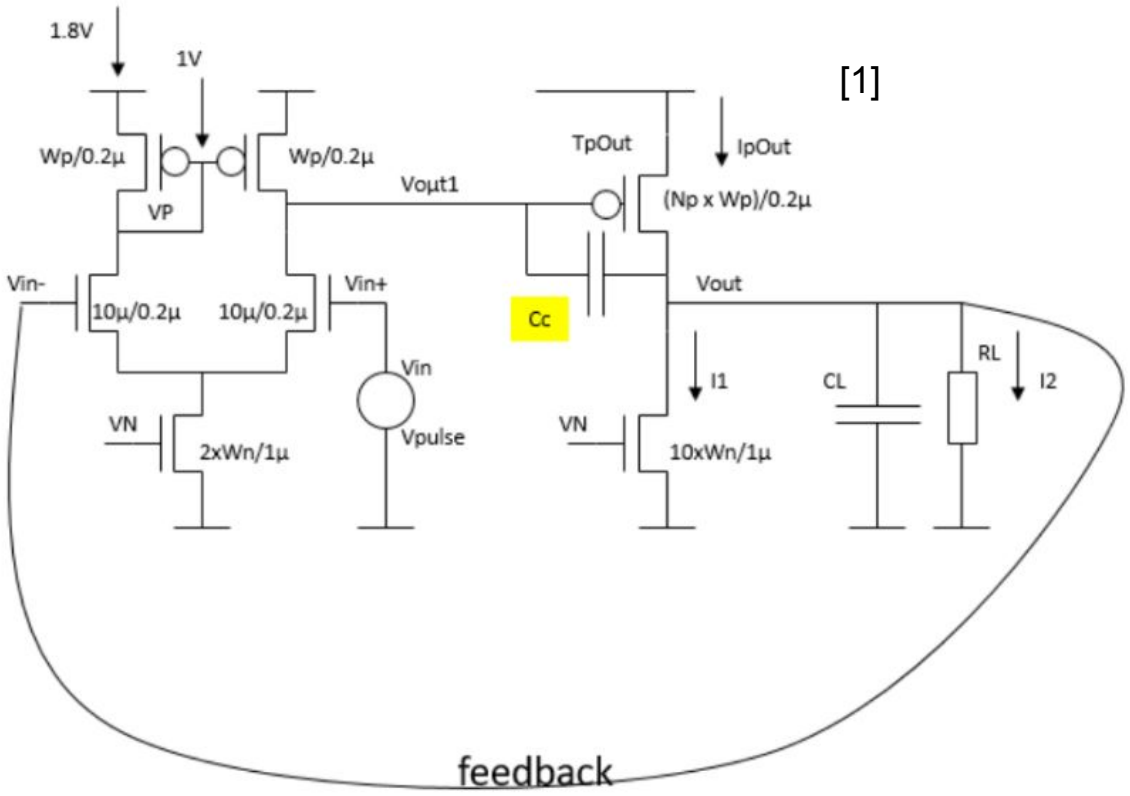


PMOS width: 4.19 um

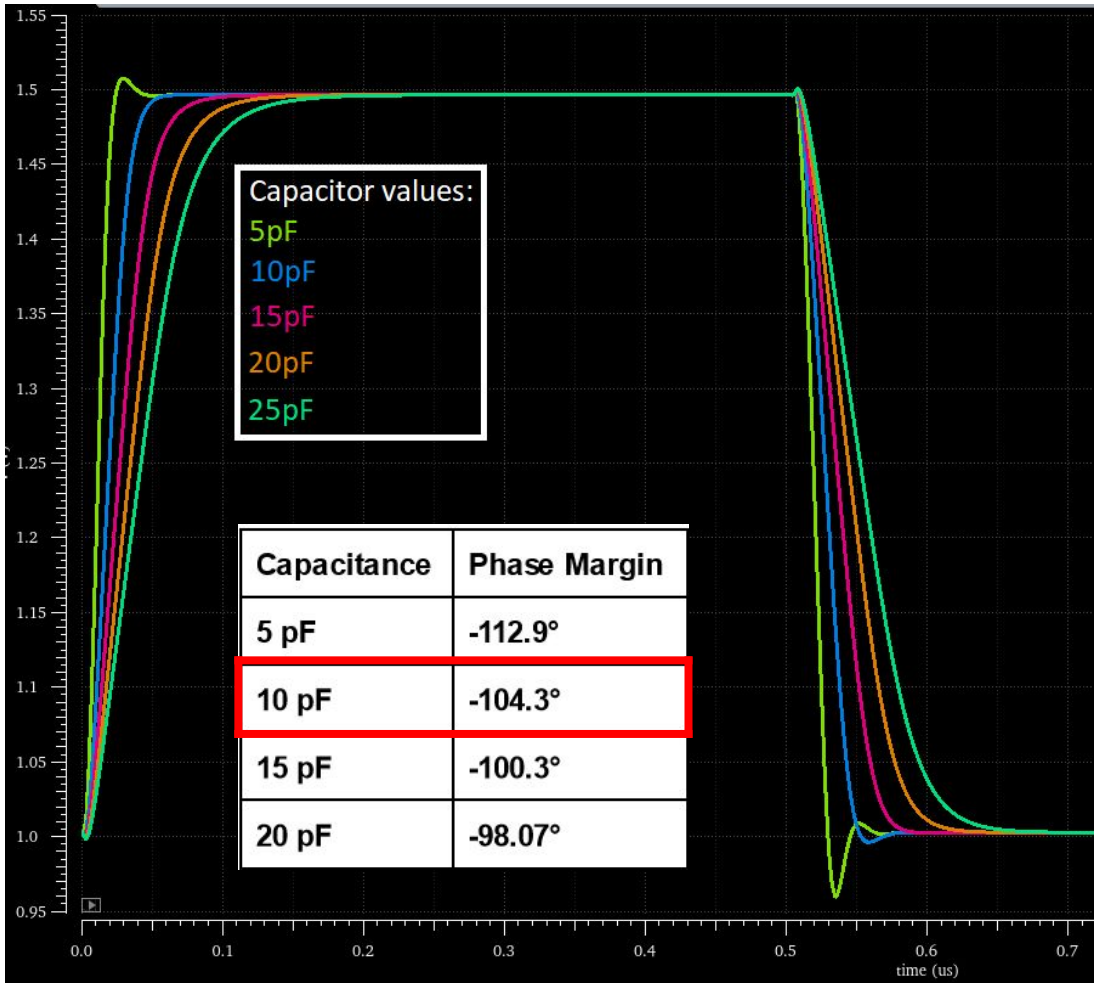
(For a voltage drop of 0.8 V in the circuits to the left)

ASIC Design: Two stage differential Amplifier (2)

Determination of Cc to stabilize the output signal



[1]



