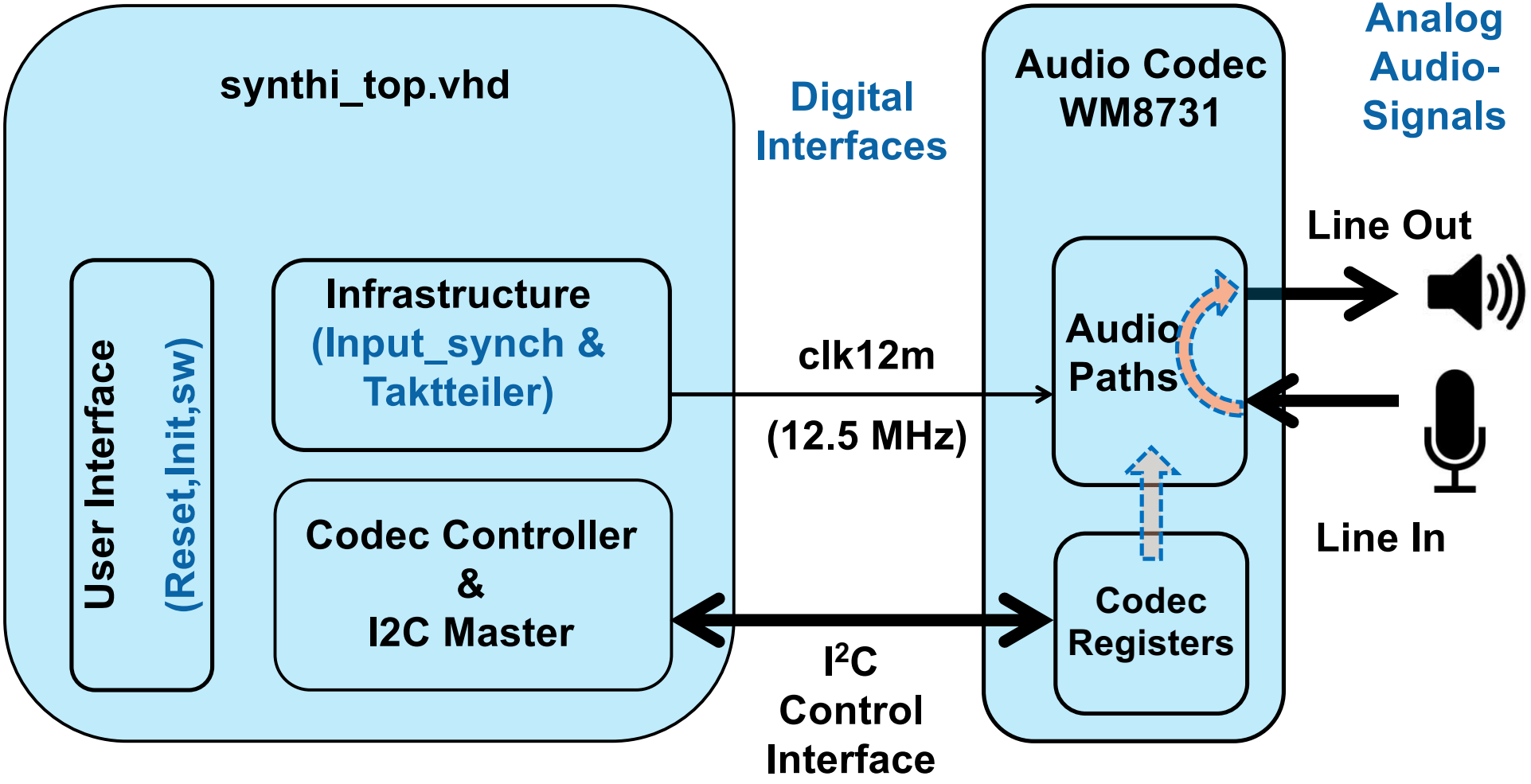
Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

Codec Steuerung mit I2C

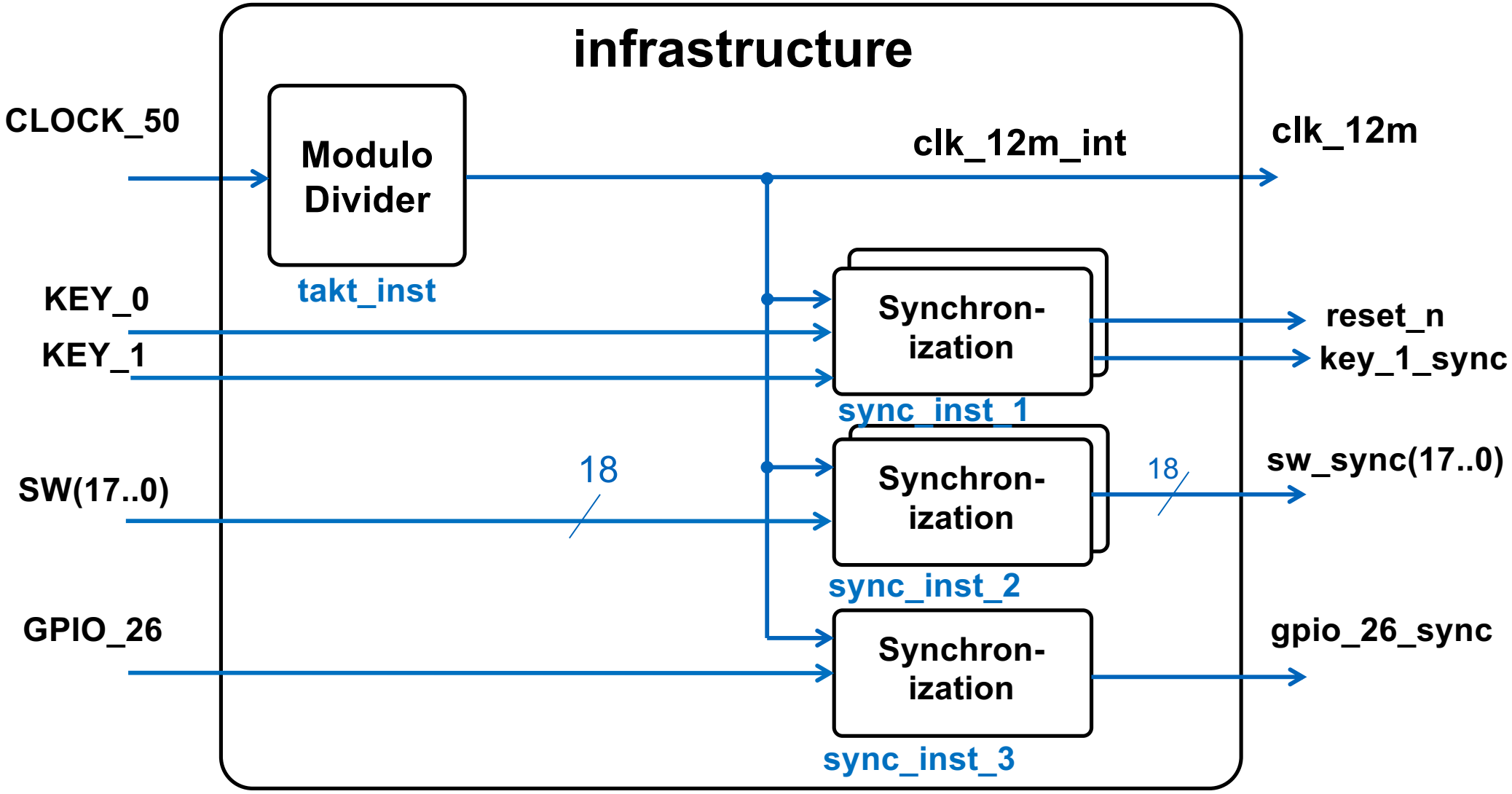
Infrastructure Block
(Teil von Phase-1)

