Zürcher Hochschule
für Angewandte Wissenschaften

Milestone 2

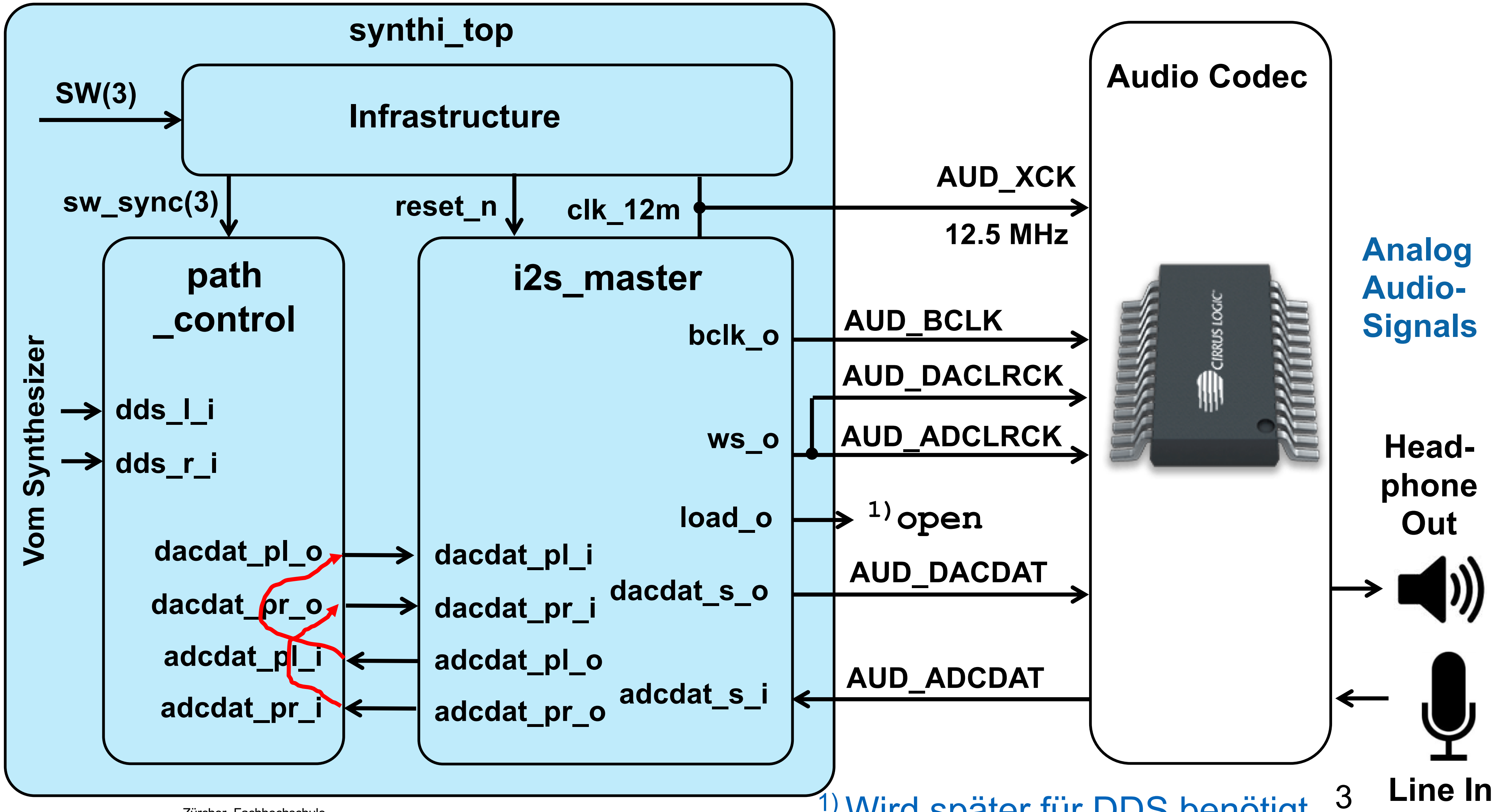
Milestone 2

Codec Digital Audio Interface

- I2S Protocol
- Block Aufteilung für Audio Interface Driver
  - I2S_Master Block
  - Path Control
  - I2S Master Design
- Subblocks und Zeitverlaufsdiagramm

