DF1 (Alpha v1.0.0)
FPGA based digital audio oversampling filter

The DF1 is the 4 to 32-times oversampling (interpolation), 2-channel, linear-phase FIR, multi-function digital filter for digital audio and instrumental reproduction equipment. The input/output interface supports input data in16 to 32-bit words with sample rate up to 768kHz, and output data in 16/18/20/22/24-bit words in either 4-times, 8-times, 16-times or 32-times oversampling selectable output mode with output sample rate up to 1536kHz.

Device has 55db Nyquist frequency rejection in x32 oversampling mode for 44.1/48kHz input sample rate with < 0.00001db pass band rejection. Also it has rich functionality to work with any DAC: parallel audio, industrial SPI or audio sigma-delta converters.

To prevent digital clipping DF1 has selectable input data attenuation with clipping indication output.

The internal system clock operates at either 768Fs or 1024Fs selectable speed (where Fs is the audio sampling frequency44.1/48kHz). Plus, operating system clock frequency can be decreased to 192Fs or 256Fs with decreasing filtering performance and maximum oversampling ratio.

Features:

■ Left/right-channel (2-channel processing)

■ 4-times to 32-times oversampling (interpolation)
- 32-times interpolation filter
- 5-stage linear-phase FIR configuration
  - 1st stage (4 modes):
    - 253-tap (with 60dBNyquist frequency rejection)
    - 175-tap (half-band)
    - 79-tap (half-band)
    - 47-tap (half-band)
  - 2nd stage: 31-tap (half-band)
  - 3rd stage: 15-tap (half-band)
  - 4rd stage: 11-tap (half-band)
  - 5rd stage (2 modes)
    - 7-tap (half-band)
    - 11-tap (half-band)
- < ±0.00001 dB passband ripple in high ripple mode
- > 120dB stop-band attenuation

■ Automatic oversampling ratio correction

■ Automatic bypass mode

■ Overflow processing & indication

■ Soft muting

■ Input attenuator to prevent digital signal clipping
- 4 selectable attenuate levels
  - 0dB
  - 1bB
  - 2dB
  - 3dB

■ Input data format
- 2s complement, MSB first
- 4 selectable formats
  - Philips I2S
  - Left-justified
  - Right-justified 24 bit
  - Right-justified 16 bit
- selectable frame length:
  - 32, 48 or 64 bit

■ Output data format
- 2s complement or offset binary, MSB first, LR simultaneous
- 16/18/20/22/24-bit serial
- 6 output data modes
  - SPI differential mode
  - SPI series differential mode
  - Double-pipeline mode
  - SPI series-pipeline mode
  - SPI series-channels mode
  - RJ24 continues BCK mode
- 6 output datarate modes
  - bck speed clk/1
  - bck speed clk/2
  - bck speed clk/3
  - bck speed clk/4
  - bck speed clk/6
  - bck speed clk/8
- Selectable deglitching signal options
- Selectable invertionbck and deglitching signals

■ 27-bit coefficients

■ 30-bit internal data word length

■ 32 × 27-bit parallel multiplier/37-bit accumulator

■ Dither round-off processing with noise shaping

■ Selectable system clock
- 768fs/1024fs selectable
- Maximum operating frequency49.152 MHz max

■ 100-pin plastic QFP

■ Intel/Altera FPGA Cyclone EP1C3 based, with 0.13u CMOS process

■ Lattice FPGA LCMXO2-2000 based, with 0.065u Flash CMOS process

■ Flexible architecture to porting on other FPGA

Minsk 2021 by Aleksey Fisko

Pins description for EP1C3T100 FPGA

FIR functional pins table

Number	Direction	Name	Functional description
1	Input	sai_sck	Serial audio system clock
2	Input	sai_dat	Serial audio data input
3	Input	sai_lrc	Channel frame clock
4	Input	oscsel	select OSC (internal pull up 25k with 3.3V supply typical) 0 - osc1ena 1 - osc2ena
5	Input	mute	1 - force to mute state (internal pull up 25k with 3.3V supply typical)
20	Input	lp_mode	FIR ripple control 0 - high ripple mode (maximum FIR performance) 1 - low power mode (minimum power consumption)
21	Input	spi_mode[2]	Serial output data mode { spi_mode [2], spi_mode [1], spi_mode [0] }: 000: differential data mode 001: series differential data mode (for dual SPI dac only) 010: pipeline data mode 011: series pipeline mode (for dual SPI dac only) 100: series channels data mode (for dual SPI dac only) 101: RJ24 mode with continue bck Other: differential data mode
22	Input	spi_mode[1]	
23	Input	spi_mode[0]	
24	Input	head_mode[2]	Head mode for SPI DAC { head _mode [2], head _mode [1], head _mode [0] }: 000: header none (for SPI DAC without header field in data frame) 001: DAC8812 header mode (two 2-bit headers with values: 0x1 - ch1, 0x2 - ch2) 010: AD5791 header mode (one 4-bit header with value 0x1) 011: AD5063 header mode (one 8-bit header with value 0x00) 100: DAC11001 header mode (one 8-bit header with value 0x01) 101: LTC1592 header mode (one 8-bit header with value 0x20) 110: LTC2752 header mode (two 8-bit headers with values: 0x30 - ch1, 0x32 - ch2) 111: reserved (header none)
25	Input	head_mode[1]	
26	Input	head_mode[0]	
27	Input	d_width[2]	Resolution of output data { d_width [2], d_width [1], d_width [0] }: 000: 16-bit 001: 18-bit 010: 20-bit 011: 22-bit 100: 24-bit Other: 24-bit
28	Input	d_width[1]	
29	Input	d_width[0]	
34	Input	spi_dsize[1]	Data field size in SPI packet (left justified) { spi_dsize [1], spi_dsize [0] }: 00: size = data resolution (d_width) 01: 16-bit 10: 20-bit 11: 24-bit
35	Input	spi_dsize[0]	
36	Input	d_speed[2]	Output bck clock speed (output data rate) { d_ speed [2], d_ speed [1], d_ speed [0] }: 000: clk/1 001: clk/2 010: clk/3 011: clk/4 100: clk/6 101: clk/8 Other: undefined (must not be used!).
37	Input	d_speed[1]	
38	Input	d_speed[0]	
39	Input	ovs_max[1]	Maximum oversampling ratio: { ovs_max [1], ovs_max [0] }: 00: x4 01: x8 10: x16 11: x32
40	Input	ovs_max[0]	
41	Input	dgl_mode[1]	Deglitching circuit hold-state transition mode { dgl_mode [1], dgl_mode [0] }: Case spi_dsize = 0 (audio dac mode): 00: transition on rising edge wck 01: transition on falling edge wck 10: transition on rasing edge 2-s bck clock (PCM1704 mode) 11: transition on rasing edge 4-s bck clock (PCM1702 mode) Case spi_dsize > 0 (spi dac mode): 00: transition on rising edge wck (synchronous DAC update) 01: transition on falling edge lutch (asynchronous DAC update) 10: transition on falling edge wck (not used for spi DAC) 11: transition on falling edge wck (not used for spi DAC)
42	Input	dgl_mode[0]	
47	Input	dgh_mode[1]	Deglitching circuit sample-state transition mode { dgh_mode [1], dgh_mode [0] }: 00: transition on 50% duty 01: transition on 53% duty 10: transition on 56% duty 11: transition on 59% duty
48	Input	dgh_mode[0]	
49	Input	dg_inv	1 - invert deglitching signal
50	Input	dg_pp	1 - force deglitching signal in pipeline mode. In this mode deglitching OpAmp has not hold state, but it switch between two pipeline DAC
51	Input	dg_off	1 - force deglitching signal to sample state

52	Input	outZ	1 - force outputs series data, bck and wck to Z-state. This option may be used by external MCU to configuring SPI DAC
53	Output	ltch2	Asynchronous update signal2 for SPI DAC
54	Output	wck2	Word clock2 for serial data output
55	Output	bck2	Bit-clock2 for serial data output
56	Output	dat2_right	Right channel serial data2 output
57	Output	dat2_left	Left channel serial data2 output
65	Output	dg	Deglitching signal output
66	Input	clk	Master clock input (main global clock)
68	Output	osc2ena	Enable signal for oscillator in 48kHz or 44.1kHz frequency domain
69	Output	osc1ena	Enable signal for oscillator in 44.1kHz or 48kHz frequency domain
70	Output	ltch1	Async update signal1 for SPI DAC
71	Output	wck1	Word clock1 for serial data output
72	Output	bck1	Bit-clock1 for serial data output
73	Output	dat1_right	Right channel serial data1 output
74	Output	dat1_left	Left channel serial data1 output
75	Input	outInv	1 - invert SPI data output: dat_left, data_right, wck, lutch
76	Input	bck_inv	Invert bck signal 0 - output data change on falling edge bck 1 - output data chance on rising edge bck
77	Input	ob_ntwc	Select output data formatting: 0 - 2's complement, 1 - offset binary
78	Input	mck_mode	Select clock domain for master clock input: 0 - 1024Fs 1 - 768Fs
84	Output	clipping	Active low indicating signal: goes LOW if internal data bus for digital data processing is overflow, else goes HIGH
85	Output	ovs_ind[2]	Active low bus for input sampling rate indicate { ovs_ind [2], ovs_ind [1], ovs_ind [0] }: 111: input Fs = 44.1/48kHz 110: input Fs = 88.2/96kHz 101: input Fs = 176.4/192kHz 100: input Fs = 352.8/384kHz 011: input Fs = 705.6/768kHz 000: input Fs >= output Fs (bypass mode)
86	Output	ovs_ind[1]	
87	Output	ovs_ind[0]	
88	Input	clrn	0 - global reset state
89	Input	att[1]	Input signal attenuate level { att [1], att [0] }: 00: 0db 01: -1db 10: -2db 11: -3db
90	Input	att[0]	
91	Input	format[1]	Select serial audio input format { format [1], format [0] }: 00: I2S 01: LJ 10: RJ24 11: RJ16
92	Input	format[0]	
97	Input	frame[1]	Select number bits for serial audio input frame { frame [1], frame [0] }: 00: 64-bit frame 01: 48-bit frame 10: 32-bit frame 11: 64-bit frame
98	Input	frame[0]	
99	Input	mck_div	Turn on divide on two for clock output (pin 100): 0 - Fmcko = Fclk 1 - Fmcko = Fclk/2
100	Output	mcko	Master clock output for audio data source synchronization

Power supply pins table

Number	Direction	Name	Functional description
9	Power	VCCA	PLL supply: +1.5V (connect to core supply bus)
18	Power	VCC IO	Input/output logic supply: +3.3V
31	Power	VCC IO	
46	Power	VCC IO	
59	Power	VCC IO	
80	Power	VCC IO	
95	Power	VCC IO	
33	Power	VCC CORE	Internal core supply: +1.5V
44	Power	VCC CORE	
82	Power	VCC CORE	
93	Power	VCC CORE	
11	Power	GND	Device power ground
19	Power	GND	
30	Power	GND	
32	Power	GND	
43	Power	GND	
45	Power	GND	
58	Power	GND	
81	Power	GND	
83	Power	GND	

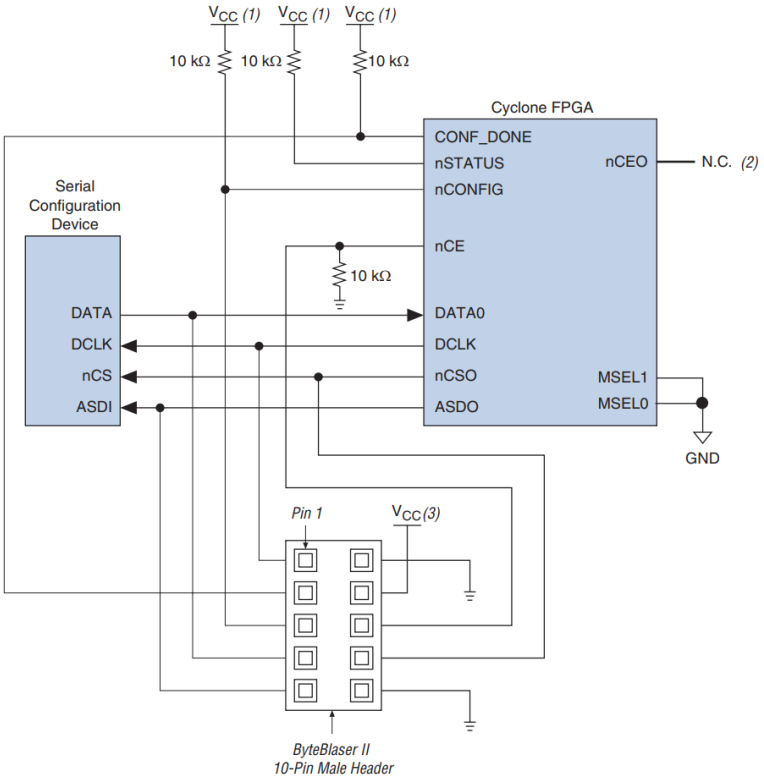
94	Power	GND
96	Power	GND

Dedicated programming pins table

Number	Direction	Name	Functional description
6	Output	nCSO	Chip select output
7	Input	DATA	Serial data input
8	Input	nCONFIG	Configuring process
12	Output	nCEO	
13	Input	nCE	
14	Input	MSEL0	Configuring mode select. Must be connected to GND
15	Input	MSEL1	
16	Output	DCLK	Serial data clock output
17	Output	ASDO	Serial data output
60	Output	CONFIG_DONE	Configuring done indicating output
61	Output	nSTATUS	Status indicating output
62	Input	JTAG_TCK	JTAG serial data clock (must be tied to ground)
63	Input	JTAG_TMS	JTAG serial frame clock
64	Output	JTAG_TDO	JTAG serial data output
67	Input	JTAG_TDI	JTAG serial data input

Configuration of cyclone FPGA require external flash SPI memory (Altera/Intel vendor only!), for example EPCS1. Configuring Cyclone device from external flash is performed in active serial (AS) mode. To indicate configuration process - connect LEDs to CONF_DONE pin.