Zürcher Hochschule
für Angewandte Wissenschaften



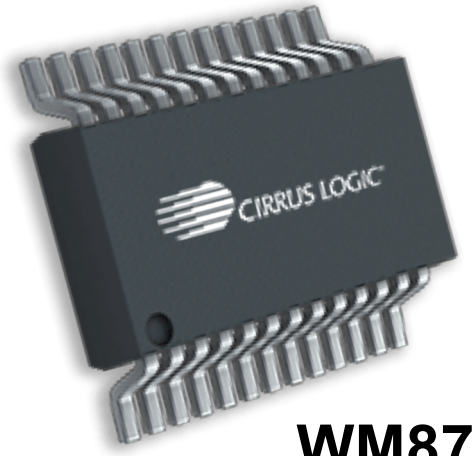
Infrastructure Block

Taktteiler & Synchronisation



Achtung: Modulo Divider und Synchronizer nicht mit Reset versehen

Audio Codec

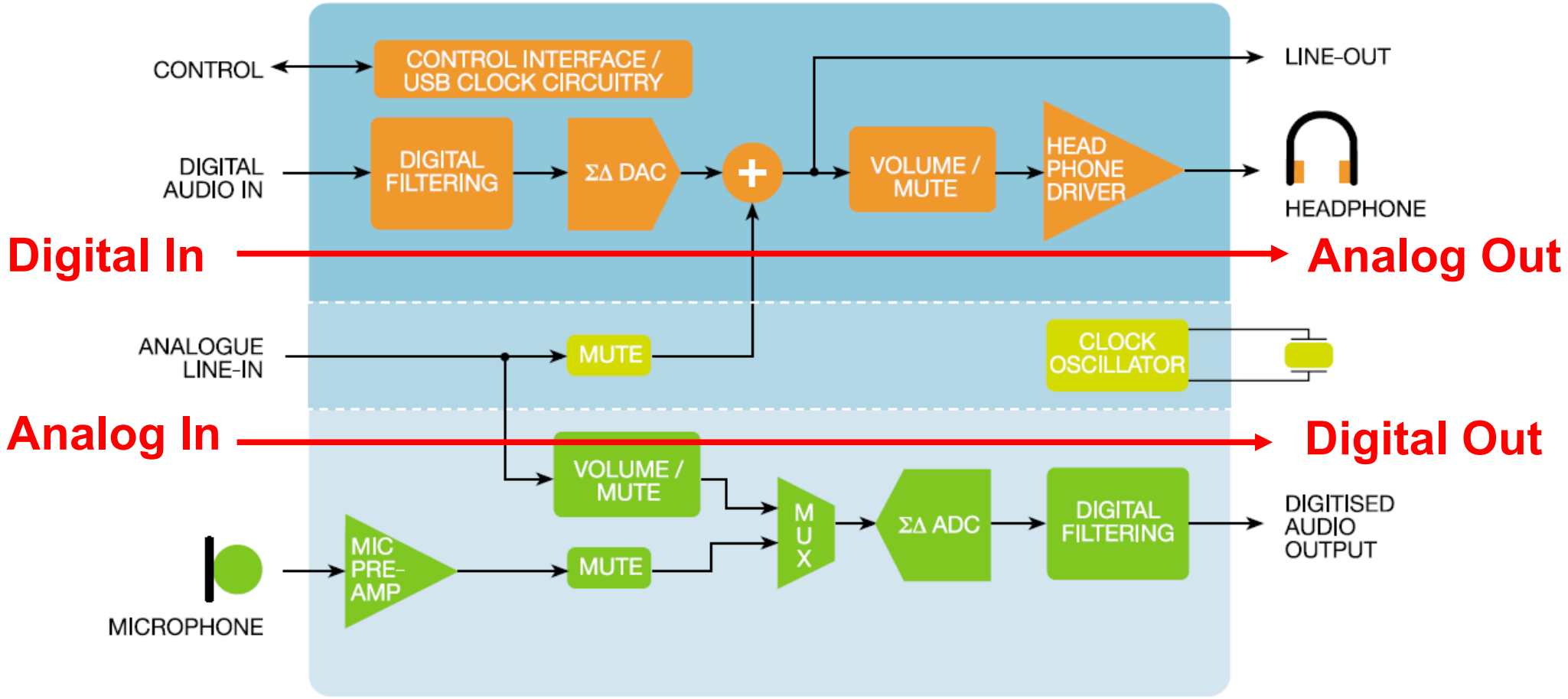


WM8731

Sampling Frequency:	8kHz – 96kHz
ADC SNR:	90dB (‘A’ weighted)
DAC SNR:	100dB (‘A’ weighted)
Headphone Amplifier:	2 x 50 mW
Preis:	SFr. 1.61 (1000 Stück)
Datenblatt:	OLAT: *Literatur*