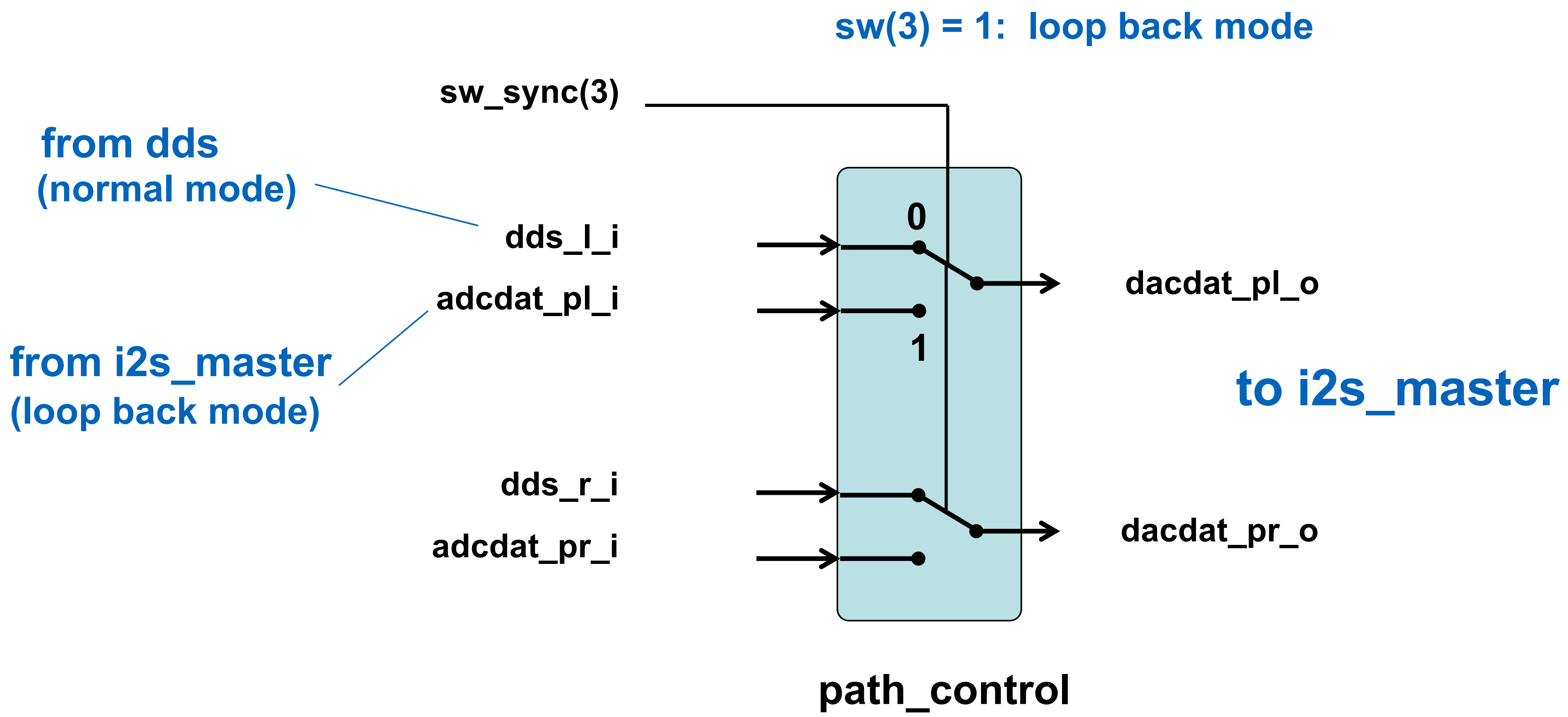
Ausbau bis Milestone-2

Digital Audio-Loop Test



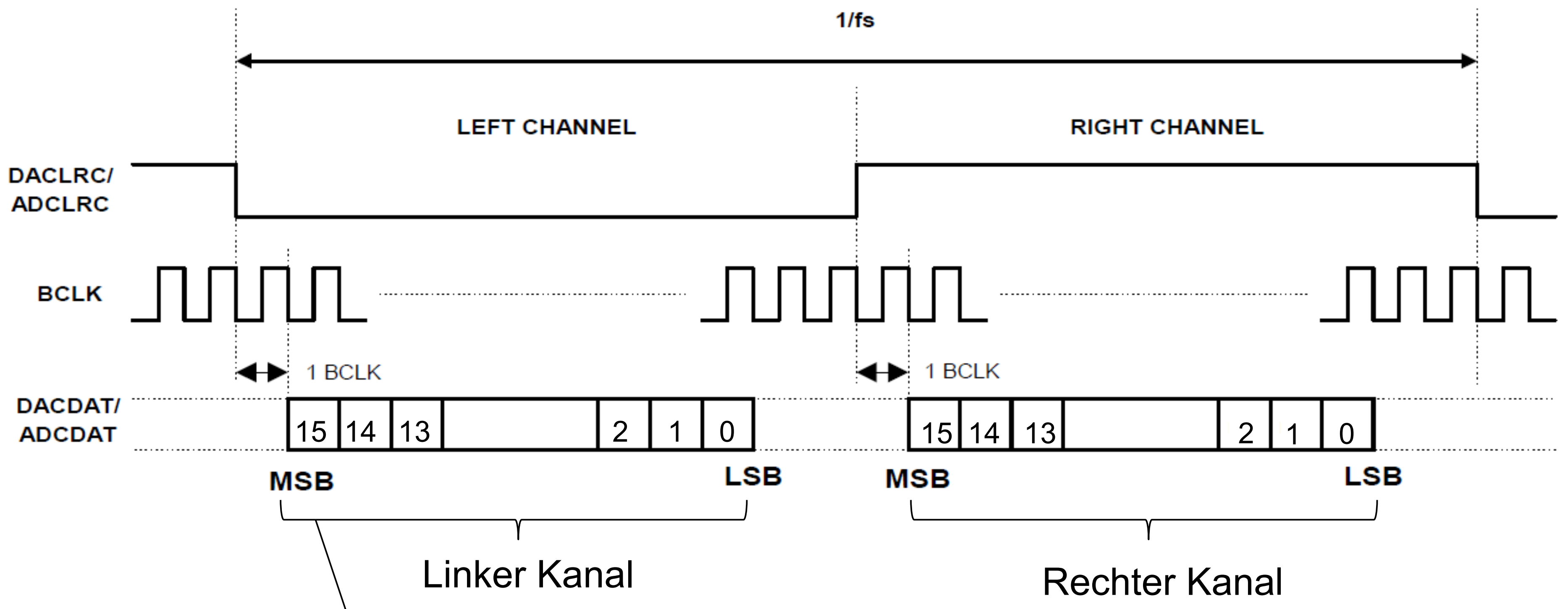
1) Wird später für DDS benötigt

Path Control



Zürcher Hochschule