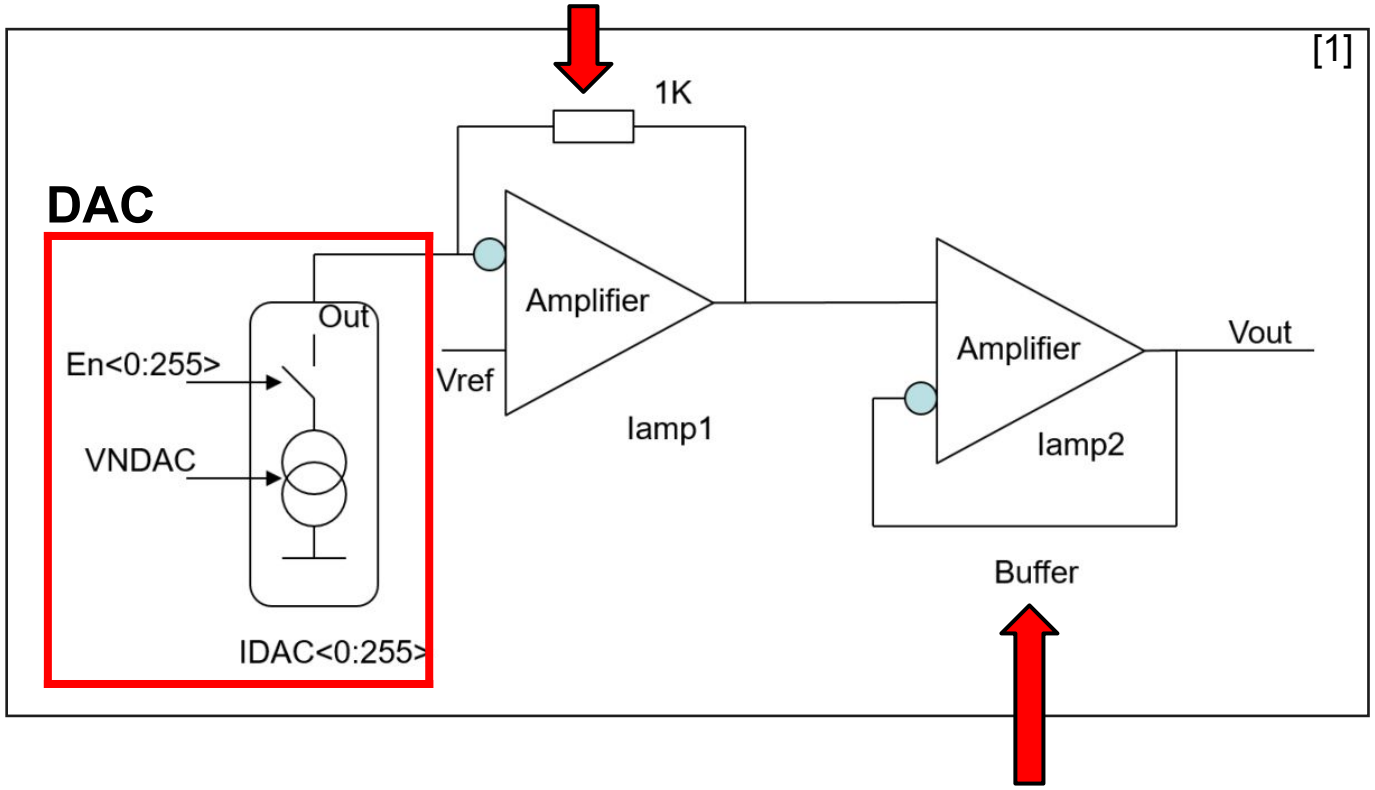
Capacitance	Phase Margin
5 pF	-112.9°
10 pF	-104.3°
15 pF	-100.3°
20 pF	-98.07°