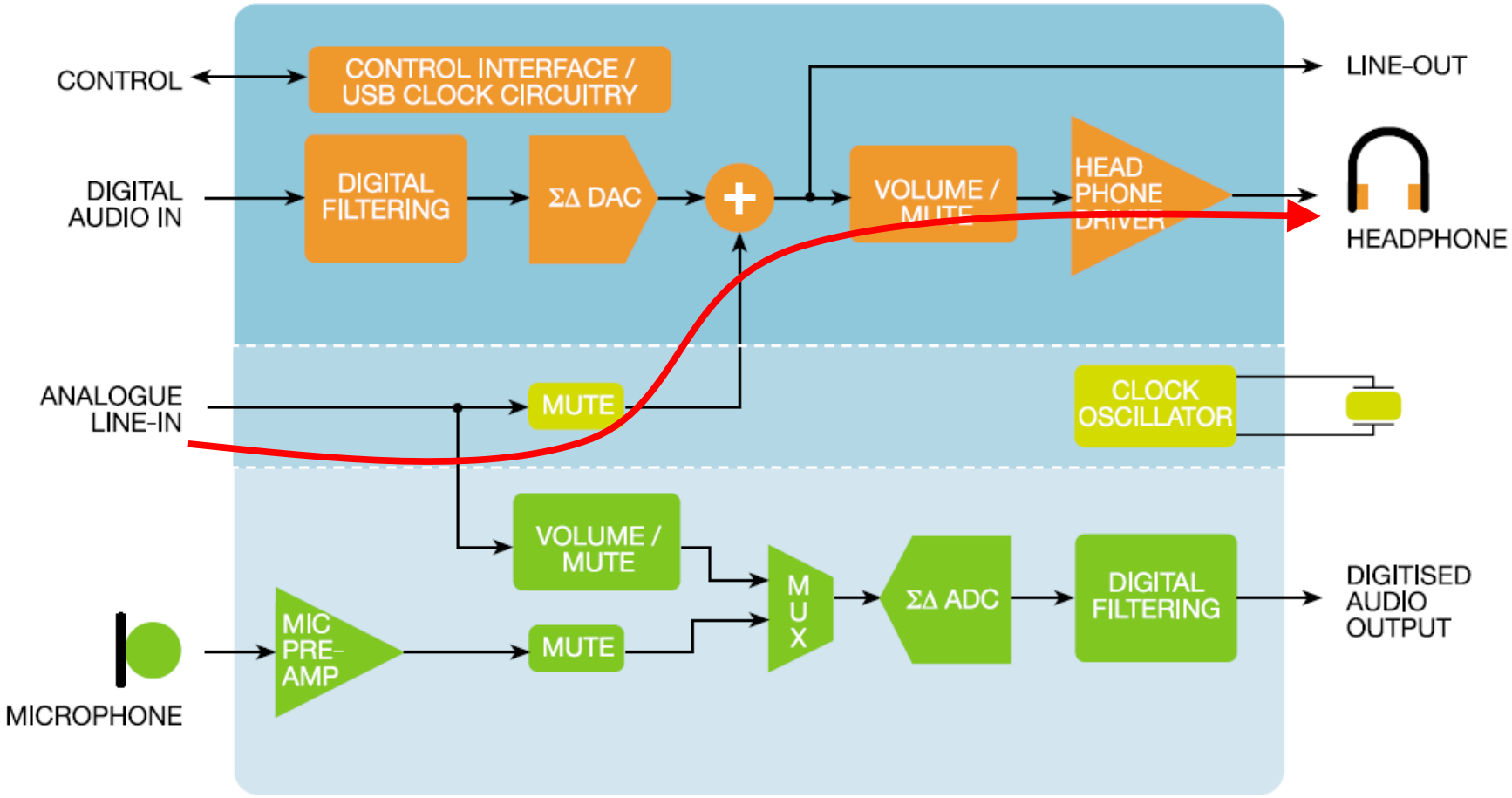
Audio Codec Block Diagram

Normal Operation



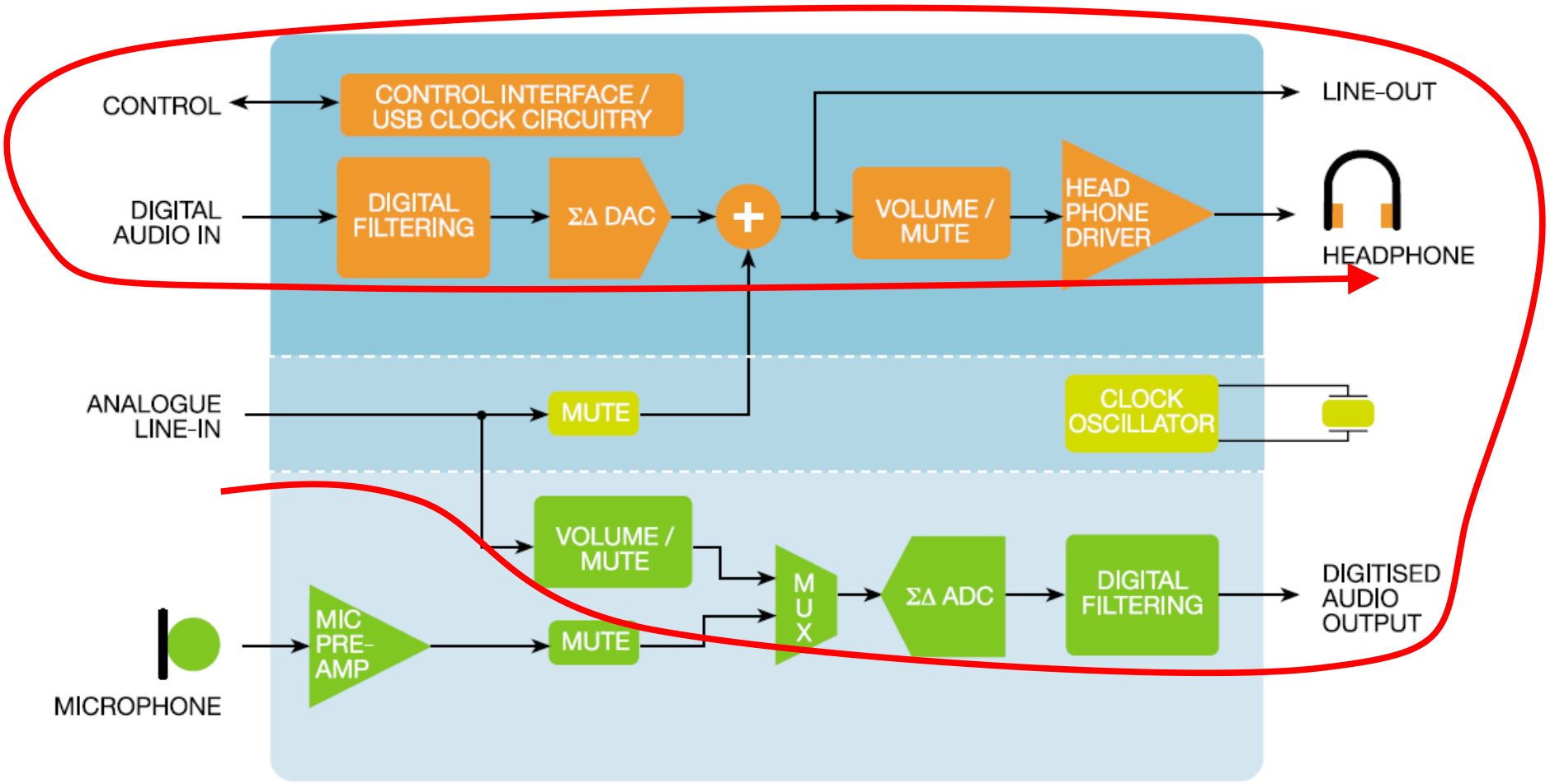
Audio Codec Block Diagram

Analog Loopback

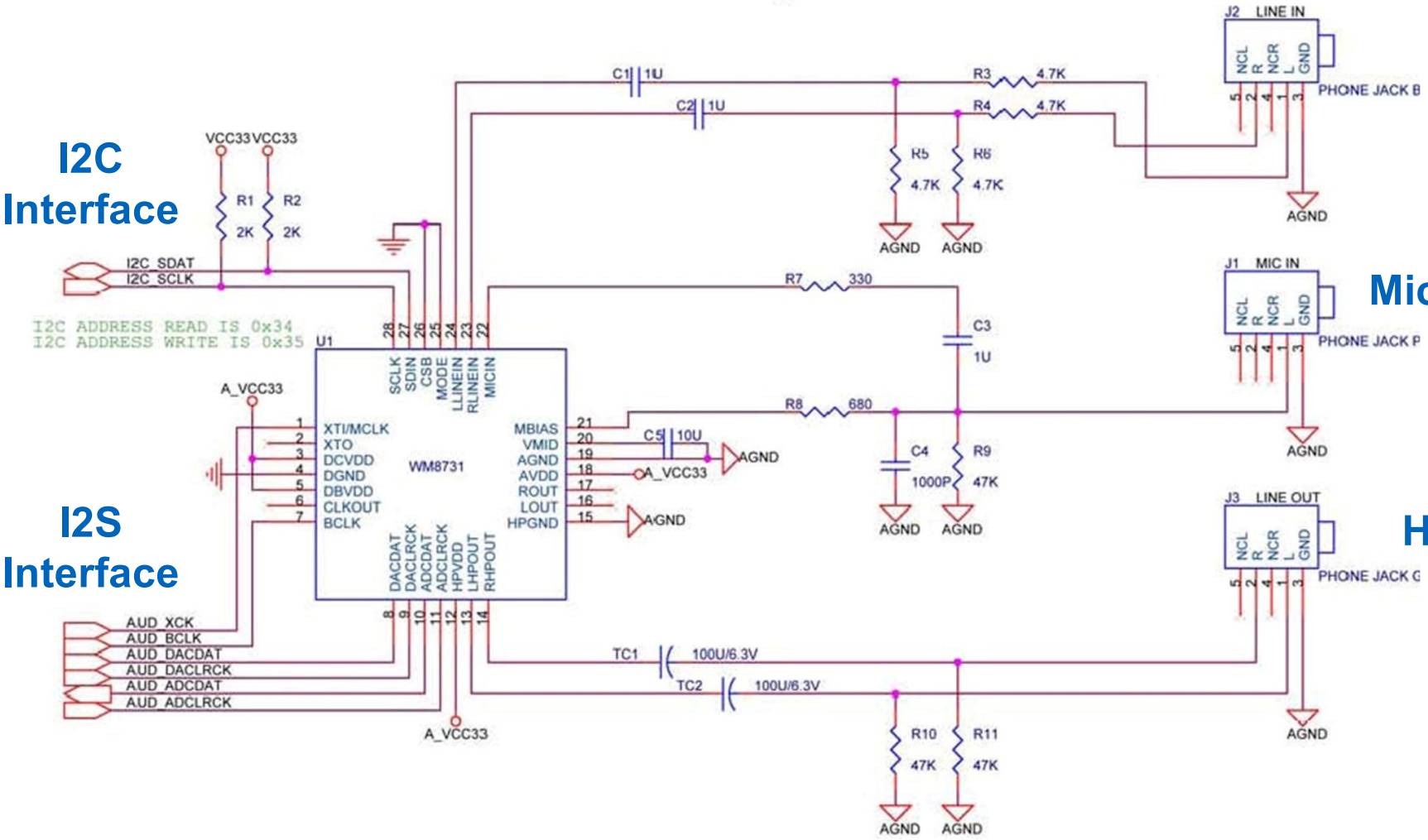


Audio Codec Block Diagram

Digital Loopback



WM8731 on the DE2-115 Board



Overview over Audio Codec Registers

- R0 : Left Line In
- R1 : Right Line In
- R2 : Left Headphone Out
- R3 : Right Headphone Out
- R4 : Analog Audio Control Path
- R5 : Digital Audio Path Control
- R6 : Power Down Control
- R7 : Digital Audio Interface Format
- R8 : Sampling Control (sample rate)
- R9 : Activation (activate digital audio interface)
- R10 - 14 : Reserved
- R15 : Reset