für Angewandte Wissenschaften

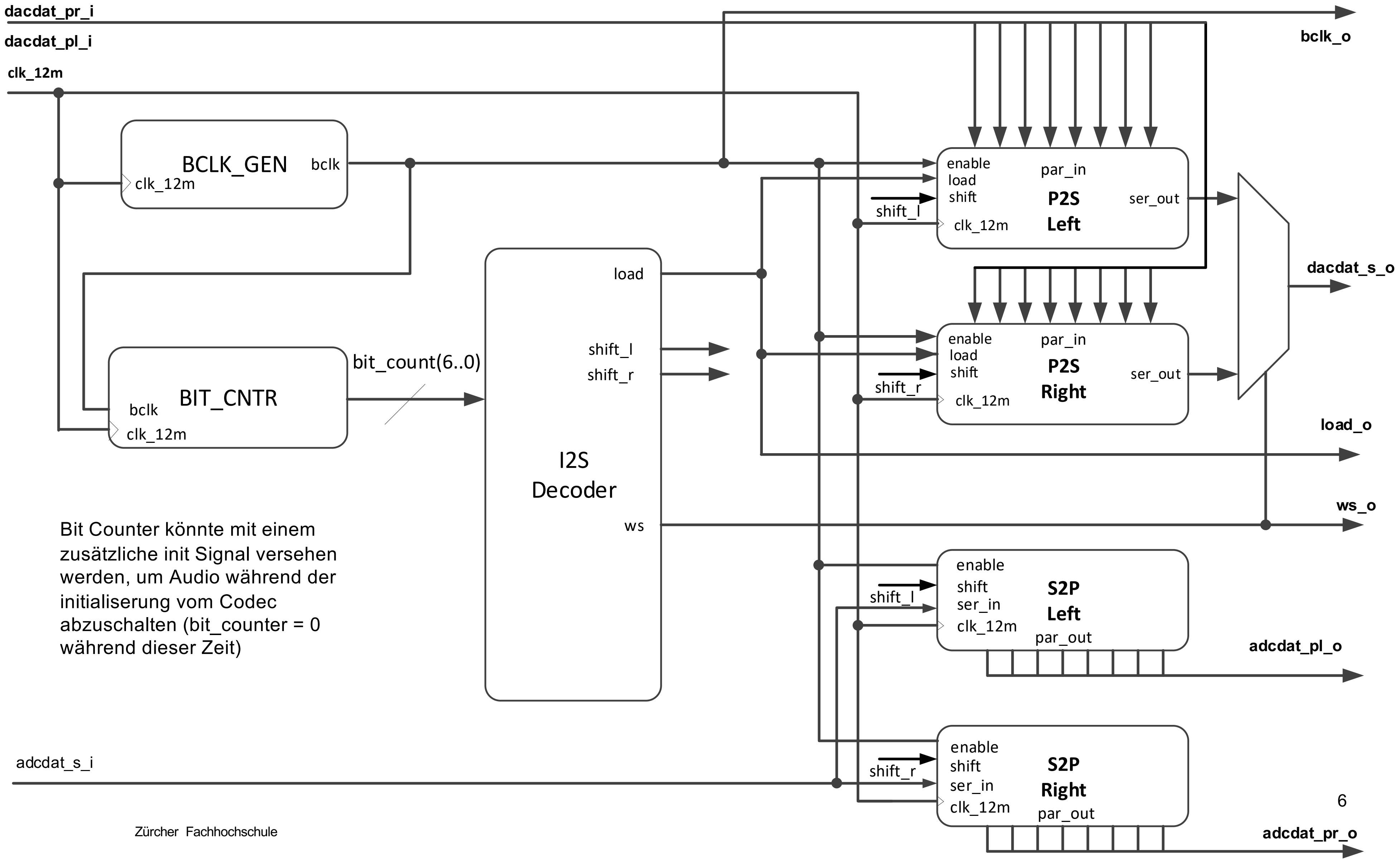
I2S Digital Audio Interface Protocol



Das MSB ist verfügbar eine BCLK Periode nach der fallenden Flanke von DACLRC bzw. ADCLRC (WS)