Figure 13–9. In-System Programming of Serial Configuration Devices



Notes to Figure 13–9:

(1) Connect these pull-up resistors to 3.3-V supply.

(2) The nCEO pin is left unconnected.

(3) Power up the ByteBlaster II cable's V_{CC} with a 3.3-V supply.

Pins description for LCMXO2-2000xx-1TG100 FPGA

FIR functional pins table

Number	Direction	Name	Functional description
1	Input	mck_div	Turn on divide on two for clock output (pin 100): 0 - Fmcko = Fclk 1 - Fmcko = Fclk/2
2	Output	mcko	Master clock output for audio data source synchronization
3	Input	sai_sck	Serial audio system clock
4	Input	sai_dat	Serial audio data input
7	Input	sai_lrc	Channel frame clock
8	Input	oscsel	select OSC (internal week pull up) 0 - osc1ena 1 - osc2ena
9	Input	mute	1 - mute state (internal weak pull-up)
20	Input	att[1]	Input signal attenuation level { att [1], att [0] }: 00: 0db 01: -1db 10: -2db 11: -3db
21	Input	att[0]	
25	Input	lp_mode	FIR ripple control 0 - high ripple mode (maximum FIR performance) 1 - low power mode (minimum power consumption)
27	Input	spi_mode[2]	Serial output data mode { spi_mode [2], spi_mode [1], spi_mode [0] }: 000: differential data mode 001: series differential data mode (for dual SPI DAC only) 010: pipeline data mode 011: series pipeline mode (for dual SPI DAC only) 100: series channels data mode (for dual SPI DAC only) 101: RJ24 mode with continue bit clock Other: differential data mode
28	Input	spi_mode[1]	
29	Input	spi_mode[0]	
30	Input	head_mode[2]	Head mode for SPI DAC { head _mode [2], head _mode [1], head _mode [0] }: 000: header none (for SPI DAC without header field in data frame) 001: DAC8812 header mode (two 2-bit headers with values: 0x1 - ch1, 0x2 - ch2) 010: AD5791 header mode (one 4-bit header with value 0x1) 011: AD5063 header mode (one 8-bit header with value 0x00) 100: DAC11001 header mode (one 8-bit header with value 0x01) 101: LTC1592 header mode (one 8-bit header with value 0x20) 110: LTC2752 header mode (two 8-bit headers with values: 0x30 - ch1, 0x32 - ch2) 111: reserved (header none)
31	Input	head_mode[1]	
32	Input	head_mode[0]	
34	Input	d_width[2]	Resolution of output data { d_width [2], d_width [1], d_width [0] }: 000: 16-bit 001: 18-bit 010: 20-bit 011: 22-bit 100: 24-bit Other: 24-bit
35	Input	d_width[1]	
36	Input	d_width[0]	
37	Input	spi_dsize[1]	Data field size in SPI packet (left justified) { spi_dsize [1], spi_dsize [0] }: 00: size = data resolution (d_width) 01: 16-bit 10: 20-bit 11: 24-bit
38	Input	spi_dsize[0]	
39	Input	d_speed[2]	Output bit clock speed (output data rate) { d_ speed [2], d_ speed [1], d_ speed [0] }: 000: clk/1 001: clk/2 010: clk/3 011: clk/4 100: clk/6 101: clk/8 Other: undefined (must not be used!)
40	Input	d_speed[1]	
41	Input	d_speed[0]	
42	Input	ovs_max[1]	Maximum oversampling ratio: { ovs_max [1], ovs_max [0] }: 00: x4 01: x8 10: x16 11: x32
43	Input	ovs_max[0]	
45	Input	dgl_mode[1]	Deglitching circuit hold-state transition mode { dgl_mode [1], dgl_mode [0] }: Case spi_dsize = 0 (audio DAC mode): 00: transition on rising edge WCK 01: transition on falling edge WCK 10: transition on rasing edge 2-s BCK (PCM1704 mode) 11: transition on rasing edge 4-s BCK (PCM1702 mode) Case spi_dsize > 0 (SPI DAC mode): 00: transition on rising edge WCK (synchronous DAC update) 01: transition on falling edge LATCH (asynchronous DAC update) 10: transition on falling edge WCK (not used for SPI DAC) 11: transition on falling edge WCK (not used for SPIDAC)
47	Input	dgl_mode[0]	

48	Input	dgh_mode[1]	Deglitching circuit sample-state transition mode { dgh_mode [1], dgh_mode [0] }: 00: transition on 50% duty 01: transition on 53% duty 10: transition on 56% duty 11: transition on 59% duty
49	Input	dgh_mode[0]	
51	Input	dg_inv	1 - invert deglitching signal
52	Input	dg_pp	1 - force deglitching signal in pipeline mode. In this mode deglitching OpAmp has not hold state, but it switch between two pipeline DACs
53	Input	dg_off	1 - force deglitching signal to sample state
54	Input	outZ	1 - force outputs series DATA, BCK and WCK to Z-state. This option may be used by external MCU to configuring SPI DAC.
57	Output	ltch2	Asynchronous update signal2 for SPI DAC
58	Output	wck2	Word clock2 for serial data output
59	Output	bck2	Bit clock2 for serial data output
60	Output	dat2_right	Right channel serial data2 output
61	Output	dat2_left	Left channel serial data2 output
62	Output	dg	Deglitching signal output
63	Input	clk	Master clock input (main global clock)
64	Output	osc2ena	Enable signal for oscillator in 48kHz/44.1kHz frequency domain
65	Output	osc1ena	Enable signal for oscillator in 44.1kHz/48kHz frequency domain
66	Output	ltch1	Async update signal1 for SPI DAC
67	Output	wck1	Word clock1 for serial data output
68	Output	bck1	Bit-clock1 for serial data output
69	Output	dat1_right	Right channel serial data1 output
70	Output	dat1_left	Left channel serial data1 output
71	Input	outInv	1 - invert SPI data output: data_left, data_right, wck, latch
74	Input	bck_inv	Invert BCK signal 0 - output data change on falling edge BCK 1 - output data change on rising edge BCK
75	Input	ob_ntwc	Select output data format: 0 - 2's complement, 1 - offset binary
76	Input	mck_mode	Select clock domain for master clock input: 0 - 1024Fs 1 - 768Fs
84	Output	clipping	Active low indicating signal: goes LOW if internal data bus for digital data processing is overflow, else – goes HIGH
85	Output	ovs_ind[2]	Active low bus for input sampling rate indicate { ovs_ind [2], ovs_ind [1], ovs_ind [0] }: 111: input Fs = 44.1/48kHz 110: input Fs = 88.2/96kHz 101: input Fs = 176.4/192kHz 100: input Fs = 352.8/384kHz 011: input Fs = 705.6/768kHz 000: input Fs >= output Fs (bypass mode)
86	Output	ovs_ind[1]	
87	Output	ovs_ind[0]	
88	Input	clrn	0 - global reset state
96	Input	format[1]	Select serial audio input format { format [1], format [0] }: 00: I2S 01: LJ 10: RJ24 11: RJ16
97	Input	format[0]	
98	Input	frame[1]	Select number of bits for serial audio input frame { frame [1], frame [0] }: 00: 64-bit frame 01: 48-bit frame 10: 32-bit frame 11: 64-bit frame
99	Input	frame[0]	

Power supply pins table

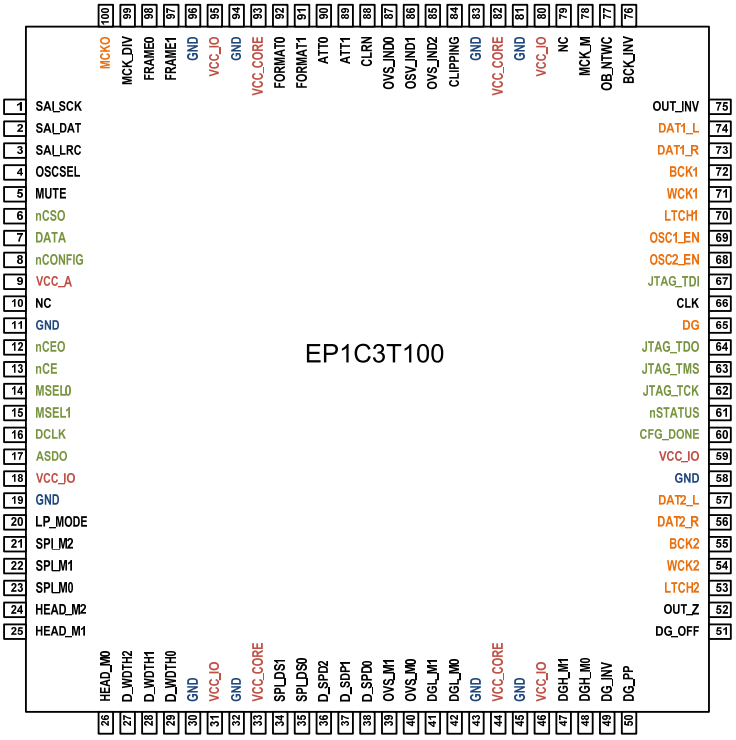
Number	Direction	Name	Functional description
5	Power	VCC IO	Input/output logic supply: +3.3V
11	Power	VCC IO	
23	Power	VCC IO	
26	Power	VCC IO	
46	Power	VCC IO	
55	Power	VCC IO	
73	Power	VCC IO	
80	Power	VCC IO	
93	Power	VCC IO	
50	Power	VCC CORE	Internal core supply: +1.2V (+3.3V for HC version)
100	Power	VCC CORE	
6	Power	GND	Device power ground
22	Power	GND	
33	Power	GND	
44	Power	GND	
56	Power	GND	
72	Power	GND	
79	Power	GND	
92	Power	GND	