Audio Codec Register Map

Register Addresses (7-bits) **Register Data (9-bits)**

REGISTER	B 15	B 14	B 13	B 12	B 11	B 10	B 9	B8	B7	B6	B5	B4	B3	B2	B1	B0
R0 (00h)	0	0	0	0	0	0	0	LRIN BOTH	LIN MUTE	0	0	LINVOL				
R1 (02h)	0	0	0	0	0	0	1	RLIN BOTH	RIN MUTE	0	0	RINVOL				
R2 (02h)	0	0	0	0	0	1	0	LRHP BOTH	LZCEN	LHPVOL						
R3 (06h)	0	0	0	0	0	1	1	RLHP BOTH	RZCEN	RHPVOL						
R4 (08h)	0	0	0	0	1	0	0	0	SIDEATT		SIDETONE	DAC SEL	BY PASS	INSEL	MUTE MIC	MIC BOOST
R5 (0Ah)	0	0	0	0	1	0	1	0	0	0	0	HPOR	DAC MU	DEEMPH		ADC HPD
R6 (0Ch)	0	0	0	0	1	1	0	0	PWR OFF	CLK OUTPD	OSCPD	OUTPD	DACPD	ADCPD	MICPD	LINEINPD
R7 (0Eh)	0	0	0	0	1	1	1	0	BCLK INV	MS	LR SWAP	LRP	IWL		FORMAT	
R8 (10h)	0	0	0	1	0	0	0	0	CLKO DIV2	CLKI DIV2	SR				BOSR	USB/ NORM
R9 (12h)	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	ACTIVE
R15(1Eh)	0	0	0	1	1	1	1	RESET								
	ADDRESS							DATA								

Audio Codec Settings

WM8731 on the DE2-115 Board

- **FPGA-System-Clock**

  Alle Blöcke in FPGA (ausser Taktteiler) werden mit dem clk_12m getriggert (synchrones Design mit «single clock-domain»).

- **clk12m:**

  Der Master Clock für das System und den Audio Codec. Wird auch MCLK oder AUD_XCK genannt (Datenblatt oder DE2-115 board)

- **Audio Abtastrate: "Normal Mode" = 48 kHz (andere Abtastraten möglich)**

SAMPLING RATE		MCLK FREQUENCY	SAMPLE RATE REGISTER SETTINGS					DIGITAL FILTER TYPE
ADC	DAC							
kHz	kHz	MHz	BOSR	SR3	SR2	SR1	SR0	
48	48	12.288	0 (256fs)	0	0	0	0	1

Audio Codec Register 8

Audio Codec Settings

Example: W8731_ADC_DAC_M12DB_48K

Zürcher Hochschule für Angewandte Wissenschaften

zh aw

Register	Address Bits B[15:9]	Data Bits B[8:0]	Beschreibung
R0 LEFT_LINE_IN	*"0000000"* = *0x00*	*"000001111"* = *0x00F*	No mute, no both, volume -12 dB
R1 RIGHT_LINE_IN	*"0000001"* = 0x01	*"000001111"* = *0x00F*	No mute, no both, volume -12 dB
R2 LEFT_HP_OUT	*"0000010"* = 0x02	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R3 RIGHT_HP_OUT	*"0000011"* = 0x03	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R4 ANALOG_AP	*"0000100"* = 0x04	"000010010" = 0x012	Mute microphone, select DAC, disable bypass, disable sidetone,…
R5 DIGITAL_AP	*"0000101"* = 0x05	"000000000" = 0x000	Disable DAC mute, disable de-emphasis, …
R6 POWER_DOWN	*"0000110"* = 0x06	"000000000" = 0x000	No power down mode selected
R7 DIGITAL_AI	*"0000111"* = 0x07	"000000010" = 0x002	Audio-IF I2S Mode, audio data 16 bits wide, …
R8 SAMPLING	*"0001000"* = 0x08	"000000000" = 0x000	Normal mode, MCLK 12,28MHz, sample rate 48kHz (both ADC and DAC), SR=[0011], …
R9 DIGITAL_ACTIVATE	*"0000111"* = 0x09	"000000001" = 0x001	Activate Audio Digital Interface
R10 – R14 RESERVED A-E	0x0A - 0x0E	"000000000" = 0x000	No need to write
R15 RESET	*"0001111"* = 0x0F	"000000000" = 0x000	Only write if SW-reset required

Audio Codec Settings

Example: W8731_BYPASS_M12DB_8K

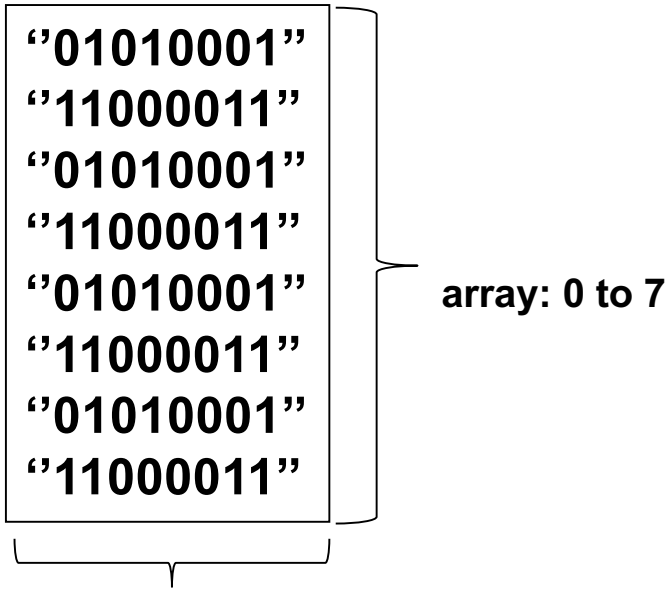
Zürcher Hochschule für Angewandte Wissenschaften

zh aw

Register	Address Bits B[15:9]	Data Bits B[8:0]	Beschreibung
R0 LEFT_LINE_IN	*"0000000" = 0x00*	*"000001111" = 0x00F*	No mute, no both, volume -12 dB
R1 RIGHT_LINE_IN	*"0000001"* = 0x01	*"000001111" = 0x00F*	No mute, no both, volume -12 dB
R2 LEFT_HP_OUT	*"0000010"* = 0x02	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R3 RIGHT_HP_OUT	*"0000011"* = 0x03	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R4 ANALOG_AP	*"0000100"* = 0x04	"000001010" = 0x00A	Mute microphone, disable DAC, enable bypass, disable sidetone,…
R5 DIGITAL_AP	*"0000101"* = 0x05	"000001000" = 0x008	Enable DAC mute, disable de-emphasis, …
R6 POWER_DOWN	*"0000110"* = 0x06	"000000000" = 0x000	No power down mode selected
R7 DIGITAL_AI	*"0000111"* = 0x07	"000000010" = 0x002	Audio-IF I2S Mode, audio data 16 bits wide, …
R8 SAMPLING	*"0001000"* = 0x08	"000000000" = 0x00C	Normal mode, MCLK 12,28MHz, sample rate 48kHz (both ADC and DAC), SR=[0011], …
R9 DIGITAL_ACTIVATE	*"0000111"* = 0x09	"000000001" = 0x001	Activate Audio Digital Interface
R10 – R14 RESERVED A-E	0x0A - 0x0E	"000000000" = 0x000	No need to write
R15 RESET	*"0001111"* = 0x0F	"000000000" = 0x000	Only write if SW-reset required

Typendefinierung eines Arrays

```
type codec_array is array(0 to 7) of std_logic_vector(8 downto 0);
```



Array mit Konstanten füllen

```
constant: C_W8731_ANALOG_BYPASS: codec_array :=
(
0 => “011001101”
1 => “111001101”,
2 => “111001101”,
3 => “111001101”,
4 => “011001101”,
5 => “011001101”,
6 => “011001101”,
7 => “011001101”,
8 => “011001101”,
9 => “011001101”
);
```