Manuel Philippin, Marius Haschka, Kishore Mangal Singh, Yang Hu

Source:
[1] [PSoC Lab Exercises 7] page 4

ASIC Design: Digital to Analog Converter

Non-inverting amplifier circuit



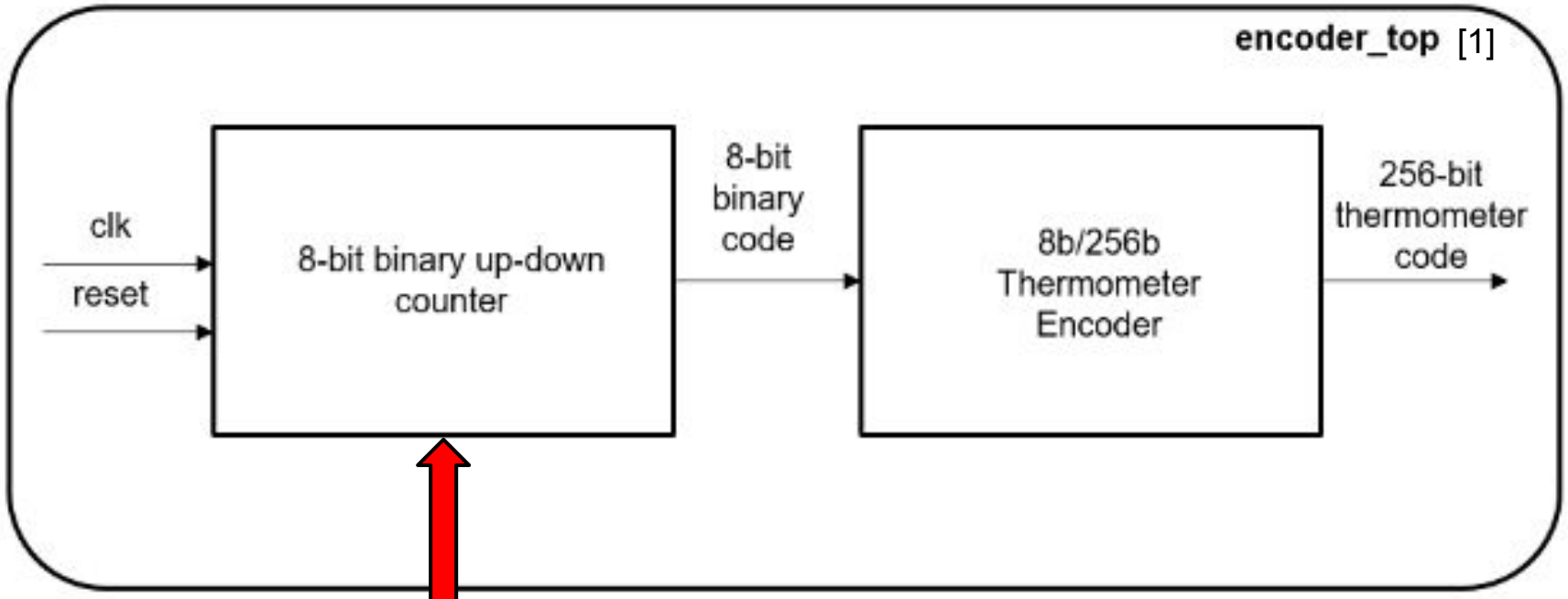
Voltage follower circuit

Thermometer code to Binary:

Number	Binary	Unary
0	0	0
1	1	10
2	10	110
3	11	1110
4	100	11110
5	101	111110
6	110	1111110
7	111	11111110

Manuel Philippin, Marius Haschka, Kishore Mangal Singh, Yang Hu
Source:
[1] [PSoC Lab Exercises 8] page 1