Quelle: WM8731L Datasheet Rev4.3

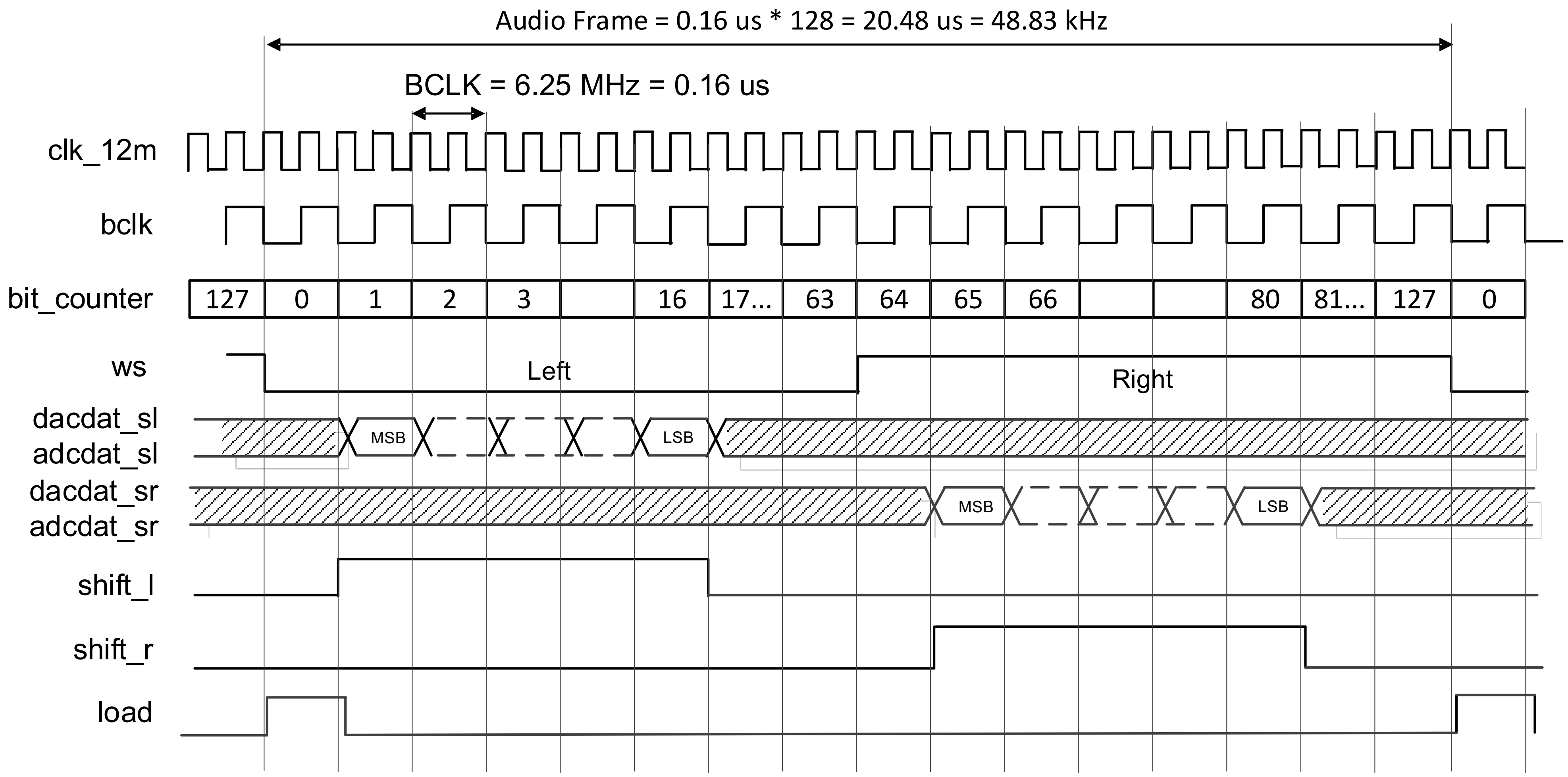
I2S Master Blockdiagram



Bit Counter könnte mit einem zusätzliche init Signal versehen werden, um Audio während der initialiserung vom Codec abzuschalten (bit_counter = 0 während dieser Zeit)

Audio Frame

Word Select: WS = 48kHz = BCLK/128



i2s Decoder

Counter	**0**[1]	**1-16**	**17-63**	**64**	**65-80**	**81-127**
Action	load	shift_l	hold_l		shift_r	hold_r