Achtung: Dateninhalt

nur Beispiel

Auslesen eines Arrays

In Entity:

```
output_data : out std_logic_vector(8 downto 0);
```

In Architektur:

```
output_data <= C_W8731_ANALOG_BYPASS(array_pointer);
```

Package: reg_table_pkg

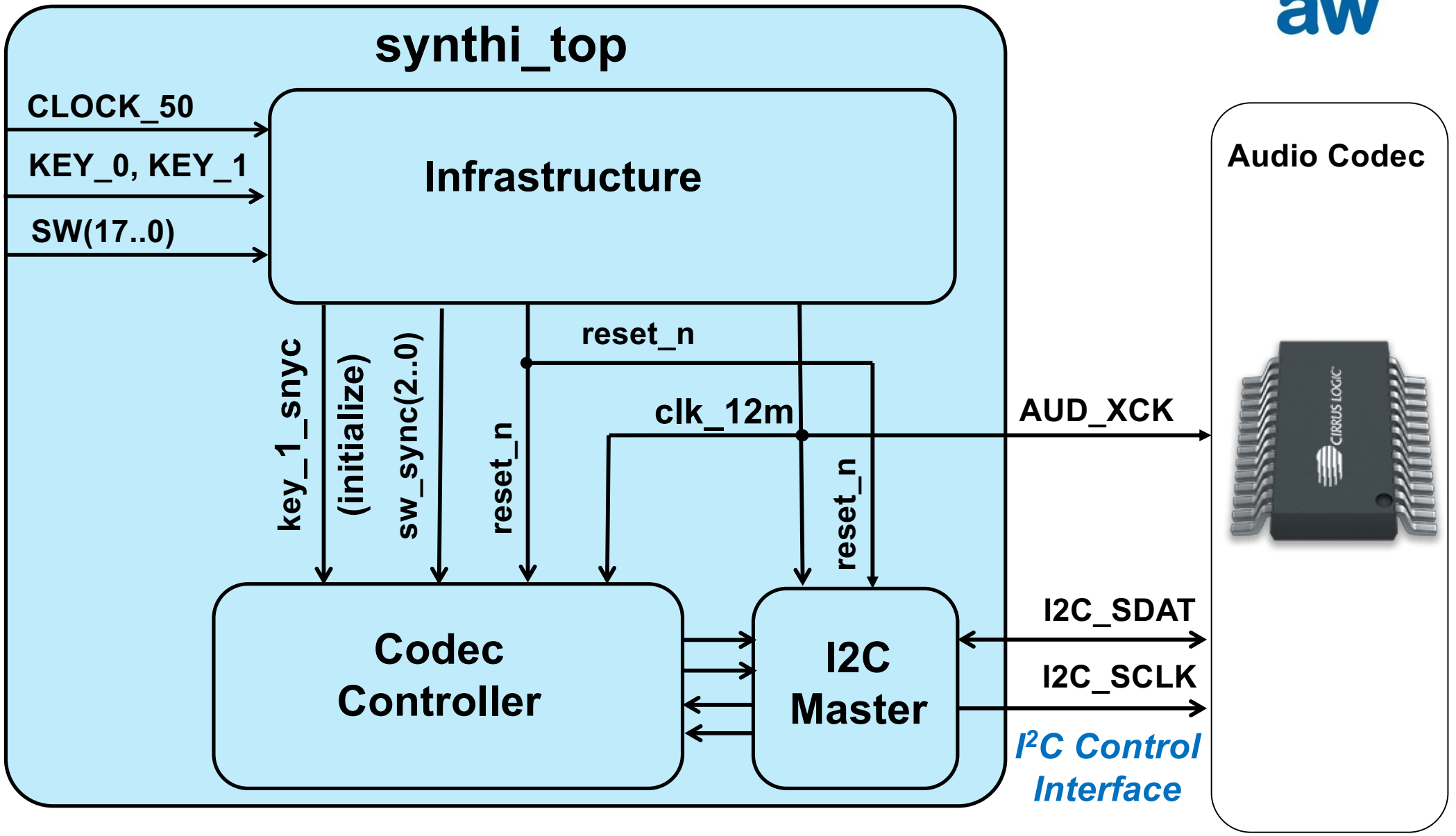
Die folgenden Initialisierungstabellen sind in *reg_table_pkg.vhd* enthalten

```
C_W8731_ANALOG_BYPASS

C_W8731_ANALOG_MUTE_LEFT

C_W8731_ANALOG_MUTE_RIGHT

C_W8731_ANALOG_MUTE_BOTH

C_W8731_ADC_DAC_0DB_48K
```

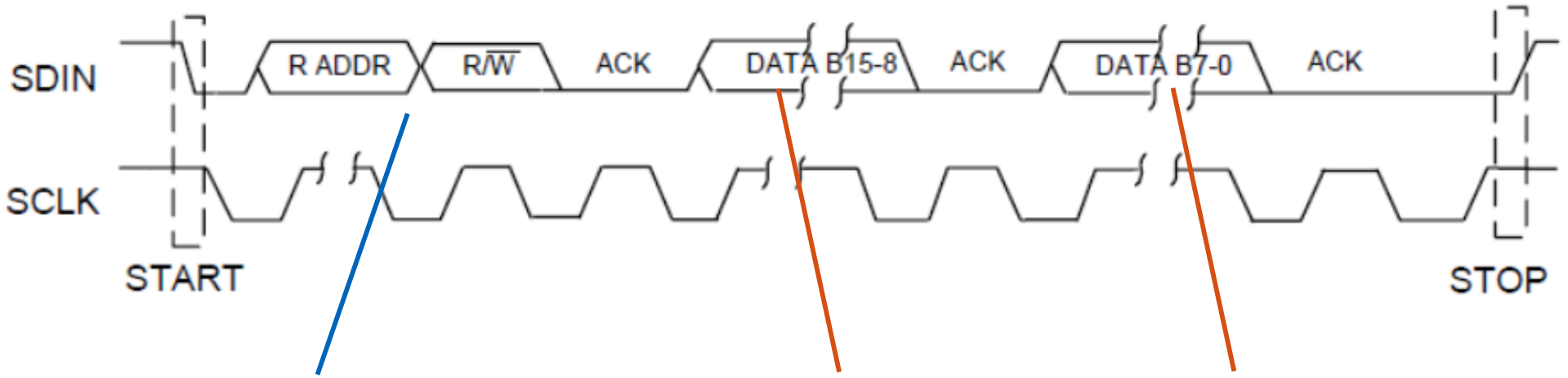
Damit die Tabellen benutzt werden können, muss *reg_table_pkg* in *codec_controller.vhd* eingebunden werden mit:

```
use work.reg_table_pkg.all;
```

Synthi-Top mit Codec Control Interface



I2C Interface (2-wire Interface)



I2C-Chip-Address (7-bit + R/W)

I2C-Data Byte 1 (MSB)