ASIC Design: Thermometer Encoding

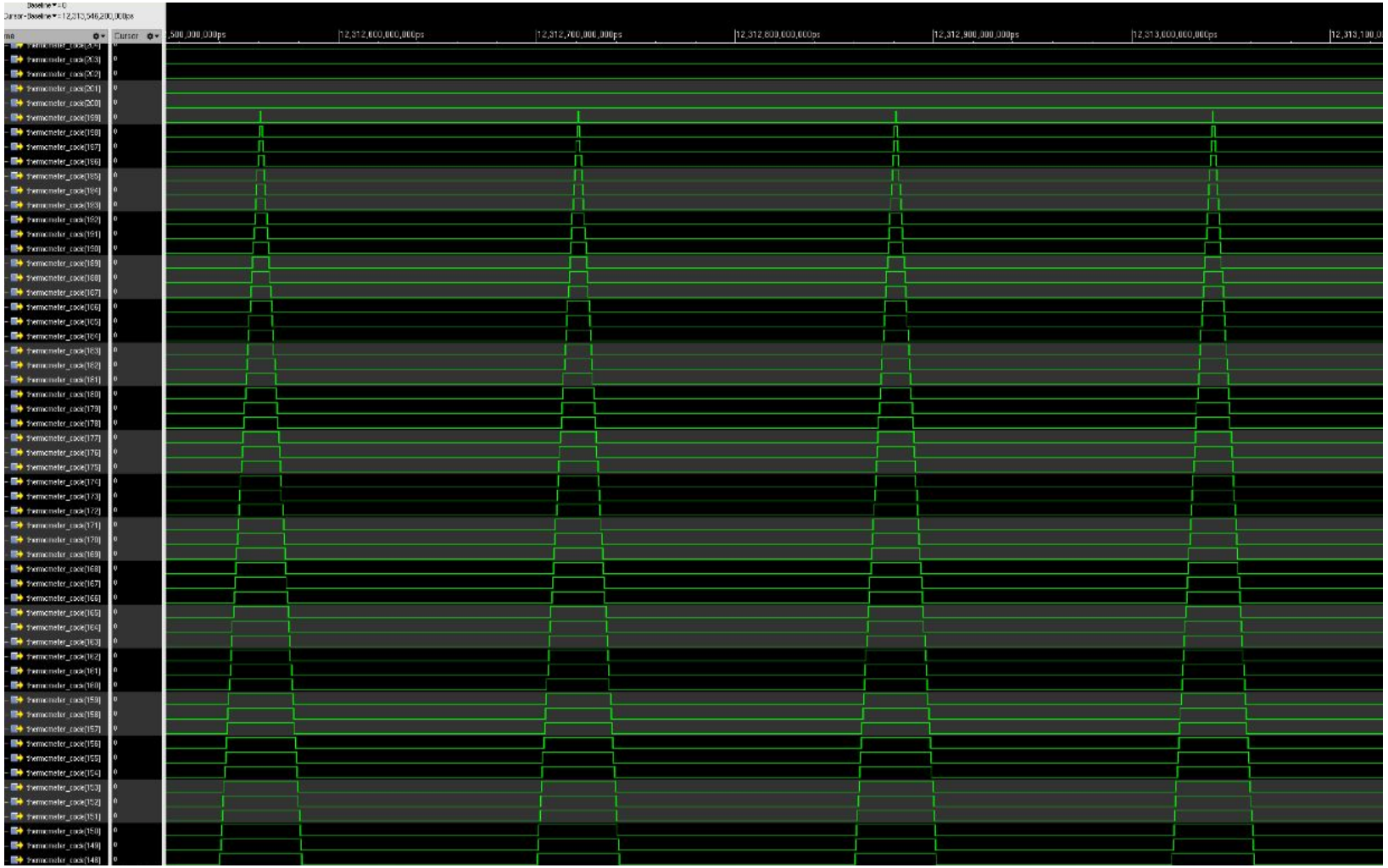



```
Parameter min = 0;
Parameter max = 200;
```

Thermometer code to Binary:

Number	Binary	Unary
0	0	0
1	1	10
2	10	110
3	11	1110
4	100	11110
5	101	111110
6	110	1111110
7	111	11111110