Dedicated programming pins table

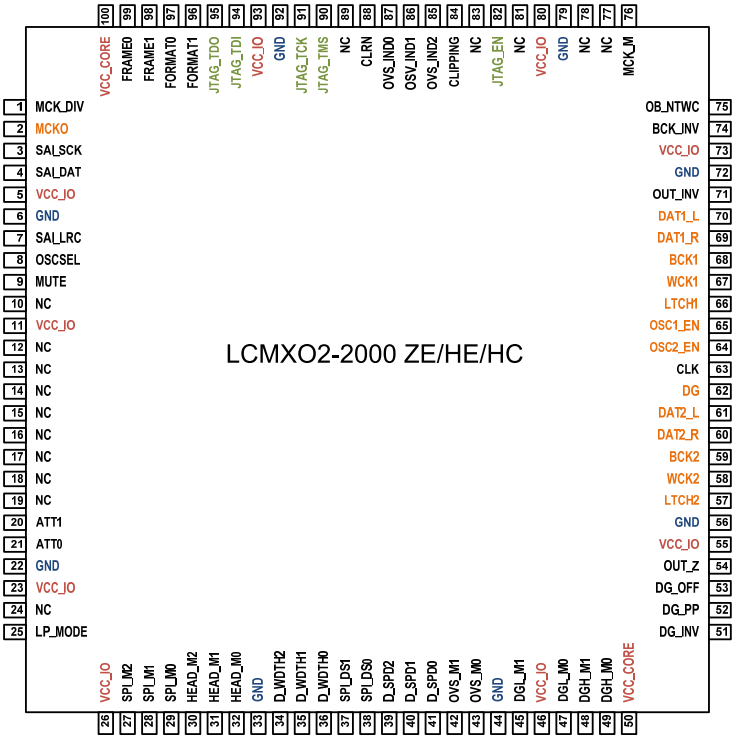
Number	Direction	Name	Functional description
82	Output	JTAG_ENB	Set to 1 enable device JTAG port
90	Input	JTAG_TMS	JTAG serial frame clock
91	Input	JTAG_TCK	JTAG serial data clock
94	Input	JTAG_TDI	JTAG serial data input
95	Output	JTAG_TDO	JTAG serial data output

INPUTS PINS OUTPUTS PINS CONFIG PINS SUPPLY PINS GROUND PINS

Altera Cyclone FPGA DF1 QFP-100 package pinout

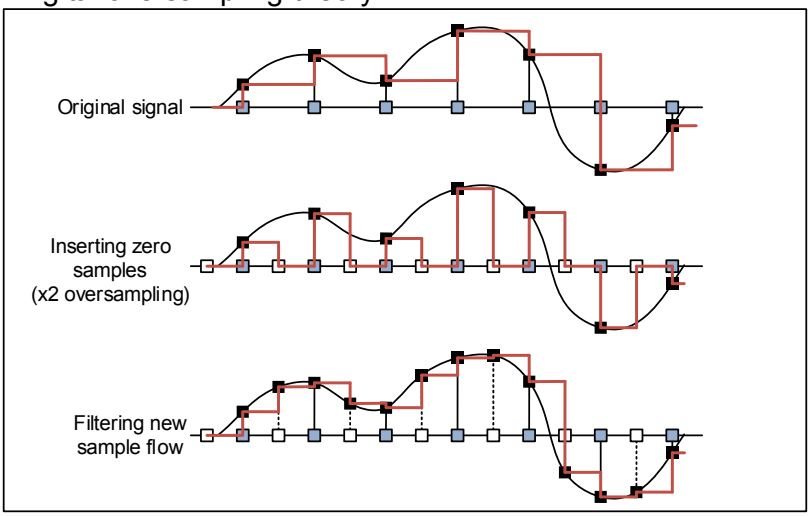


Lattice MachXO2 FPGA DF1 QFP-100 package pinout

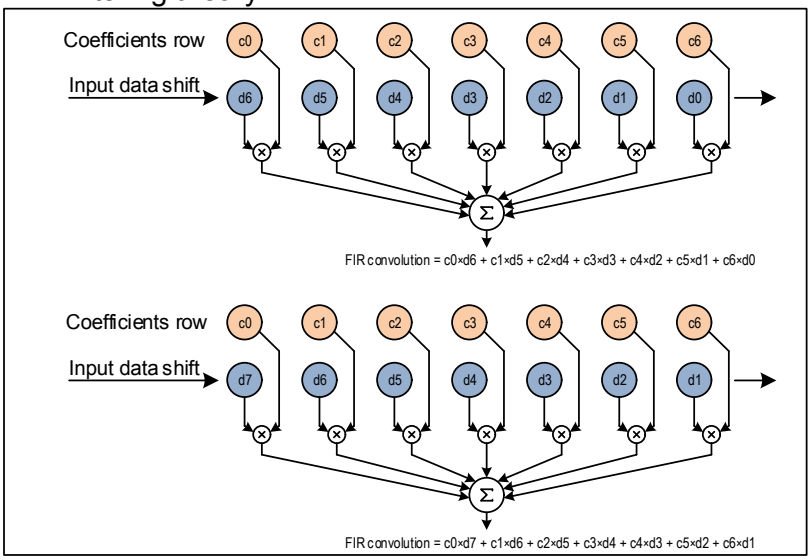


Theory of operation

Digital oversampling theory



FIR filtering theory



Oversampling even order linear phase FIR arithmetic

FIR step Row1

c0	c1	c2	c3	c4	c5	c4	c3	c2	c1	c0
0	d5	0	d4	0	d3	0	d2	0	d1	0	d0	0

First interpolated sample convolution = c0×0 + c1×d4 + c2×0 + c3×d3 + c4×0 + c5×d2 + c4×0 + c3×d1 + c2×0 + c1×d0 + c0×0
Arithmetic optimized form = (d0 + d4)×c1 + (d1 + d3)×c3 + d2×c5

FIR step Row2

c0	c1	c2	c3	c4	c5	c4	c3	c2	c1	c0
d6	0	d5	0	d4	0	d3	0	d2	0	d1	0	d0

Second interpolated sample convolution = c0×d5 + c1×0 + c2×d4 + c3×0 + c4×d3 + c5×0 + c4×d2 + c3×0 + c2×d1 + c1×0 + c0×d0
Arithmetic optimized form = (d0 + d5)×c0 + (d1 + d4)×c2 + (d2 + d3)×c4

Half-Band FIR step Row1

c0	0	c2	0	c4	1	c4	0	c2	0	c0
0	d5	0	d4	0	d3	0	d2	0	d1	0	d0	0

First interpolated sample convolution = c0×0 + 0×d4 + c2×0 + 0×d3 + c4×0 + 1×d2 + c4×0 + 0×d1 + c2×0 + 0×d0 + c0×0
Arithmetic optimized form = d2

Half-Band FIR step Row2

c0	0	c2	0	c4	1	c4	0	c2	0	c0
d6	0	d5	0	d4	0	d3	0	d2	0	d1	0	d0

Second interpolated sample convolution = c0×d5 + 0×0 + c2×d4 + 0×0 + c4×d3 + 1×0 + c4×d2 + 0×0 + c2×d1 + 0×0 + c0×d0
Arithmetic optimized form = (d0 + d5)×c0 + (d1 + d4)×c2 + (d2 + d3)×c4

Functional description

1. Serial audio data input

There are 4 input data modes set by pins **format[1:0]** (I2S, LJ, RJ24 and RJ16) and three modes of frame bit clock set by pins **frame[1:0]**: 64bit, 48bit and 32bit. For more details see figure 1.

To prevent digital clipping the input contains configurable digital attenuator with four attenuation levels that set by pins **att[1:0]** (binary value attenuate level in db: 0: 0bd, 1: -1db, 2: -2db and 3: -3db).

2. Output data rate and coding

There are 6 output data rates set by pins **d_speed[2:0]** that set divide option for output bit clock. The highest speed is set by zero value of **d_speed[]** and in this mode output bit clock frequency is equal input master clock (divide by one). Value 5 of **d_speed[]** sets the lowest speed: divide output clock by 8. For more details see figure 2.

By default BCK mode serial output data updates on falling edge of BCK. Logic 1 in input **bck_inv** inverts BCK signal and sets output data update on rising edge of BCK. For more details see figure 3.

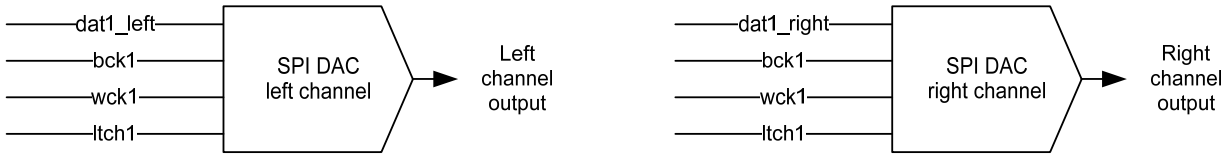
There are two output data coding options: 2's complement (**ob_ntwc** set to 0) and offset binary (**ob_ntwc** set to 1). Serial output diagram in offset binary mode is on figure 4.

3. SPI modes

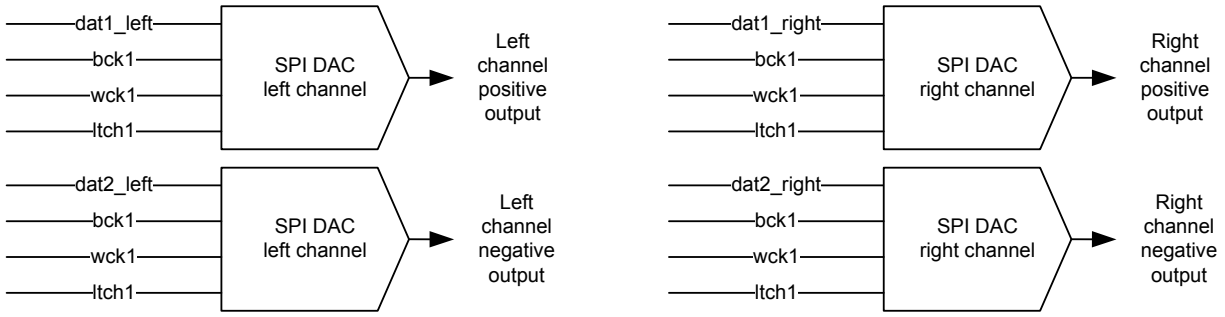
There are 6 output data modes set by pins **spi_mode[2:0]**. For more details see figure 8.

SPI differential mode (spi_mode == 0)

In this mode data output to pins **dat1_left** and **dat1_right**, while **dat2_left** and **dat2_right** output invertion data flow. Other SPI signals (**bck1** & **bck2**, **wck1** & **wck2**, **ltch1** & **ltch2**) are output synchronously from each other. For simple parallel DAC **dat2** outputs are not used:



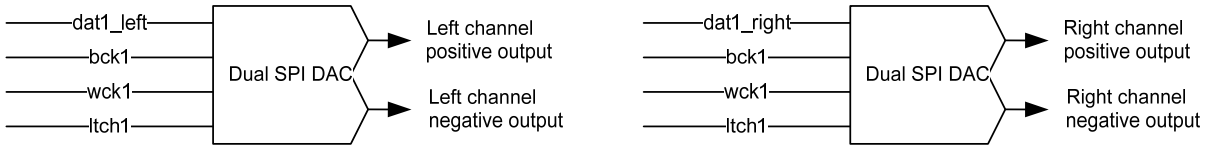
But if necessary obtain differential DAC from two simple ones, outputs with inverted data (**dat2_left**, **dat2_right**) are used:



In this configuration possibly use **bck2**, **wck2** and **ltch2** instead **bck1**, **wck1** and **ltch1** signals for all DACs but it is not recommended using both signal sets simultaneously because of possible out of sync.

SPI serial differential mode (spi_mode == 1)

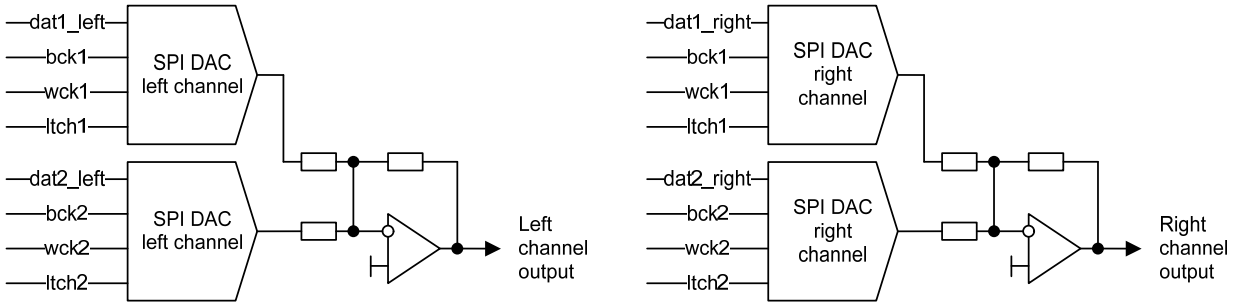
This mode supports dual SPI DAC only. There are **dat1_left** and **dat1_right** output differential data sample in two frames sequentially. Positive data frame first than follow negative data frame. Each frame has header to load data in selected DAC inside dual SPI device. So, in this mode dual SPI DAC works as single channel DAC with differential output:



Latch input of SPI DAC must be used for synchronous output update.