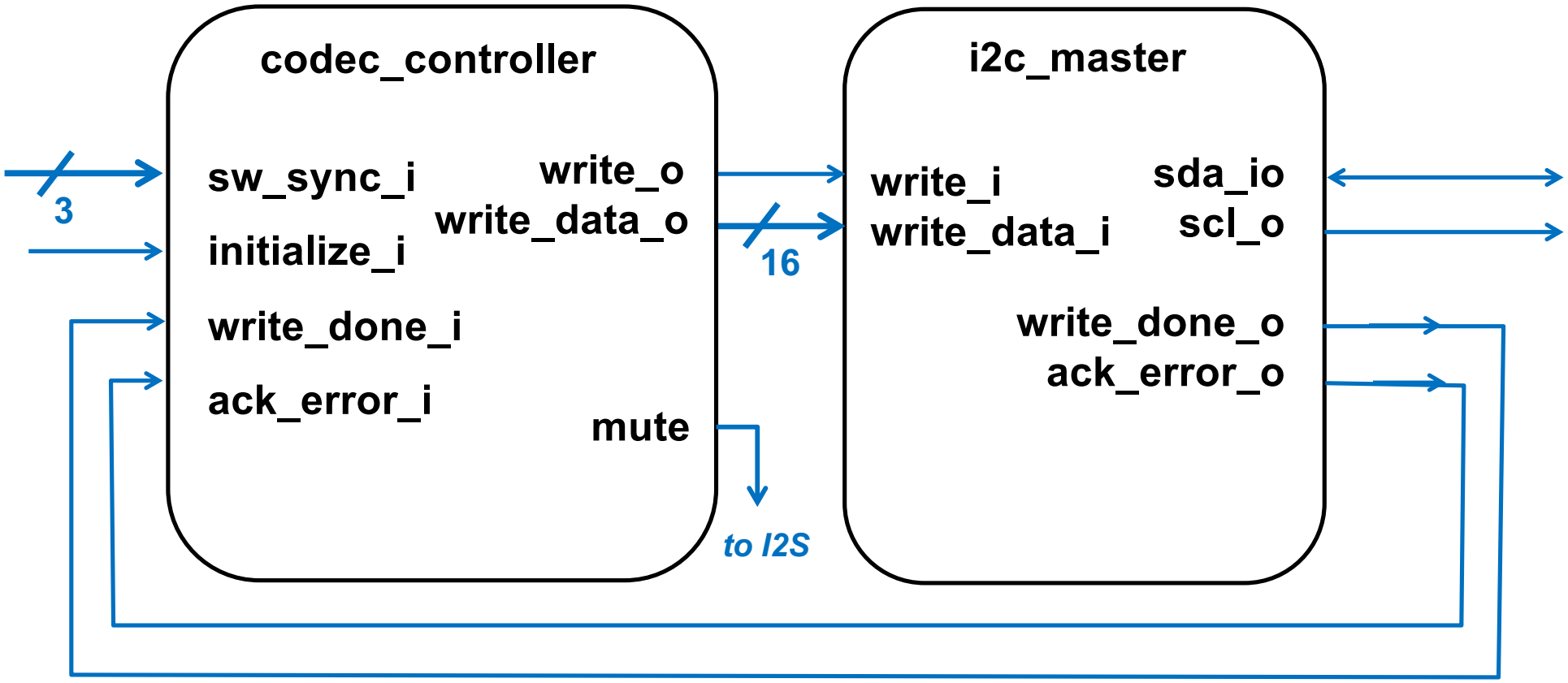
I2C-Data Byte 2 (LSB)

Register Addresses (7-bits)

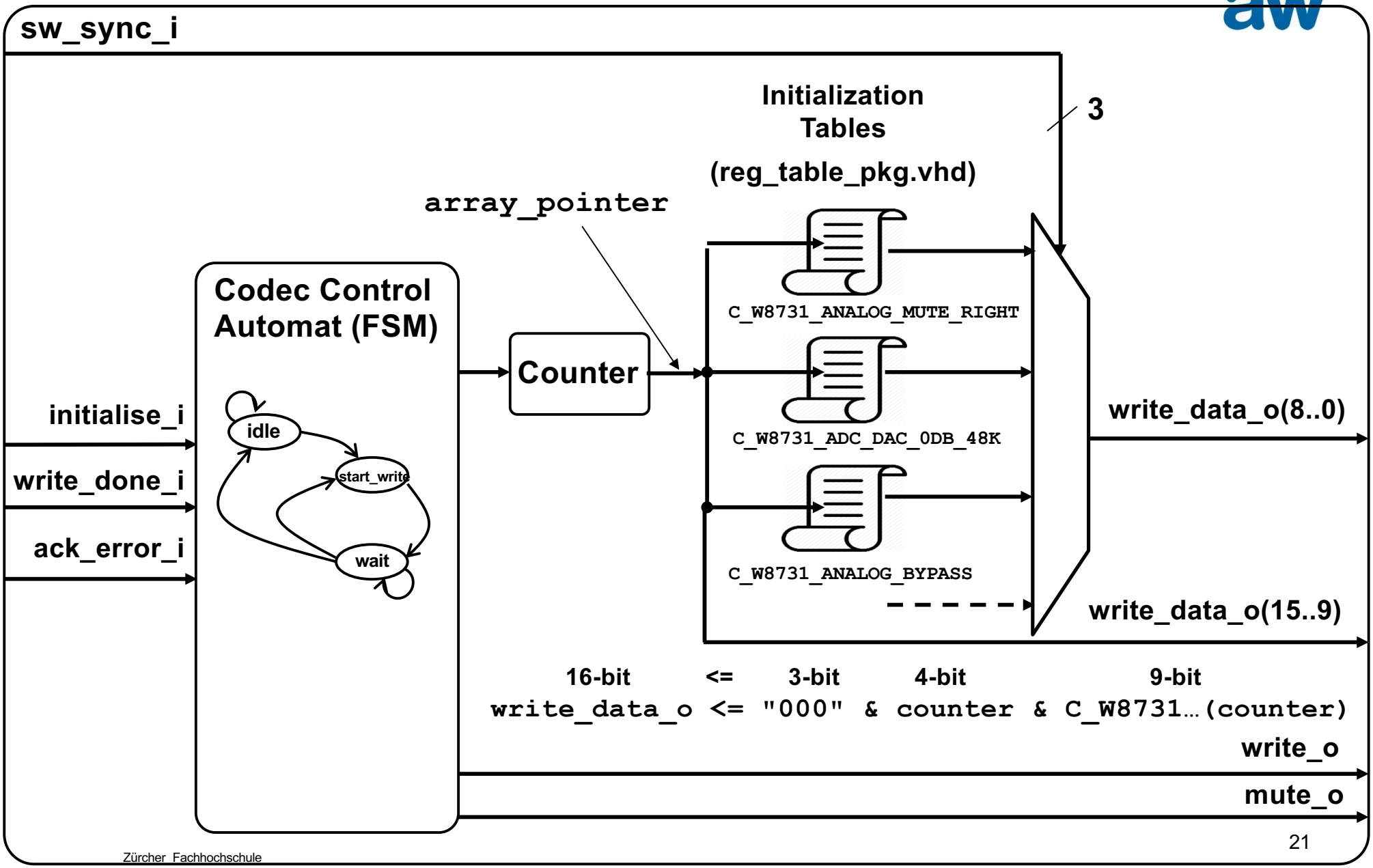
Register Data (9-bits)

REGISTER	B 15	B 14	B 13	B 12	B 11	B 10	B 9	B8	B7	B6	B5	B4	B3	B2	B1	B0
R0 (00h)	0	0	0	0	0	0	0	LRIN BOTH	LIN MUTE	0	0	LINVOL				