SIGNALS						
ws	0 (left)			1 (right)		
shift_l	0	1	0	0	0	0
shift_r	0	0	0	0	1	0
load	1	0	0	0	0	0

[1] Bei Zähler Reset, wird sofort das *load* Signal aktiviert und somit ein gültiges Signal in das Schieberegister geladen
(Vermeidung von Knacken nach Reset)

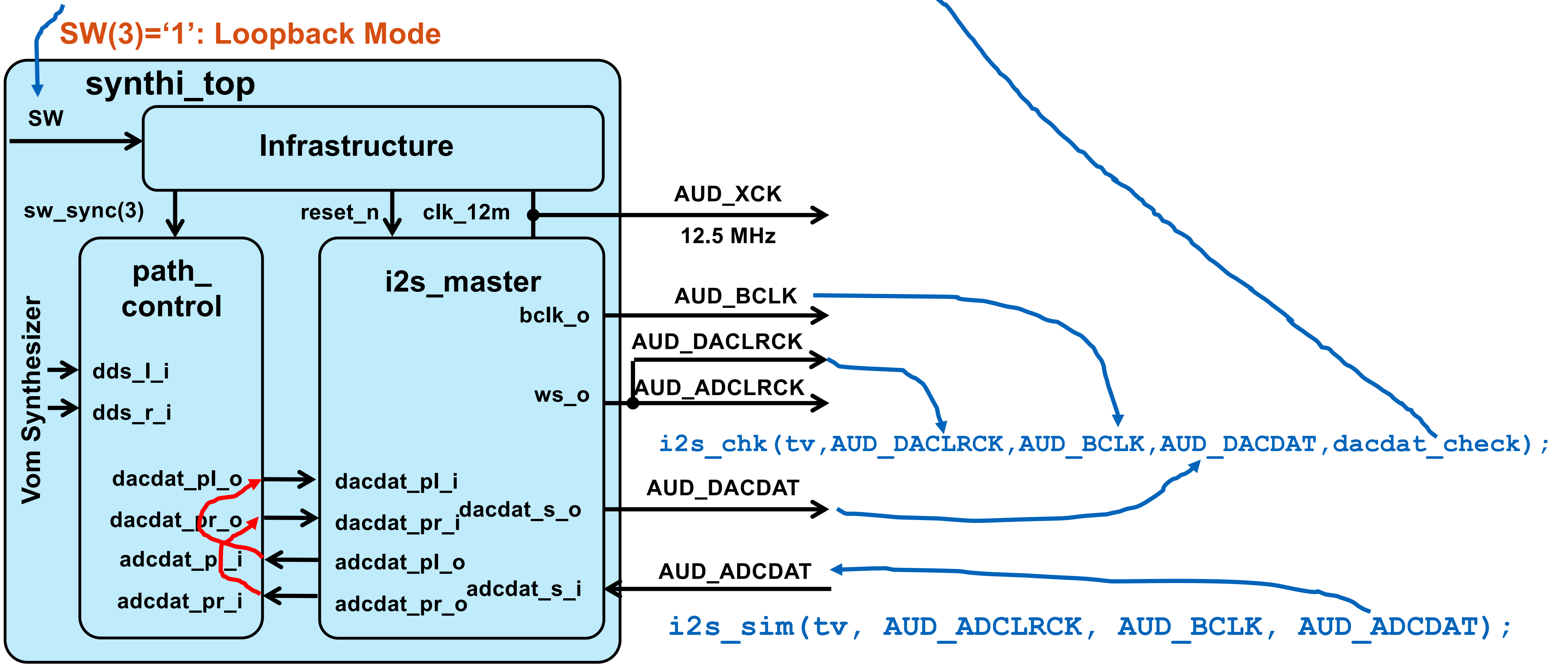
Digital Loop Testbench

```
signal switch : std_logic_vector(31 downto 0);

gpi_sim(tv, switch);

SW(3) <= switch(3);
```