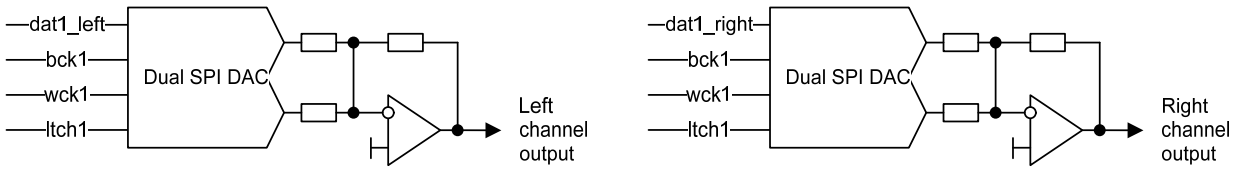
SPI pipeline mode (spi_mode == 2)

In this mode data output goes into two flows in pipeline mode. Each data flow goes to separate DAC:



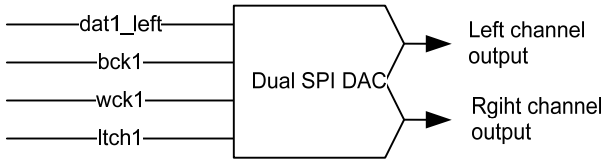
SPI series pipeline mode (spi_mode == 3)

Like serial differential this mode supports dual SPI DAC only. But each even data sample has header DAC channel1 and each odd data sample has header DAC channel2. Thus each channel of DAC works with half output sample frequency:



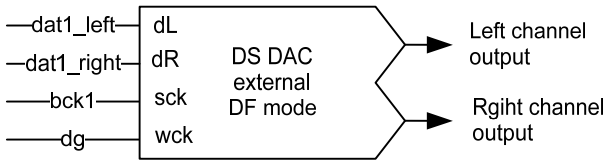
SPI series channels data mode (spi_mode == 4)

This mode also supports dual SPI DAC only. Output data sample of both channels loads to DAC in two consequently frames. Signal **ltch** synchronously updates DAC outputs:



RJ24 data mode (spi_mode == 5)

This mode intended to use with DS DAC in external DF mode. There is a bit clock (**bck**) force to continue state and **dg** signal force to duty cycle 50%. Left and right data channels output in parallel mode but LSB justified to falling edge of DG signal:



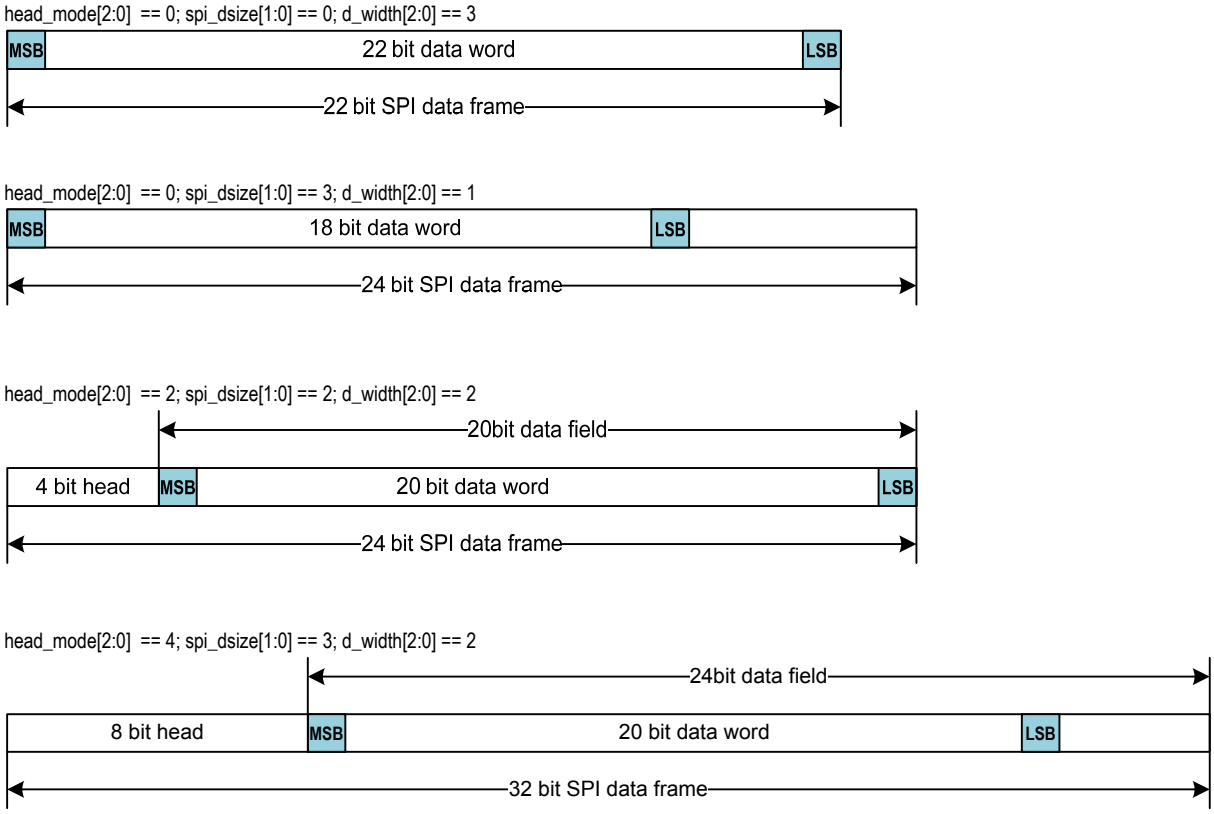
Only even values are allowed for bck speed dividers in this mode (value of two for d_speed[] not allowed).

3. SPI header modes

There are 7 header modes for output SPI data packet. Selection mode is set by pins **head_mode[2:0]**. Zero value turn head off, i.e. SPI packet consist data field only. Selection non zero value add header before data in SPI frame. **head_mode[2:0]** pins define the type of header that corresponds to a specific DAC: DAC8812, AD5791, AD5063, DAC11001, LTC1592 or LTC2752. For more details see timing diagrams in figure 9.

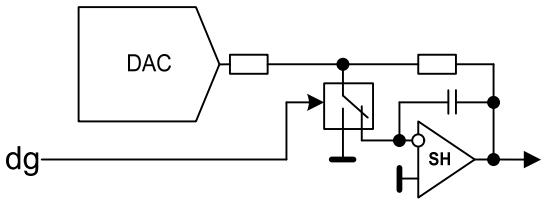
4. Data resolution and SPI data size

spi_dsize[1:0] value sets one of four data field length of output SPI packet. If **spi_dsize[]** goes LOW (audio DAC mode) that output has no header and data size equal **d_width[]** value. Non zero value of **spi_dsize[]** sets on three data field length: 16, 20 or 24 bit.
It's possible to set any value of data length for selected size SPI data field. For example:



5. Deglitcher and DAC update control signals

Deglitching output **dg** is used to decrease parallel DAC distortions. Deglitching circuit for parallel DAC is a sample-hold amplifier that goes into hold state before DAC output updates. When DAC output is settled the SH amplifier goes back to sample mode.
Simplified typical deglitching circuit:

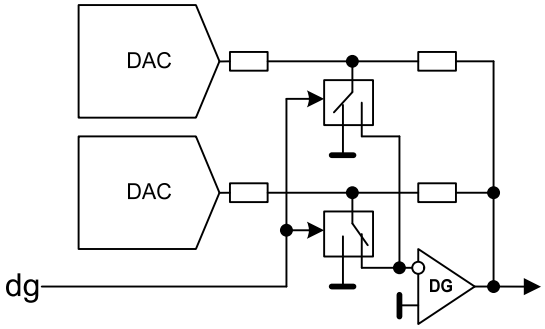


By default **dg** signal has low level to sample and high level to hold state. But pull up **dg_inv** input inverts **dg** signal (sample high / hold low).
Parallel DACs (audio and SPI) can update output in set of cases:

- falling edge WCK (audio DAC, like AD1851)
- rising edge WCK (SPI DAC, like DAC8830)
- falling edge of LATCH (SPI DAC with asynchronous input, like DAC8812)
- rising edge 2-s clock of BCK (PCM1704)
- rising edge 4-th clock of BCK (PCM1702)

Therefore **dgl_mode[1:0]** input controls deglitching transition to hold state. For more details see figure 5.
Also **dgh_mode[1:0]** input controls hold time of deglitching circuit with change duty cycle of **dg** signal. For more details see figure 6.

In pipeline mode no need for hold state because outputs of DACs are updated one by one. Therefore recommended deglitching circuit for pipeline mode is:



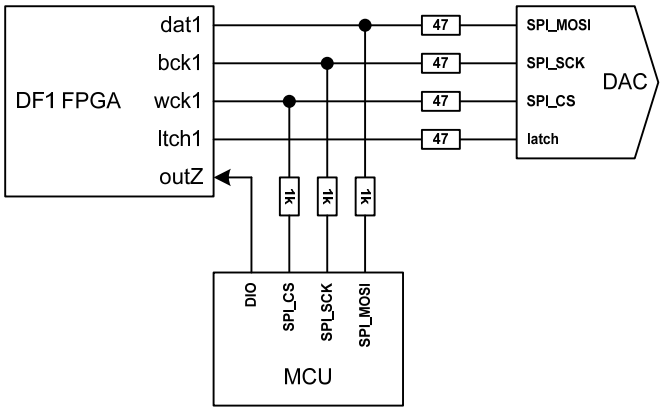
To switch **dg** signal in pipeline mode pull **dg_pp** input to 1. In this mode duty cycle of **dg** signal is always 50%. For more details see figure 7. Logic 1 to input **dg_off** forces **dg** signal to always sample state allowing the deglitching circuit to be disabled.

Linearity of the DAC conversion depends on the jitter of the update signal of the DAC output. Therefore, resynchronization of the DAC update signal from clean clock is needed. In case when deglitching circuit is used **dg** signal must be resynchronized. Otherwise **wck** or **ltch** signal must be resynchronized (or **bck** and **data** for PCM1702/1704).

6. SPI DAC configuration

Some SPI DACs require set of configuring commands to proper operation. Data and configuration settings are transmitted through common SPI port. Input **outZ** is used to provide access to external configuring MCU to the DAC SPI port. Logic 1 to **outZ** input switches outputs **dat1_left**, **dat1_right**, **bck1**, **wck1**, **dat2_left**, **dat2_right**, **bck2** and **wck2** to Z-state. When **outZ** input is set to 0 external MCU must turn it's SPI outputs to Z-state to exclude load from outputs FPGA ports.

External configuring MCU connection example:



7. Oversampling and master-clock

The DF1 is a linear phase filter based on set of x2 oversampling stages. There are 4 maximum oversampling ratio modes: x4, x8, x16, x32 that set by pins **ovs_max[1:0]**. Maximum oversampling ratio sets the output sampling rate which for x4 mode is 176,4 / 192kHz and for x32 - 1411,2 / 1536kHz. At the same time the minimum oversampling ratio is set automatically and depends on the input sampling rate.

For example: if maximum oversampling ratio is set by x16 and input sampling rate is 44,1kHz than actual oversampling ratio is equal maximum ratio: x16. But if input sampling frequency is changes to 192k than actual oversampling ratio automatically turn to x4 keeping output sampling rate unchanged.

When input sampling rate is equal to output sampling rate DF1 automatically goes into bypass mode. In this mode input data goes to output without oversampling processing but with selected attenuate level. If input sampling rate higher than output sampling rate that means invalid state and output data will be incorrect.

The DF1 has 4 sets of coefficients to first x2 interpolating stage and automatically chooses coefficients for this stage depending on input sample rate. When **lp_mode** input is set to 0 the DF1 chooses coefficients for the first stage with maximum performance. When **lp_mode** input is set to 1 the first stage processing performs with the shortest interpolating stages that decrease digital filter performance and power consumption.

The DF1 has best performance with master-clock frequency 1024Fs (44.1kHz/48kHz x 1024) on **clk** input. But if necessary it is possible to use a lower master-clock frequency with corresponding performance decrease. There are two main frequency domains. By default when input **mck_mode** is 0 the DF1 works in 1024Fs domain. Force **mck_mode** to 1 is activating 768Fs frequency domain. For both domains x32 max oversampling rate is allowed wjth the same maximum input sampling rate of 768kHz.