Codec Control Interface

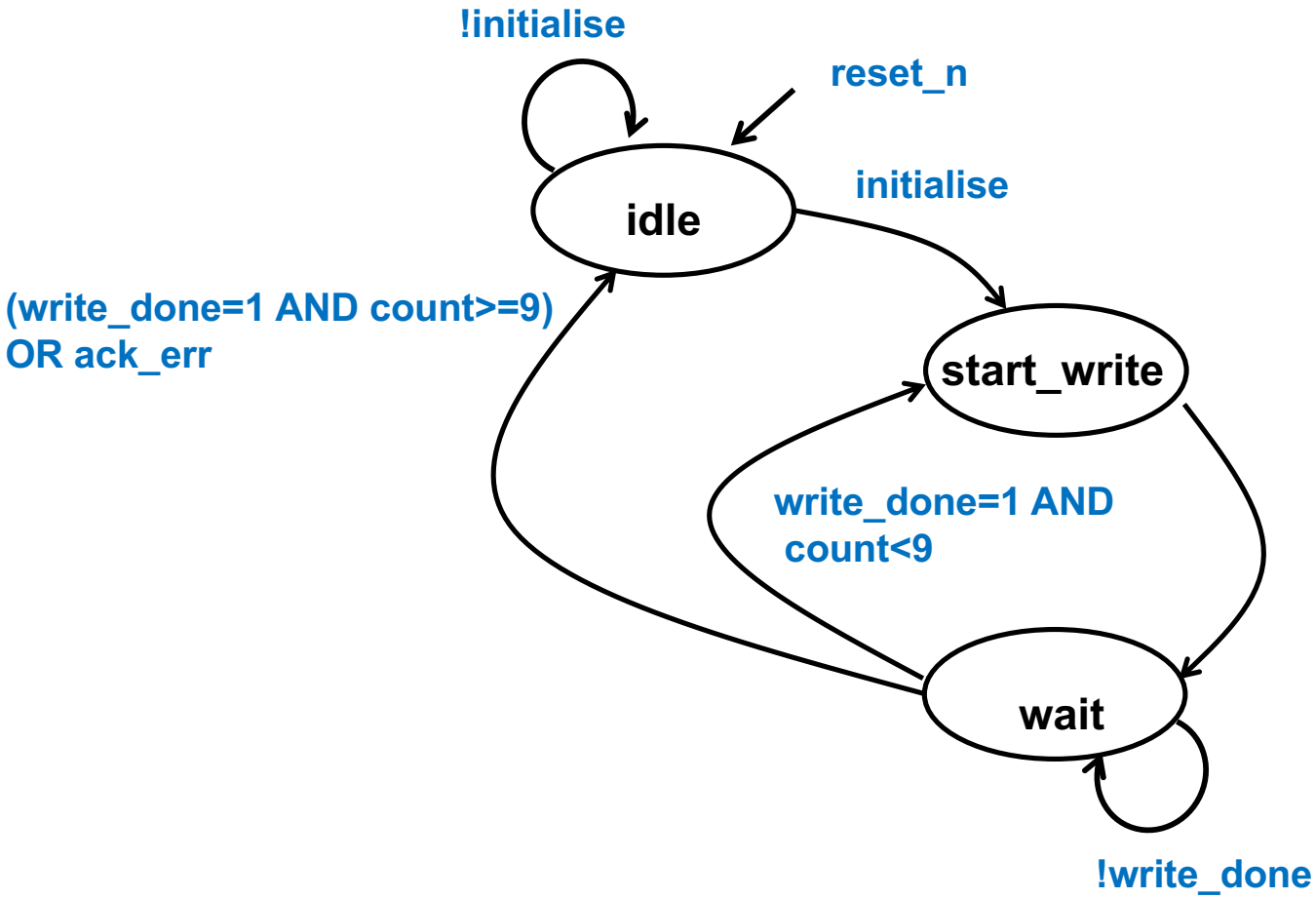


Codec Controller



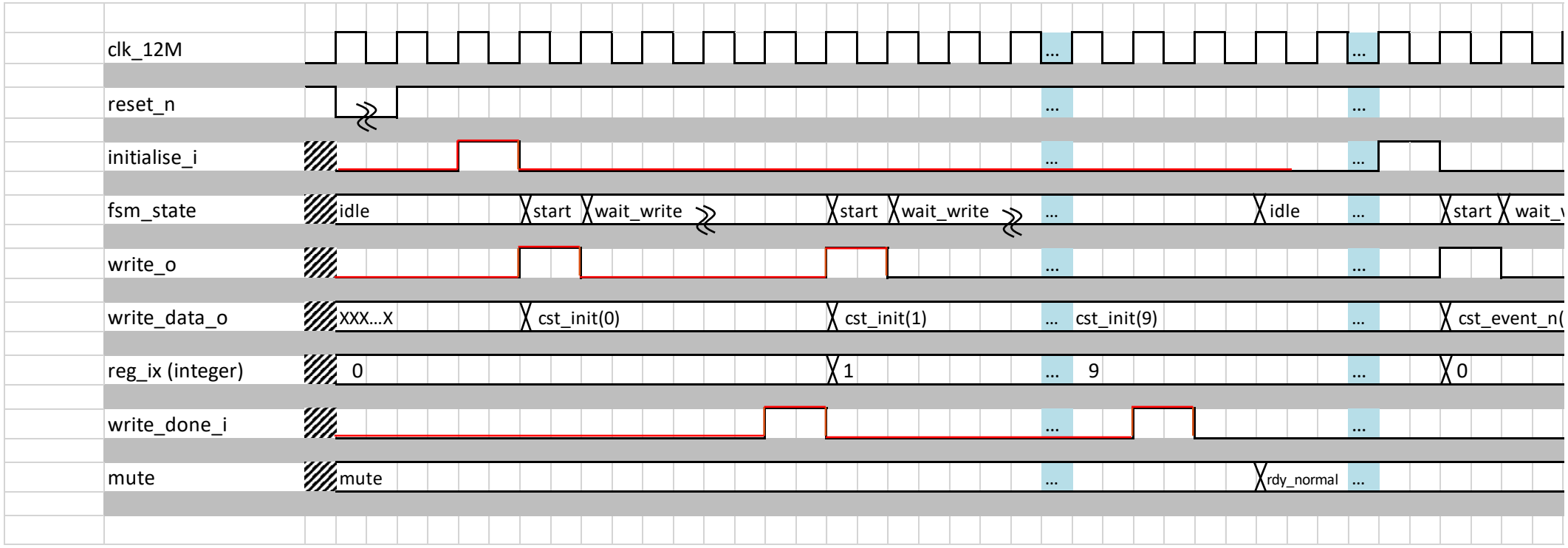
SW(2)	SW(1)	SW(0)	Codec Initialisierungs-Tabelle
0	0	1	C_W8731_ANALOG_BYPASS
1	0	1	C_W8731_ANALOG_MUTE_LEFT
0	1	1	C_W8731_ANALOG_MUTE_RIGHT
1	1	1	C_W8731_ANALOG_MUTE_BOTH
x	x	0	C_W8731_ADC_DAC_0DB_48K

Codec Control Automat (FSM)