ASIC Design: Thermometer Encoding

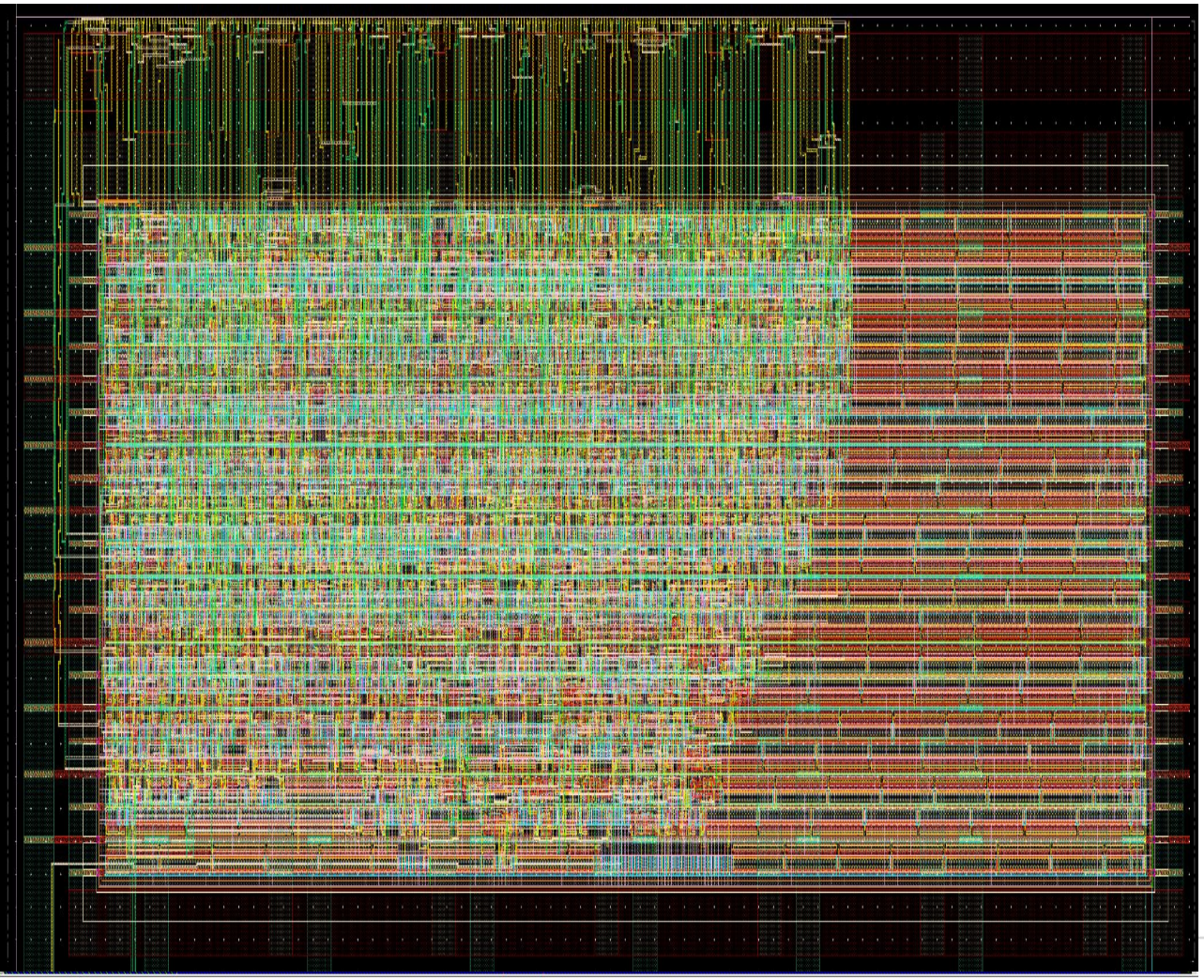


Thermometer code to Binary:

Number	Binary	Unary
0	0	0
1	1	10
2	10	110
3	11	1110
4	100	11110
5	101	111110
6	110	1111110
7	111	11111110

```
Parameter min = 0;
Parameter max = 200;
```

ASIC Design: Standard Cell Methodology



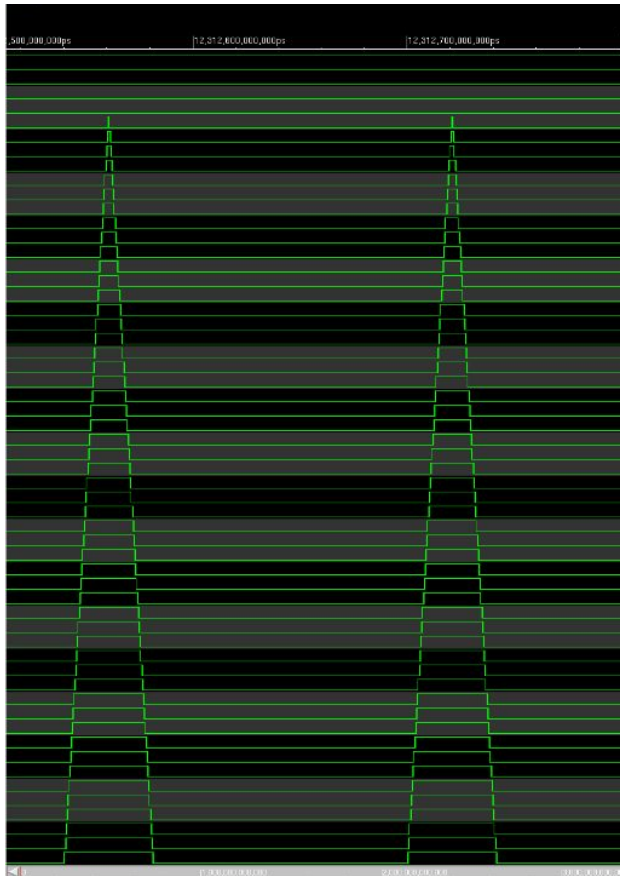
- Digital Design flow:
  - Design and implement a verilog model for 8 bit binary to 256 bit thermometer encoding.
  - Simulate to check the functionality.
  - Synthesize the verilog model using Cadence Genus.
  - Post Synthesis gate level simulation.
  - Floorplanning of the design for constraints.
  - Run automatic place and route flow using Cadence Innovus.
  - At every stage, the timing an analysis is done.
  - Stream out design and import it in Virtuoso for later integration with our Amplifier Design.