It is possible to double or quad decrease master-clock frequency for any domain with corresponding decrease oversampling performance, maximum oversampling ratio and maximum input sample rate. For example, if master-clock is 256Fs than maximum oversampling ratio is 32/4 = 8 (**ovs_max[1:0]** == 3) and maximum input sampling rate is 768/4 = 192kHz. In this case maximum oversampling ratio is set as x8 and the input sampling rate is 96kHz and actual oversampling ratio will turn to x4. The DF1 will work with 1024Fs master-clock, 384kHz input sampling frequency and maximum oversampling ratio x32. **lp_mode** must be set to 1 when master-clock frequency decrease is used.

8. Indicating

The DF1 has indicating input sample rate (**ovs_ind[2:0]** output) and digital clipping (**clipping** output). All indicating outputs are configured active low push-pull. Led indicating connecting example:

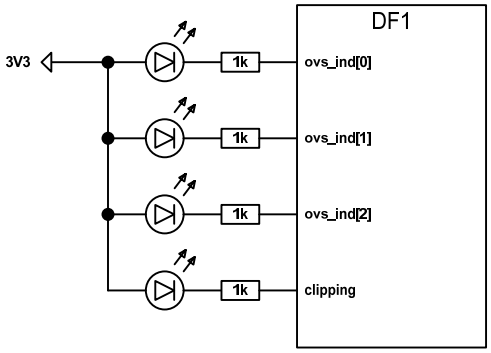


Figure 1. Serial audio input data formatting

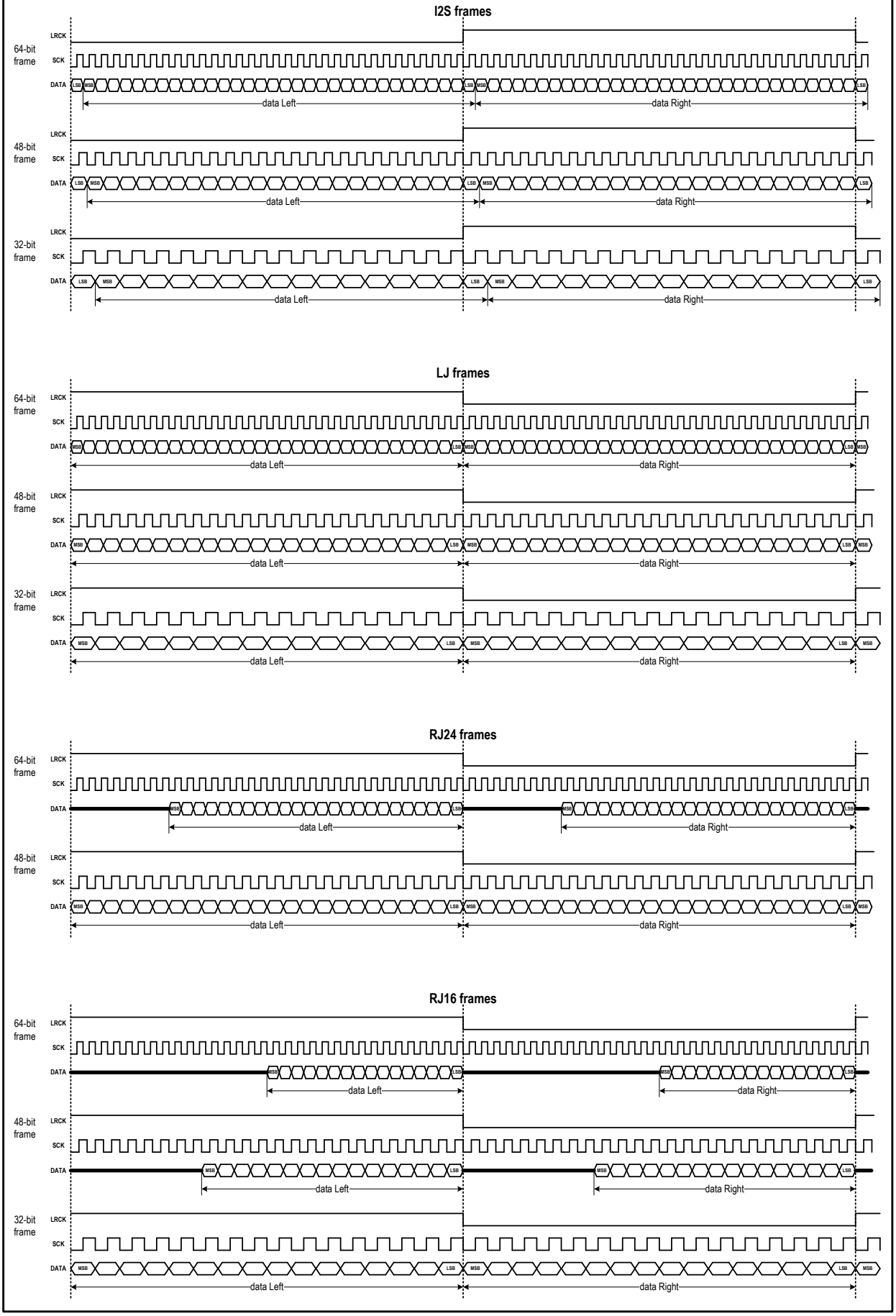


Figure 2. **Data rate options for serial data output description with input data: 0x800111 – left, 0xC00333 – right**

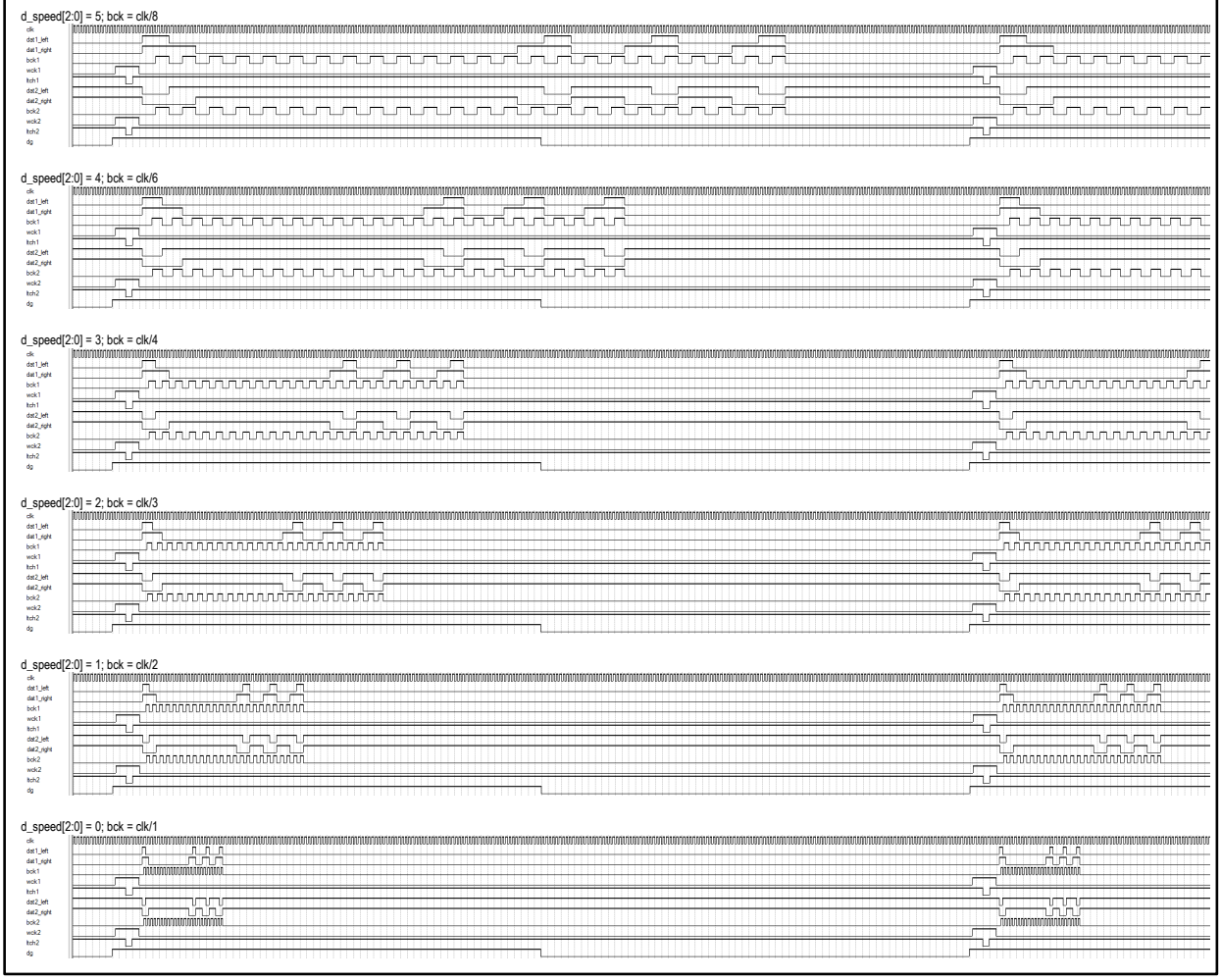


Figure 3. **Bck invertion option for serial data output description with input data: 0x800111 – left, 0xC00333 – right**

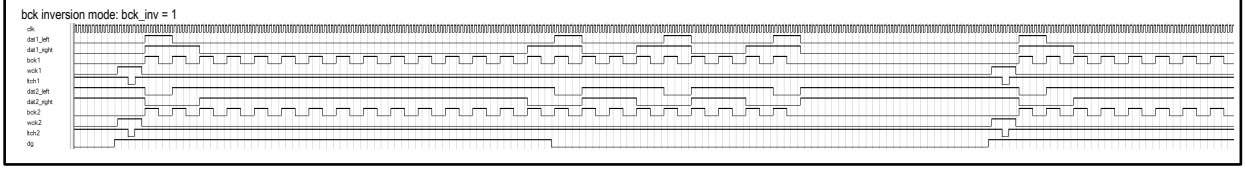


Figure 4. **Offset binary mode set description with input data: 0x800111 – left, 0xC00333 – right**

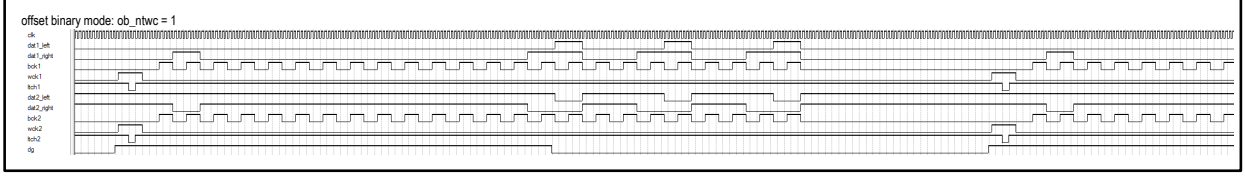


Figure 5. **Deglitcher sample to hold transition options description with input data: 0x800111 – left, 0xC00333 – right**