SW(3)='1': Loopback Mode

Observationssignal:

```
signal dacdat_check : std_logic_vector(31 downto 0);
```



```
i2s_chk(tv,AUD_DACLRCK,AUD_BCLK,AUD_DACDAT,dacdat_check);

i2s_sim(tv, AUD_ADCLRCK, AUD_BCLK, AUD_ADCDAT);
```

Bespiel Testprotokoll

```
rst_sim
ini_cod 00
gpi_sim 00 00 00 08
i2s_sim 11 22 33 44
i2s_chk 11 22 33 44
```

Zürcher Hochschule
für Angewandte Wissenschaften

I2s_sim Procedure

Left channel

```
procedure i2s_sim
    (
      variable tv     : inout test_vect;
      signal ws       : in    std_logic;
      signal bclk     : in    std_logic;
      signal acdat_s  : out   std_logic
      ) is

    variable line_out : line;
  begin
    acdat_s <= 'X';
    wait until ws = '0';
    wait until bclk = '1';
    wait until bclk = '0';
    --beginn of left channel transmit
    i2sloop1 : for i in 7 downto 0 loop
      acdat_s <= tv.arg1(i);
      wait until bclk = '1';
      wait until bclk = '0';
    end loop i2sloop1;

    i2sloop2 : for i in 7 downto 0 loop
      acdat_s <= tv.arg2(i);
      wait until bclk = '1';
      wait until bclk = '0';
    end loop i2sloop2;
    acdat_s <= 'X';
```

Right channel

```
wait until ws = '1';
wait until bclk = '1';
wait until bclk = '0';
i2sloop3 : for i in 7 downto 0 loop
  acdat_s <= tv.arg3(i);
  wait until bclk = '1';
  wait until bclk = '0';
end loop i2sloop3;

i2sloop4 : for i in 7 downto 0 loop
  acdat_s <= tv.arg4(i);
  wait until bclk = '1';
  wait until bclk = '0';
end loop i2sloop4;
acdat_s <= 'X';
wait until bclk = '1';
wait until bclk = '0';

hwrite(line_out, tv.arg1);
hwrite(line_out, tv.arg2);
hwrite(line_out, tv.arg3);
hwrite(line_out, tv.arg4);
write(line_out, string'(" WRITTEN TO I2S BUS"));
writeline(OUTPUT, line_out);
```

I2S_chk Procedure

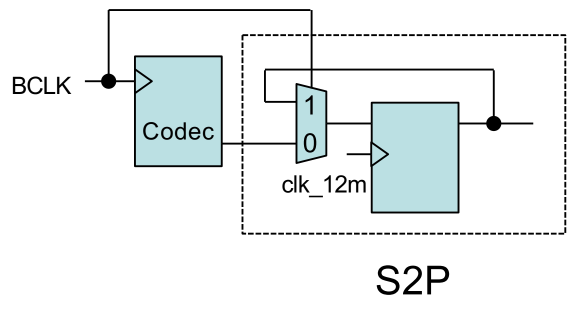
```
 procedure i2s_chk
   (variable tv     : inout test_vect;
     signal ws       : in    std_logic;
     signal bclk     : in    std_logic;
     signal dacdat_s : in    std_logic;
     signal obs_data : out   std_logic_vector(31 downto 0) )
is
   variable line_out     : line;
   variable dacdat_reg_l : std_logic_vector(15 downto 0);
   variable dacdat_reg_r : std_logic_vector(15 downto 0);
  begin
   wait until ws = '0';
   wait until bclk = '1';
   wait until bclk = '0';
   wait until bclk = '1';
   for abit in 15 downto 0 loop
     dacdat_reg_l(abit)     := dacdat_s;
     obs_data(31 downto 16) <= dacdat_reg_l;
     wait until bclk = '0’; wait until bclk = '1';
   end loop;  -- abit

   wait until ws = '1';
   wait until bclk = '1';
   wait until bclk = '0';
   wait until bclk = '1';
   for abit in 15 downto 0 loop
     dacdat_reg_r(abit)    := dacdat_s;
     obs_data(15 downto 0) <= dacdat_reg_r;
     wait until bclk = '0’;  wait until bclk = '1';
   end loop;
```