Challenges: Parameter/Register definition issues

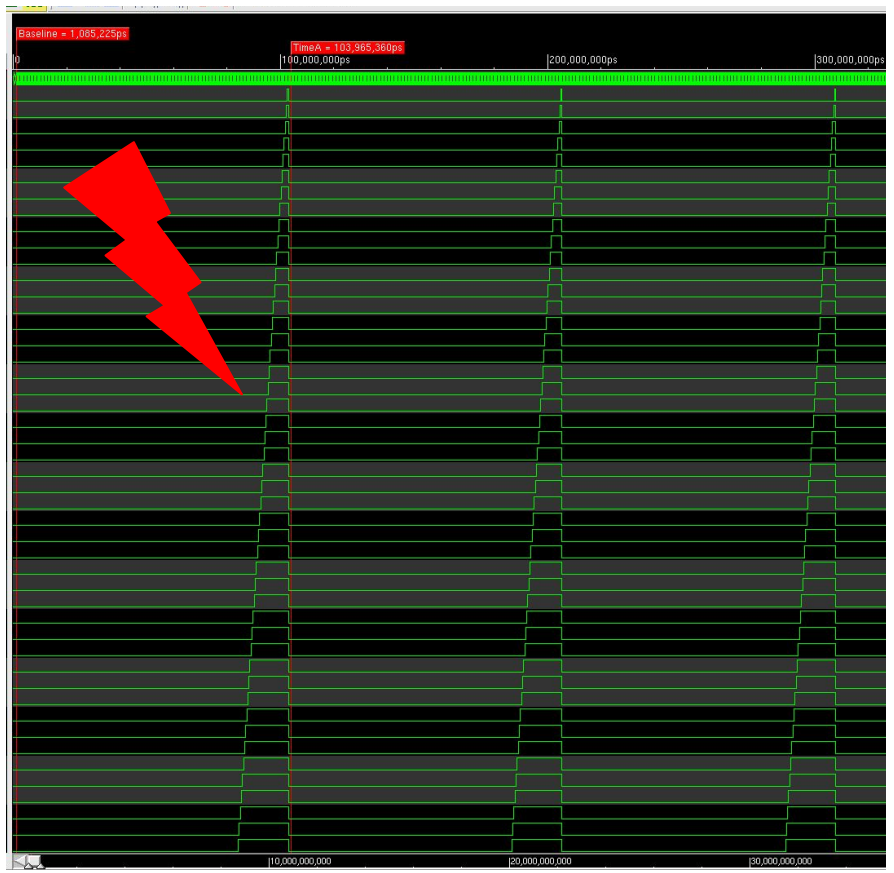
- Original implementation
  ```
  Reg [7:0] min = 0;
  Reg [7:0] max = 200;
  ```
- Correct in RTL simulation
- Counter Error in GTL simulation
  Synthesis not working