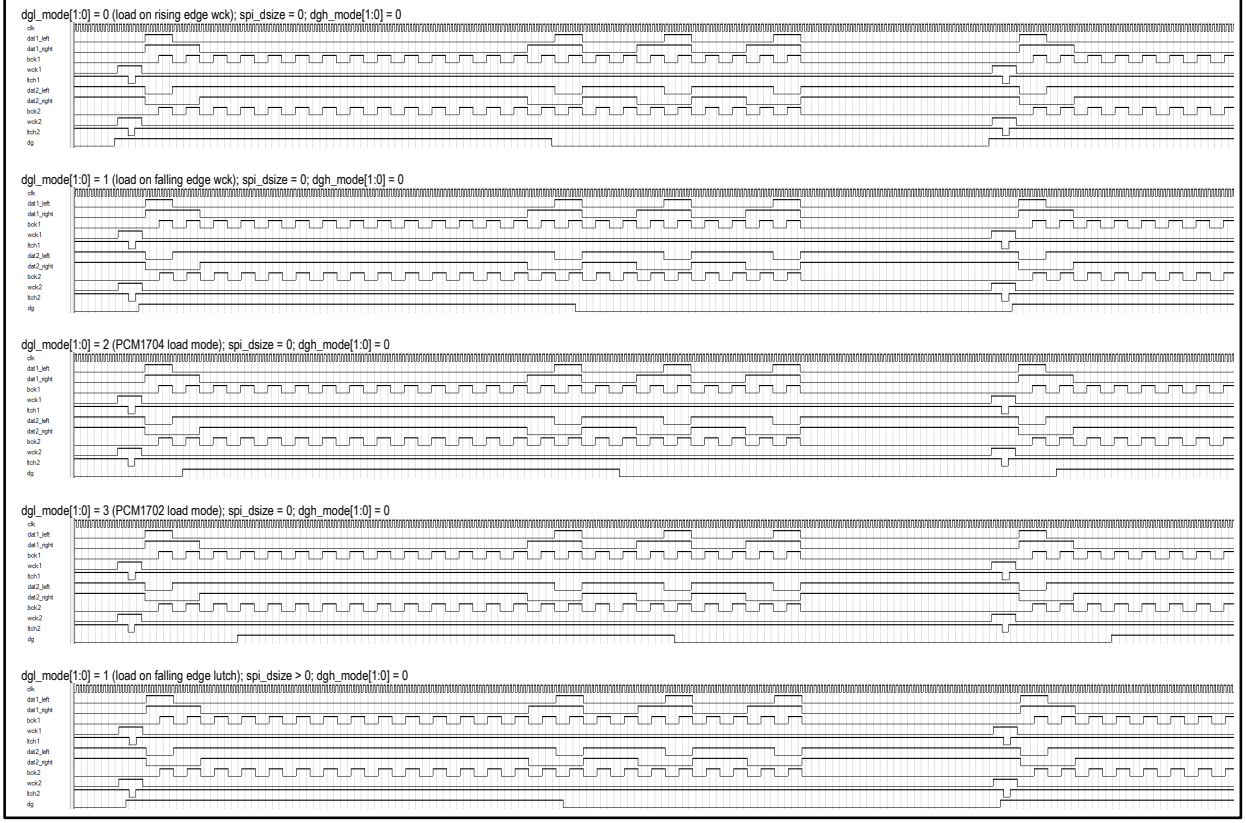


Figure 6. **Deglitcher hold to sample transition options description with input data: 0x800111 – left, 0xC00333 – right**

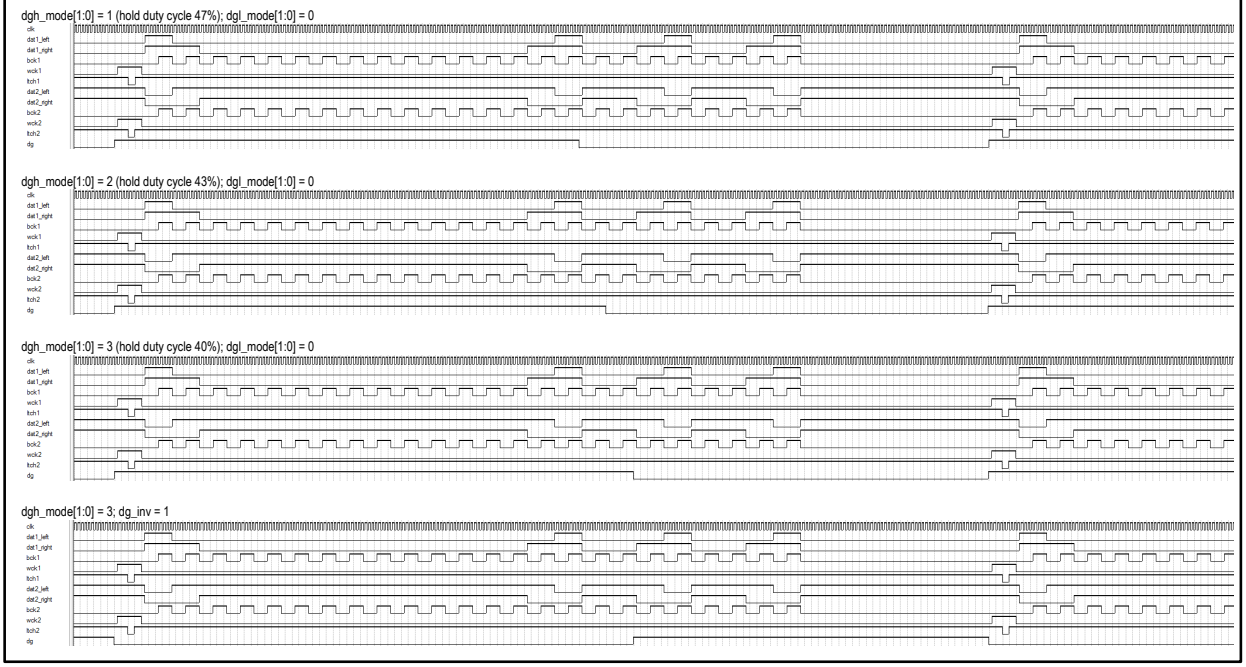


Figure 7. **Deglitcher pipeline mode option description with input data: 0x800111 – left, 0xC00333 – right**

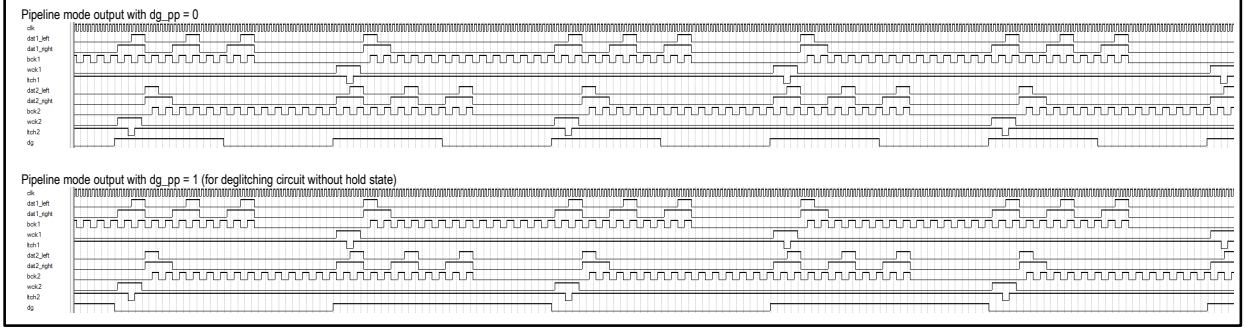


Figure 8. **SPI modes for serial data output description with input data: 0x800111 – left, 0xC00333 – right**

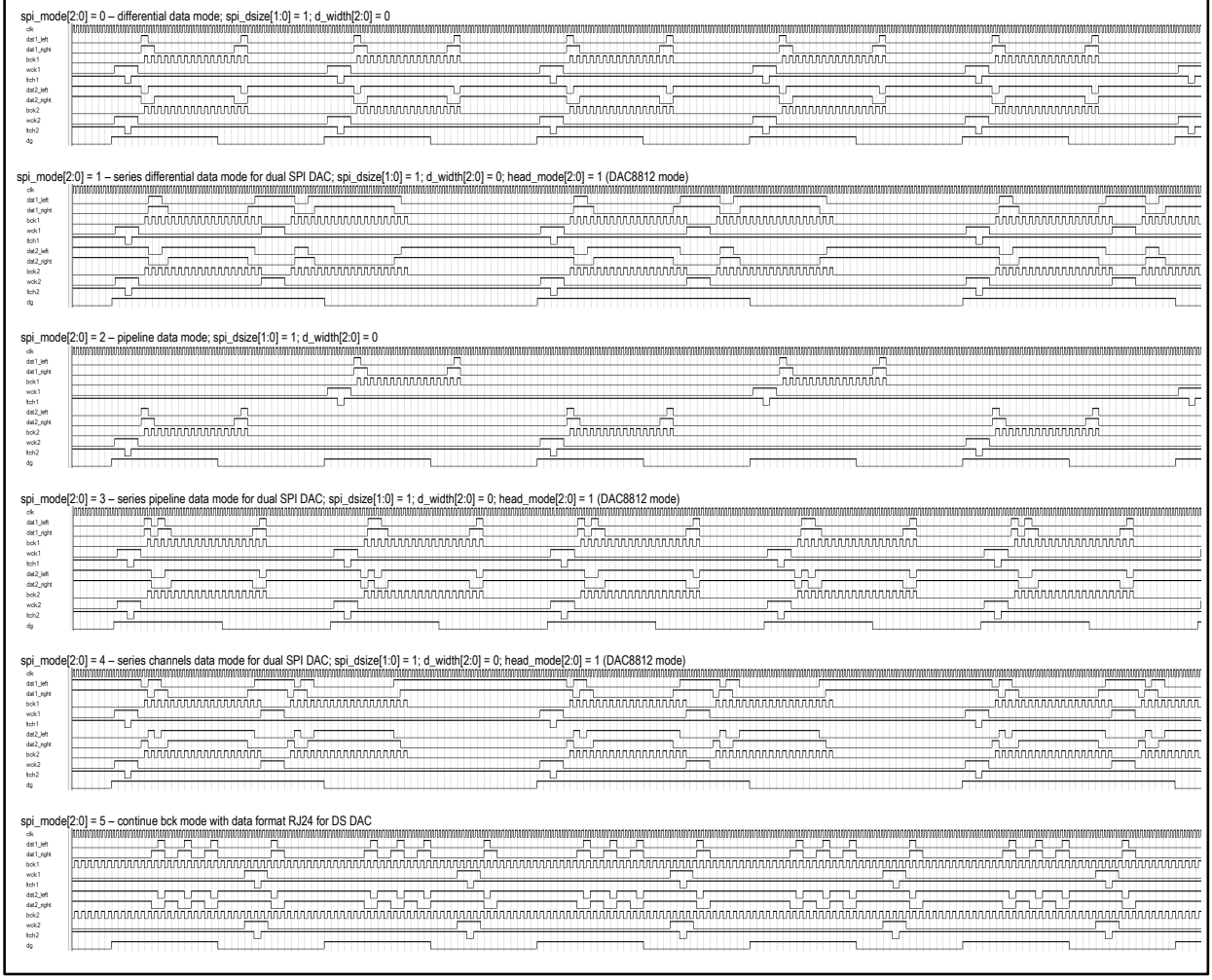


Figure 9. **SPI header modes description with input data: 0x800111 – left, 0xC00333 – right**