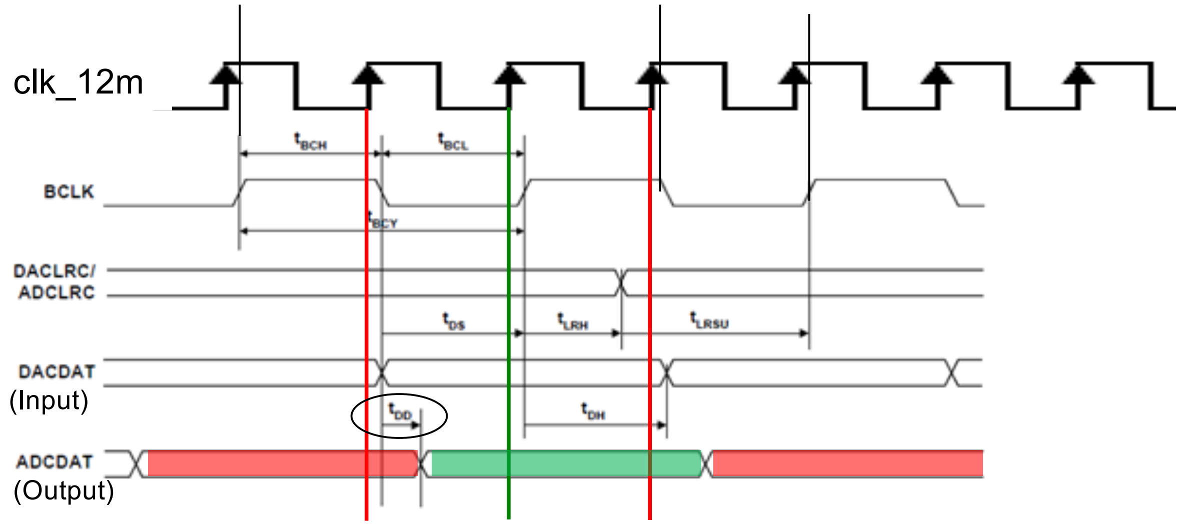
```
if (
     (tv.arg1(7 downto 0) = dacdat_reg_l(15 downto 8)) and
     (tv.arg2(7 downto 0) = dacdat_reg_l(7 downto 0)) and
     (tv.arg3(7 downto 0) = dacdat_reg_r(15 downto 8)) and
     (tv.arg4(7 downto 0) = dacdat_reg_r(7 downto 0))
     ) then
     tv.fail_flag := false;
     write(line_out, string'("AUDIO DATA RECEIVED BY CODEC, O.K. "));
     hwrite(line_out, tv.arg1);
     hwrite(line_out, tv.arg2);
     hwrite(line_out, tv.arg3);
     hwrite(line_out, tv.arg4);
     writeline(OUTPUT, line_out);      -- write the message
   --writeline(outfile,line_out);
   else
     tv.fail_flag := true;
     write(line_out, string'("ERROR:CODEC RECEIVED WRONG AUDIO DATA "));
     hwrite(line_out, dacdat_reg_l & dacdat_reg_r);
     write(line_out, string'(" EXPECTED "));
     hwrite(line_out, tv.arg1);
     hwrite(line_out, tv.arg2);
     hwrite(line_out, tv.arg3);
     hwrite(line_out, tv.arg4);
     writeline(OUTPUT, line_out);
   --writeline(outfile,line_out);
   end if;


  end procedure i2s_chk;
```

Timing Serial to Parallel Converter

S2P Schieberegister muss mit BCLK=0 schieben

Der Codec gibt Daten bei der fallenden Flanke von BCLK aus. Würde das FPGA die Daten an der gleichen Flanke übernehmen, würde das FPGA diese auf Grund von t_{DD} noch nicht sampeln