- Solution:
  ```
  Parameter min = 0;
  Parameter max = 200;
  ```

Think about the physical part when writing Verilog code



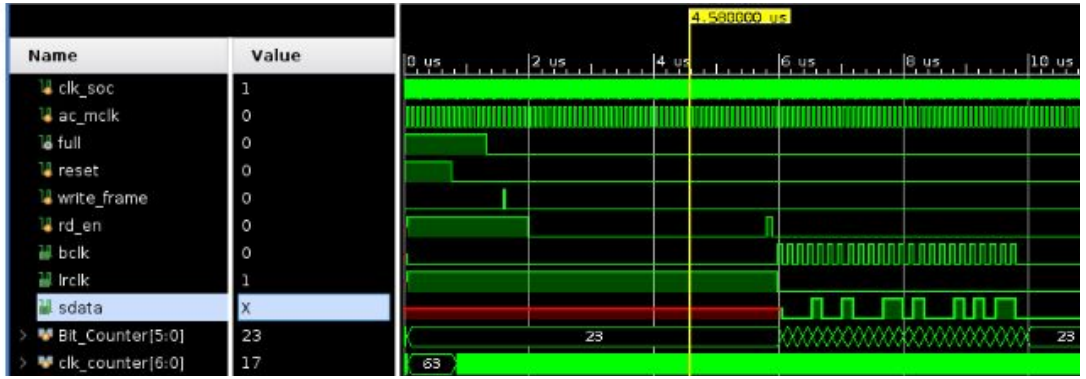
RTL Simulation



Error in GTL Simulation

Challenges: I2S

Module started to instantly send data & trigger bclk without data from the FIFO



Introduced a “*data_av_flag*”

➔ Pulled high after data is received
➔ Checked for every transmission

```
left_data_buffer <= left_data;
right_data_buffer <= right_data;
data_av_flag = 1'b1;
```

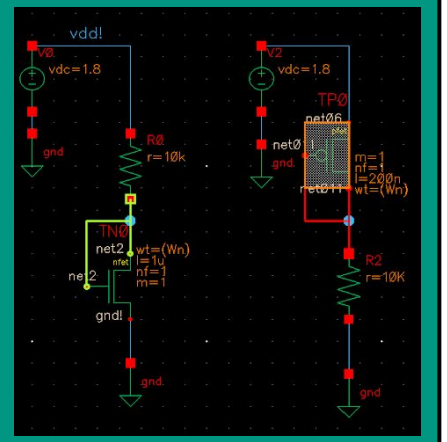
```
if (data_av_flag == 1) begin    //only
transmit data if there is some data in the
buffer
    if (bclk == 1) begin
```

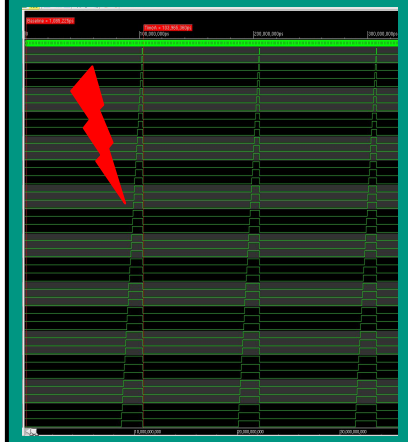

Manuel Philippin, Marius Haschka, Kishore Mangal Singh, Yang Hu

What we learned

Optimizing the team structure through **teamwork** and the distribution of the tasks increases the **work efficiency**

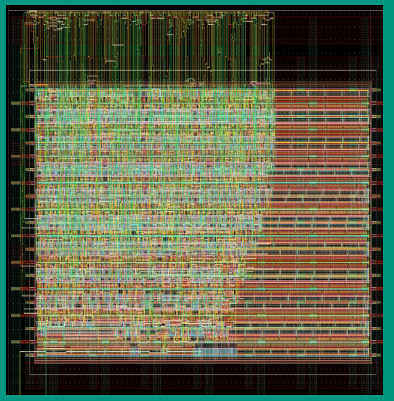
Combining our theoretical knowledges from lectures DAS, HSO, HMS etc. in the **actual physical design**





Deeper knowledge about debugging is not found in theoretical lectures and has to be acquired by working on actual projects

Gaining **practical knowledge** of working with tools that are standard in the industry



Source:
[1] https://www.pexels.com/photo/photo-of-people-near-wooden-table-31848/