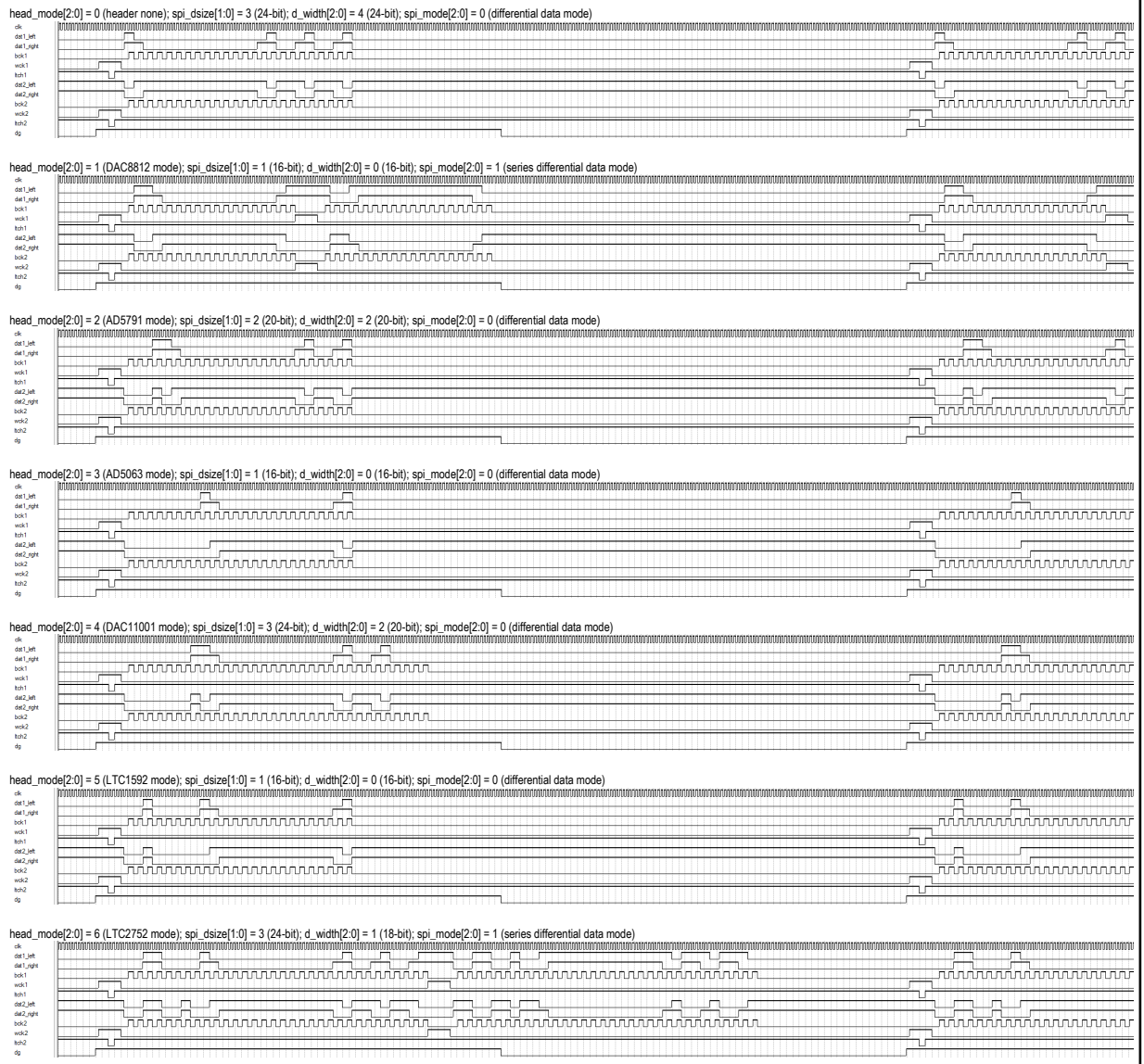
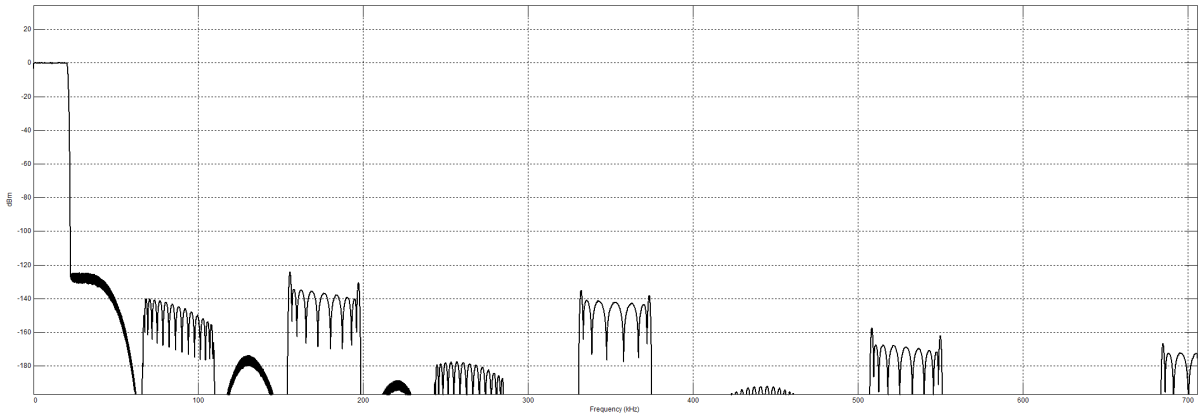
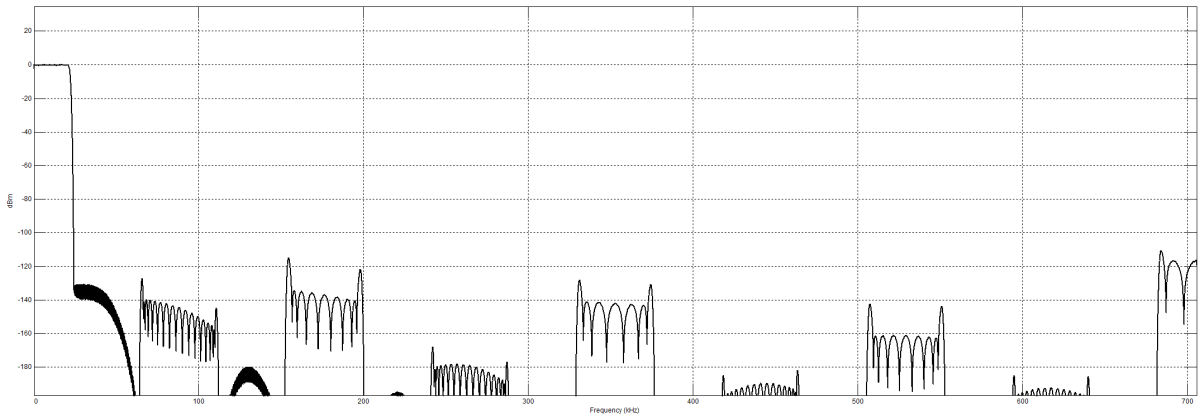


Figure 10. **FIR frequency response with ovs_max[] = 3 (x32)**

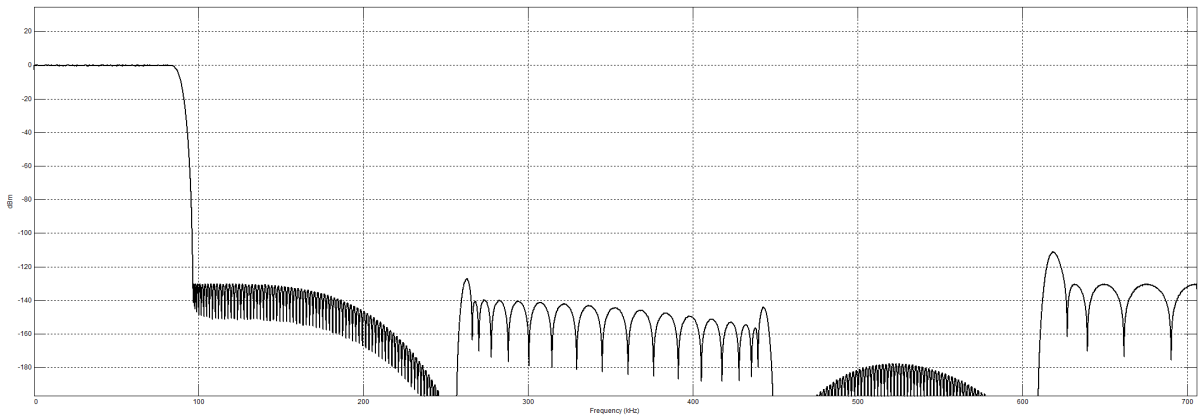
Input sample rate 44.1kHz; lp_mode = 0



Input sample rate 44.1kHz; lp_mode = 1



Input sample rate 176.4kHz; lp_mode = 0



Input sample rate 176.4kHz; lp_mode = 1

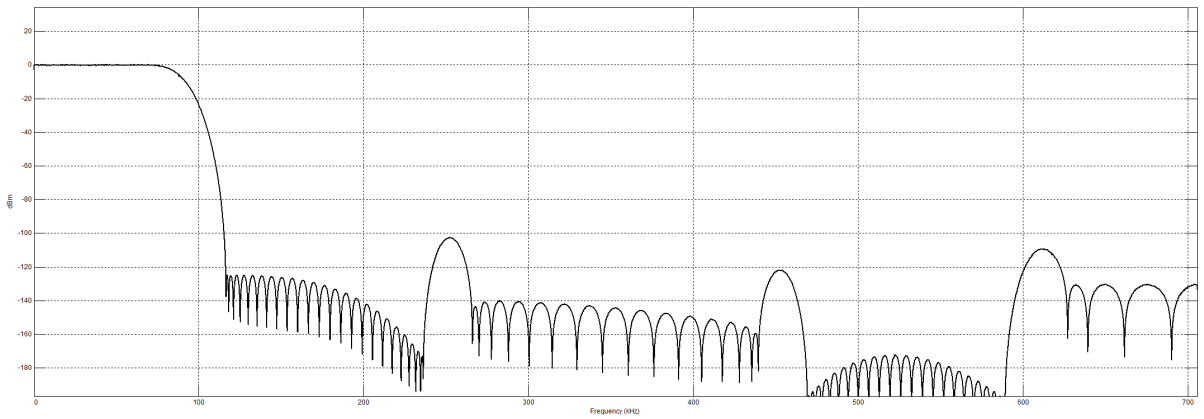
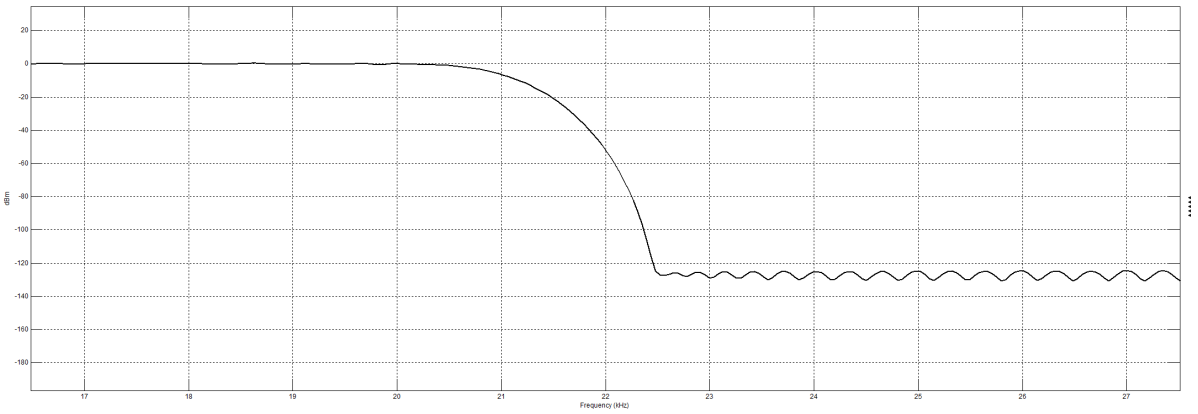
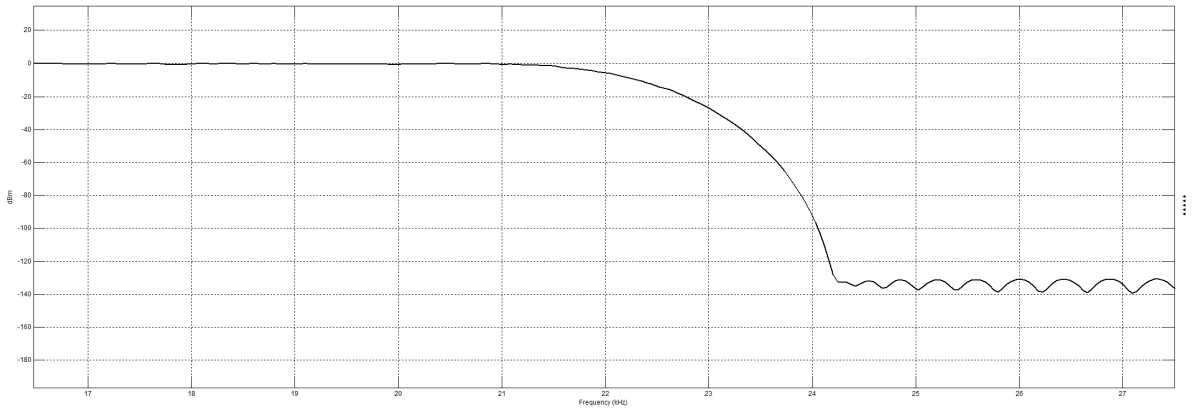


Figure 11. **FIR passband and Nyquist frequency response with ovs_max[] = 3 (x32); input sampling frequency 44.1kHz**

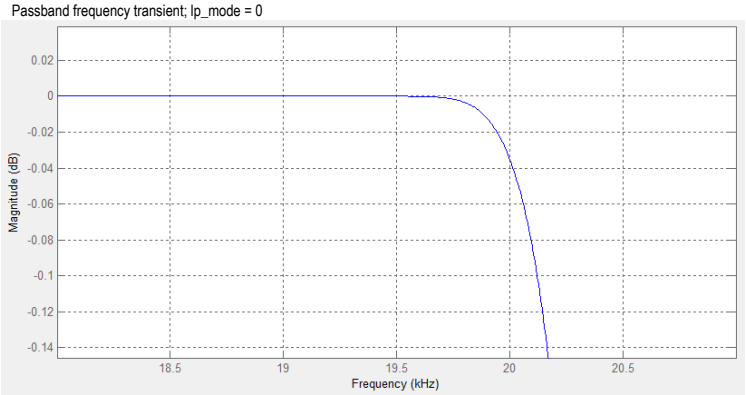
Nyquist frequency transient; lp_mode = 0