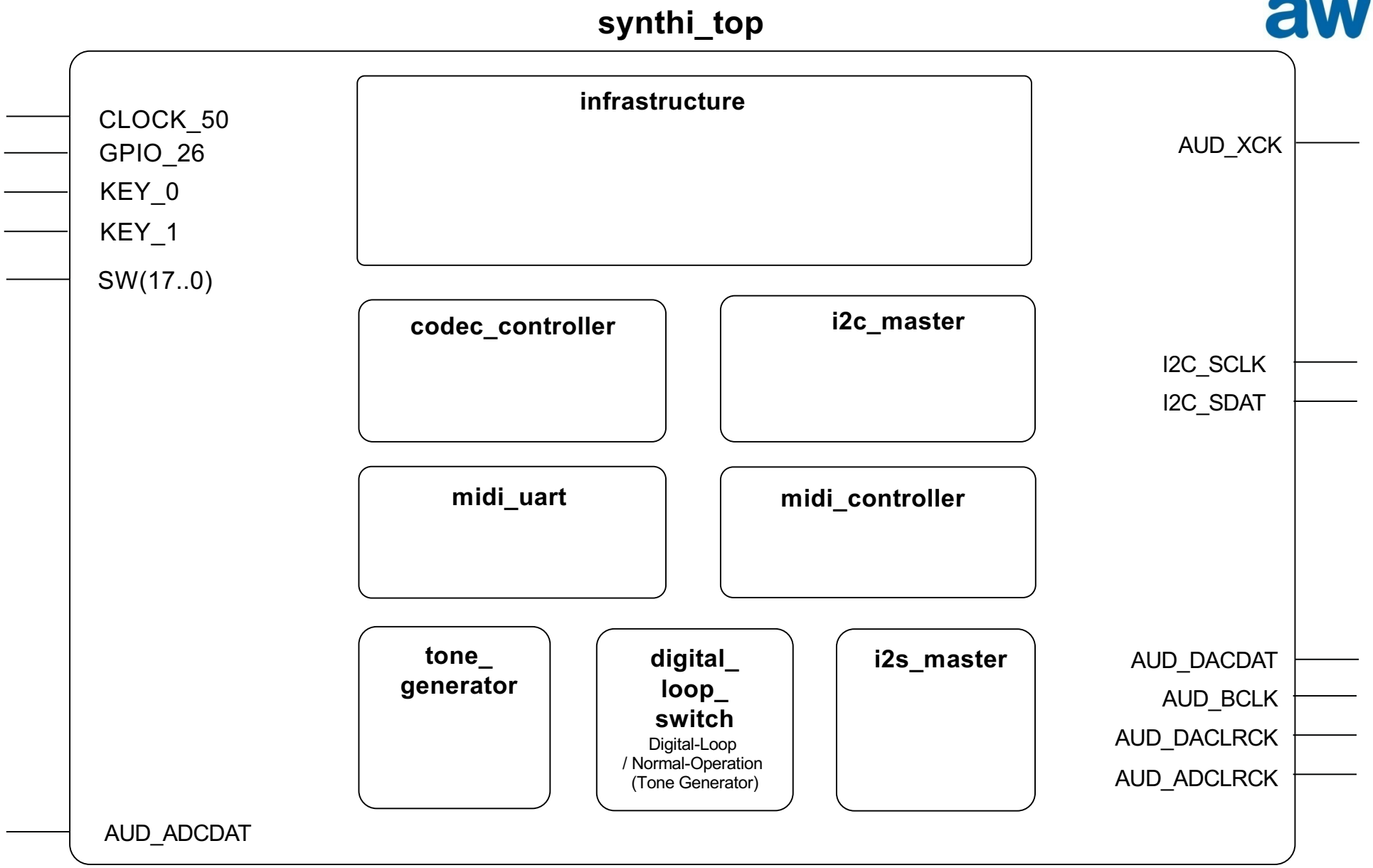


Codec Control Interface

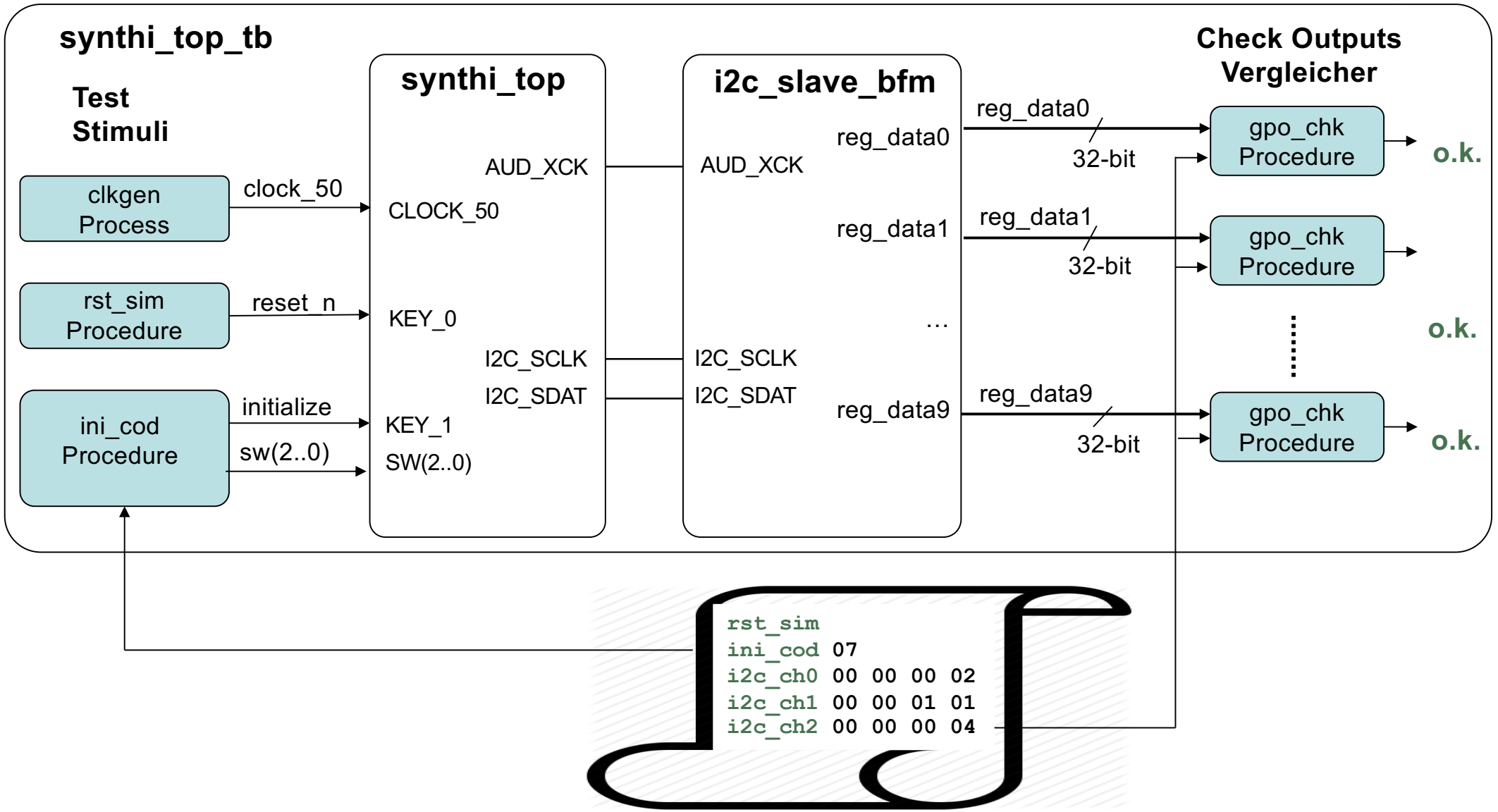
Timing zwischen codec_controller and i2c_master Block



Synthi Top

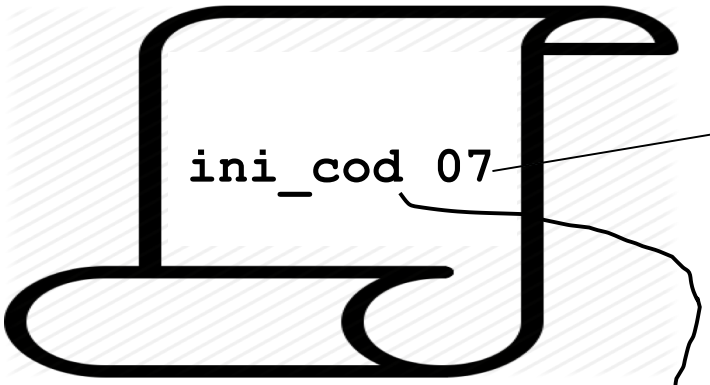

synthi_top
CLOCK_50
GPIO_26
KEY_0
KEY_1
SW(17..0)
AUD_ADCDAT
infrastructure
codec_controller
i2c_master
midi_uart
midi_controller
tone_
generator
digital_
loop_
switch
Digital-Loop
/ Normal-Operation
(Tone Generator)
i2s_master
AUD_XCK
I2C_SCLK
I2C_SDAT
AUD_DACDAT
AUD_BCLK
AUD_DACLRCK
AUD_ADCLRCK

Testbench Ausbau für i2c



Test Script: testcase.dat

Stimulating the Codec Controller



Bits (2..0) von Argument 1
definieren sw(2..0)

Test Vector Script: testcase.dat

synthi_top_tb

```
ini_cod(tv, SW(2 downto 0), KEY_1);
```

synthi_top

codec_controller

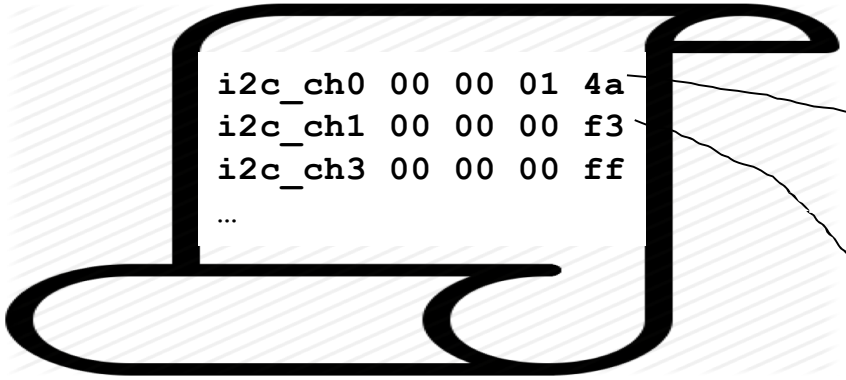
KEY_1 → initialize

SW(2 downto 0) → sw_sync_i

write_o

write_data

Checking the Output of the I2C_slave_bfm



Vergleichswert = 4 x Byte = 32-bit von denen nur die 9 LSb verwendet werden

Test Vector Script: testcase.dat

synthi_top_tb

```
elsif cmd = string'("i2c_ch0") then
    gpo_chk(tv, reg_data0);
elsif cmd = string'("i2c_ch1") then
    gpo_chk(tv, reg_data1);
```

I2c_slave_bfm

I2C_SCLK
I2C_SDAT

reg_data0
reg_data1
reg_data2
...
reg_data9

```
signal reg_data0 : std_logic_vector(31 downto 0);
signal reg_data1 : std_logic_vector(31 downto 0);
signal reg_data2 : std_logic_vector(31 downto 0);
```

Zusätzliche
Testbench
bei Bedarf