Test Conditions

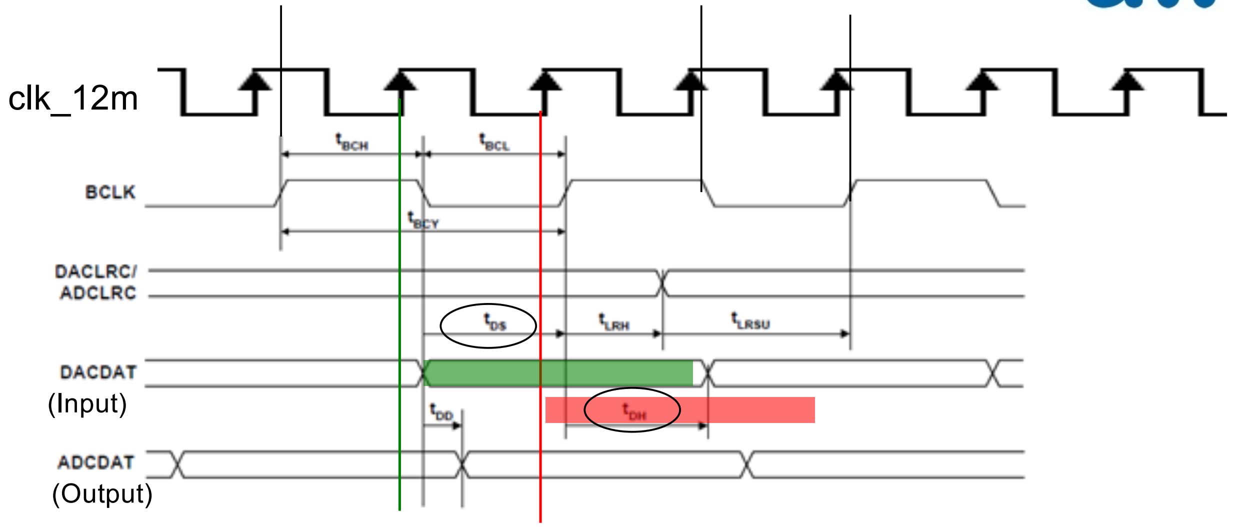
AVDD, HPVDD, DBVDD = 3.3V, AGND = 0V, DCVDD = 1.5V, DGND = 0V, T_A = +25°C, Slave Mode, fs = 48kHz, XTI/MCLK = 256fs unless otherwise stated.

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Audio Data Input Timing Information						
BCLK cycle time	t_{BCY}		50			ns
BCLK pulse width high	t_{BCH}		20			ns
BCLK pulse width low	t_{BCL}		20			ns
DACLRC/ADCLRC set-up time to BCLK rising edge	t_{LRSU}		10			ns
DACLRC/ADCLRC hold time from BCLK rising edge	t_{LRH}		10			ns
DACDAT set-up time to BCLK rising edge	t_{DS}		10			ns
DACDAT hold time from BCLK rising edge	t_{DH}		10			ns

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
ADCDAT propagation delay from BCLK falling edge	t_{DD}		0		35	ns

Zürcher Hochschule
für Angewandte Wissenschaften

Timing Parallel to Serial Converter



P2S Schieberegister muss mit BCLK=1 schieben

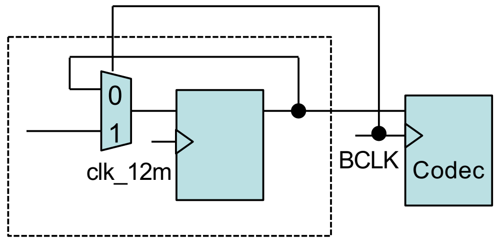
Codec übernimmt Daten bei steigender Flanke von BCLK. Die Setupzeit könnte evtl. nicht eingehalten werden, wenn die Daten erst bei der steigenden Flanke von BCLK kommen

Test Conditions

AVDD, HPVDD, DBVDD = 3.3V, AGND = 0V, DCVDD = 1.5V, DGND = 0V, T_A = +25°C, Slave Mode, fs = 48kHz, XTI/MCLK = 256fs unless otherwise stated.

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Audio Data Input Timing Information						
BCLK cycle time	t_{BCY}		50			ns
BCLK pulse width high	t_{BCH}		20			ns
BCLK pulse width low	t_{BCL}		20			ns
DACLRC/ADCLRC set-up time to BCLK rising edge	t_{LRSU}		10			ns
DACLRC/ADCLRC hold time from BCLK rising edge	t_{LRH}		10			ns
DACDAT set-up time to BCLK rising edge	t_{DS}		10			ns
DACDAT hold time from BCLK rising edge	t_{DH}		10			ns

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
ADCDAT propagation delay from BCLK falling edge	t_{DD}		0		35	ns



P2S