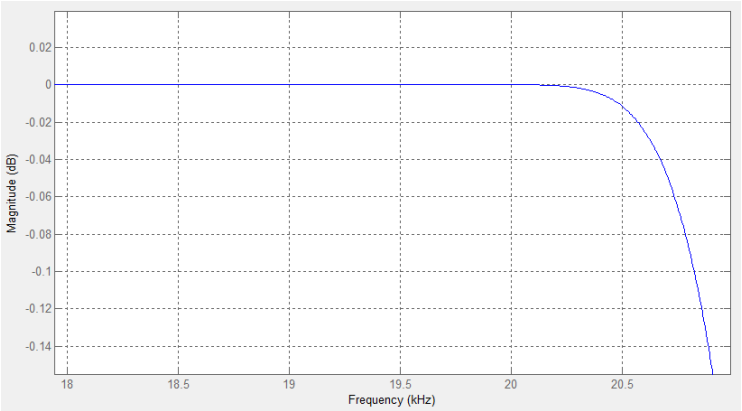
Nyquist frequency transient; lp_mode = 1



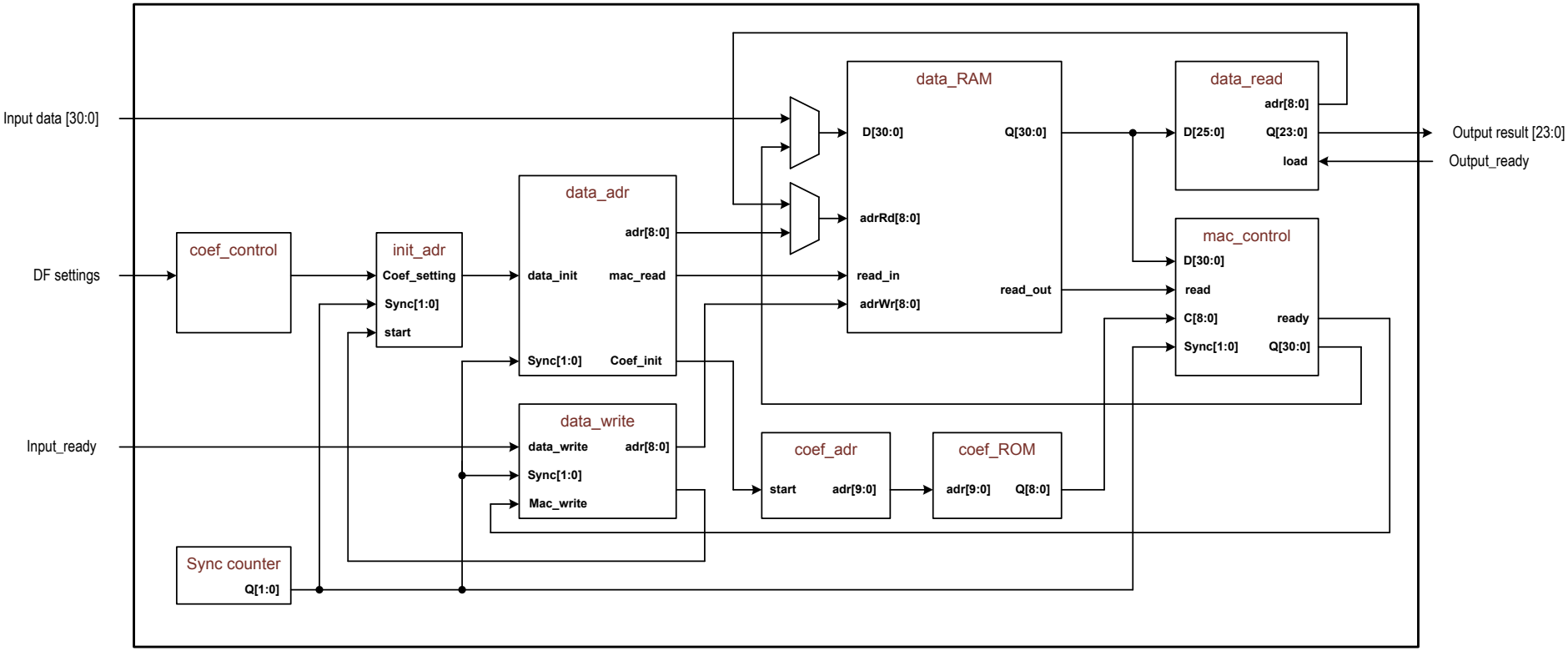
Passband frequency transient; lp_mode = 0



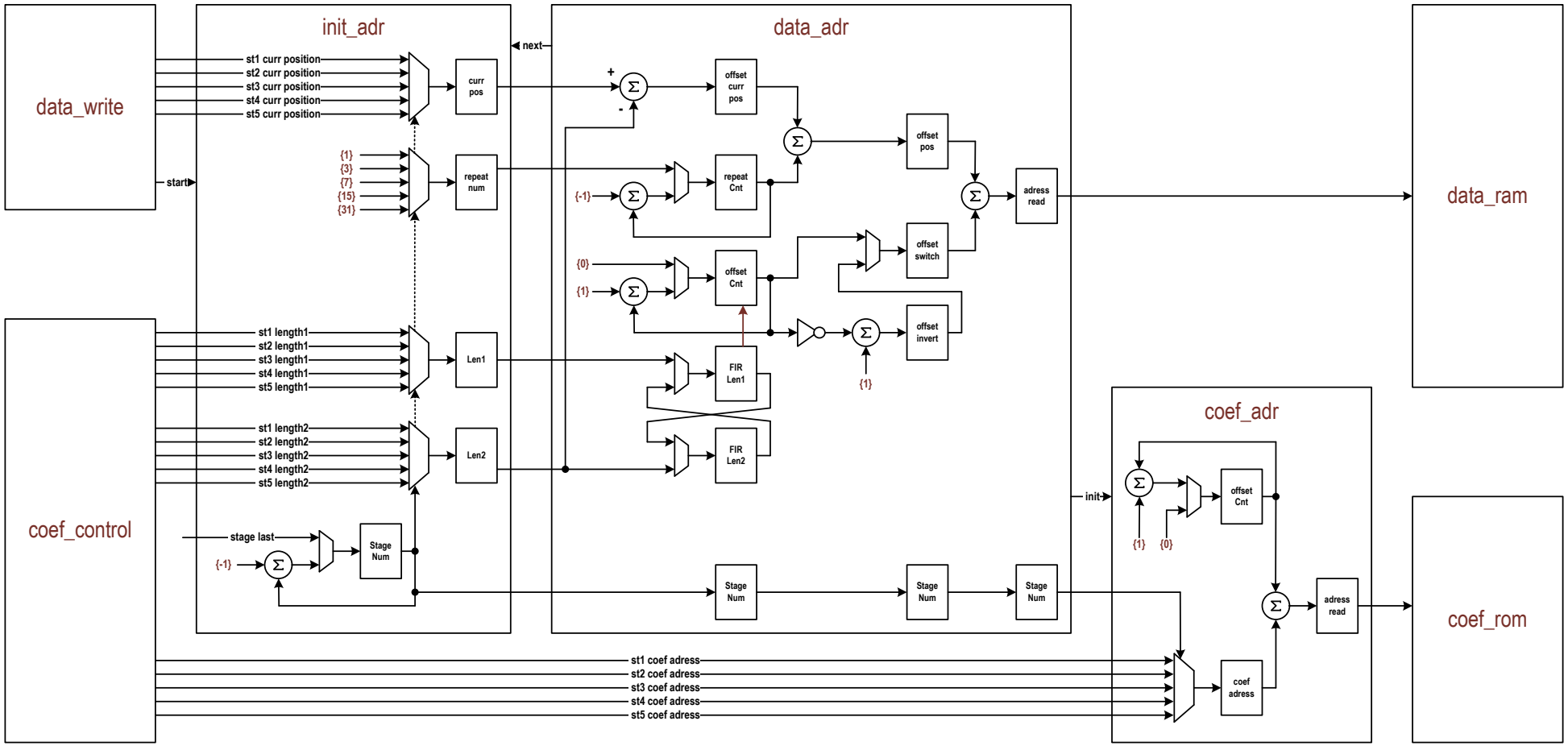
Passband frequency transient; lp_mode = 1



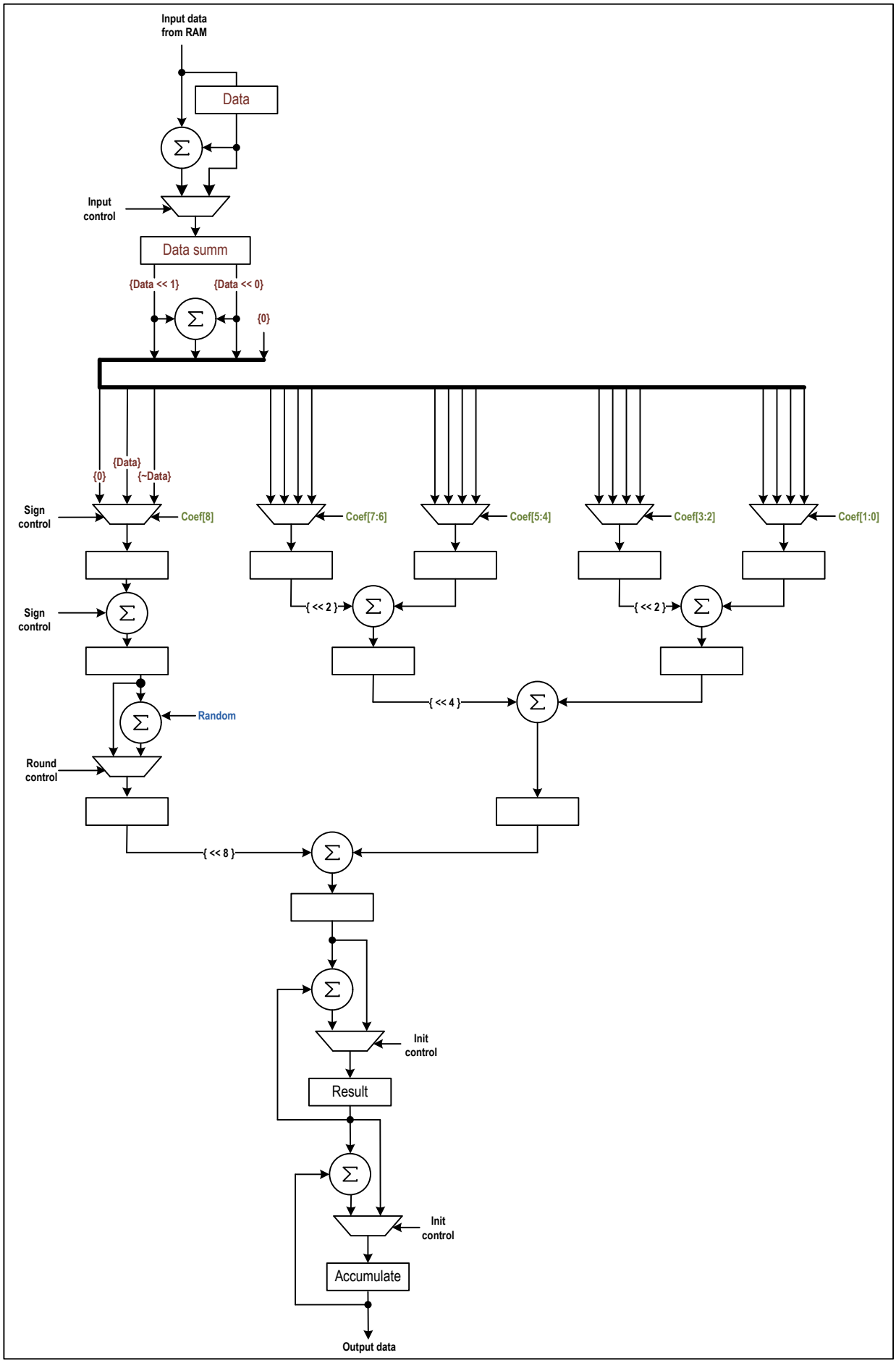
DF1 FIR core simplified diagram



Simplified adress generation machine diagram for symmetric linear phase finite response filter



Altera FPGA optimized MAC architecture



Lattice FPGA optimized MAC architecture