SDM Decimator IP core

Stefan Kull, Roy Seitz, Marco Zollinger

December 2, 2016

1 Overview

This Document describes the functionality and usage of the SDM Decimator IP core. It is supplied as a complete, ready to use IP core.

The project Soundloc contains three microphones that deliver a $\Sigma\Delta$-modulated bitstream. This IP generates the clock needed by the microphones and processes the received bitstreams. It outputs filtered and decimated signed values and an interrupt to indicate when new values are available. It is widely configurable by generic parameters at compile time as well as by software. See section 2 and 8 for detailed information.

The filter consists of a CIC filter, which is configurable up to 3rd order and a decimation factor up to its internal data width. This is described in section 5. An additional IIR DC blocker with a configurable pole can be added to remove a DC component. See section 6 for detailed information.

Finally a post divider is implemented to scale the decimated data by powers of $1/2$. This can be used to eliminate problems caused by overflow in the following cross-correlation that may occur with high order and decimation factor. The post divider is controlled by software and described in section 7.

2 Parameter description

The IP core can be configured at compile time by several parameters, listed in table 1.

Table 1: Parameters for the SDM_Decimator

Parameter	Default	Range	Type	Description
D_WIDTH	16	1...32	integer	Internal data width
D_OUT_WIDTH	16	1...18	integer	Output data width
DIVIDE	40	4...1024	integer	Prescaler for bitstream clock

3 Register description

Table 2 describes the available registers and their functionality. There are a total of five registers. Each can be accessed directly by their address or more conveniently by the software driver described in section 8. The individual bits in the status register are described in table 3.

Table 2: Register space

Name	Address offset	Mode	Description
Status	0x00	R/W	Contains Status and control bits described in table 3
Decimation	0x04	R/W	Contains the decimation factor of the CIC filter
Value0	0x08	R	Contains the actual, filtered value for microphone 2
Value1	0x0C	R	Contains the actual, filtered value for microphone 1
Value2	0x10	R	Contains the actual, filtered value for microphone 3

Table 3: Status register

Bit	Name	Description
1:0	order	Binary coded order "00" deactivates filter. Outputs are set to 0
2	irq_ena	Enables the interrupt signal
3	iir_ena	Enables the IIR DC blocker
7:4	iir_sr	Sets the pole of the IIR DC blocker
11:8	post_divide	Sets the post divider

4 Bitstream settings

The Bitstream clock is derived from the AXI Interface clock, prescaled by the parameter DIVIDE. The microphones apply the new bits on falling edge and the Filter reads them on rising edge. As the filter operates on signed values, the bitstream {1,0} is interpreted as {+1, –1}.

5 CIC filter

This section explains the functionality of the implemented CIC filter. A CIC filter is fully characterized by its order, the differential delay and the decimation factor. The resulting transfer function is shown in equation 1.

$$H_{CIC}(z) = \frac{\left(1 - z^{-RM}\right)^N}{\left(1 - z^{-1}\right)^N} = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N \tag{1}$$

The differential Delay is fixed to $M = 1$. The decimation factor R and the order N are set by software. See section 8 for further information. The decimation factor is limited to the internal register width defined

by parameter D_WIDTH. A value greater than $2^{D\_WIDTH} - 1$ results in undefined behavior. The order can be set to $0 \ldots 3$. *Order 0 deactivates the filter and forces its outputs to 0.*

With high decimation factors and orders, overflow can occur. The input is interpreted as 2bit signed value ± 1. Therefore, to avoid overflow the internal register width D_WIDTH must be at least

$$D\_WIDTH \geq N \log_2 (RM) + 1 \tag{2}$$

6 IIR DC blocker

A simple IIR DC blocker is available to remove the DC component due to microphone offsets. This may be needed as the correlation accumulates this DC component and may overflow. The filter has the transfer function given in equation 3. The pole is set close to $z = 1$ to obtain a low cutoff frequency. The exact location is given in equation 4. $iir\_sr \in [0, 15]$ is set by software.

$$H_{IIR} = \frac{1 - z^{-1}}{1 - (1 - 2^{-iir\_sr})\, z^{-1}} \tag{3}$$

$$z_p = \frac{2^{iir\_sr} - 1}{2^{iir\_sr}} \tag{4}$$

7 Post divider

The post divider can divide the internally calculated values by powers of two. The high gain from the CIC filter can therefore be scaled. This is handy if the internal width is chosen greater than the output width. The divider is implemented as a simple arithmetic shift right by post_divider $\in [0 \ldots 15]$. Thus setting post_divider to 0 disables the divider. The output value results in

$$out = out_{internal} \cdot 2^{-post\_divider} \tag{5}$$

8 Driver

The IP core comes along with a driver to allow easy access by software. There are two categories of functions, high and low level. The usage of high level functions is recommended.

8.1 High level functions

A typical use case with high level functions is as follows:

- call SDM_DECIM_setStatus(baseaddr, mask) to initialize the IP core.
  This sets the content of the status register to mask. See section 3 for further information.
  Typically: SDM_DECIM_setStatus(SDM_DECIM_BASEADDR, 0x307).
- Call SDM_DECIM_setDecimation(baseaddr, decimation).
  This sets the decimation factor. Typical values are about 30.
- If the raw values are of interest, call SDM_DECIM_getValue(baseaddr, mic).
  This returns a 32 bit signed integer that corresponds to the microphone mic.
  mic can be one of SDM_DECIMATOR_MICx, with $x \in \{1, 2, 3\}$.

8.2 Low level functions

The core can also be used by addressing the registers directly. For this purpose two low-level functions are available. These functions are implemented as macros and should only be used if really necessary. The definitions from table 4 can be used to easily access these Registers. Care must be taken to not confuse the microphones. On the PCB they are numbered from 1 to 3 but in hardware the numbering goes from 0 to 2. The macros SDM_DECIMATOR_MICx are provided for mapping the indices.

- SDM_DECIM_mWriteReg(BaseAddress, RegOffset, Data)
  Write data to the specified Register.
- SDM_DECIM_mReadReg(BaseAddress, RegOffset)
  Read the content of the specified register.

8.3 Defined constants

Table 4: Software defined constants

Name	Value
SDM_DECIMATOR_S_AXI_REG0_STATUS_OFFSET	0
SDM_DECIMATOR_S_AXI_REG1_DECIMATION_OFFSET	4
SDM_DECIMATOR_S_AXI_REG2_VALUE0_OFFSET	8
SDM_DECIMATOR_S_AXI_REG3_VALUE1_OFFSET	12
SDM_DECIMATOR_S_AXI_REG4_VALUE2_OFFSET	16
SDM_DECIMATOR_MIC1	1
SDM_DECIMATOR_MIC2	0
SDM_DECIMATOR_MIC3	2

9 Test and verification

The core can easily be simulated by using the AXI Trafic generator IP from Xilinx and feeding some value to the bitstream inputs. Therefore no additional simulation files are available. Software verification can be done by writing and reading data to the status and decimation factor registers.

10 Compatibility and License

The core is tested under Vivado version 2016.2 and on Artix7 and Zynq7 FPGA. Since the core does not use any hardware specific resources, it should run on every other FPGA as well. However, this is not tested and may require changes to the core hdl or driver files. The core is supplied under no license or copyright but is the intellectual property of the authors.