

## 1. Features

SERIAL\_INTF is a companion core that can be used for MVD modulator core initialization. It allows to parameter settings and to read status.

External input interface can be :

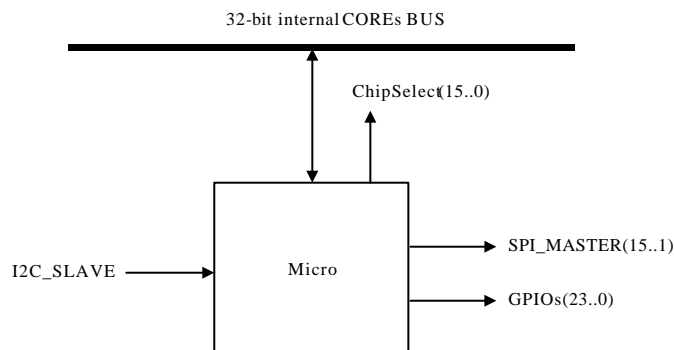
- I2C 400KHz slave interface for 8-bit registers access
- UART interface with 9600 Bauds or 115200 Bauds, 8-bit, NO parity, 1 STOP bit

## 2. Applications

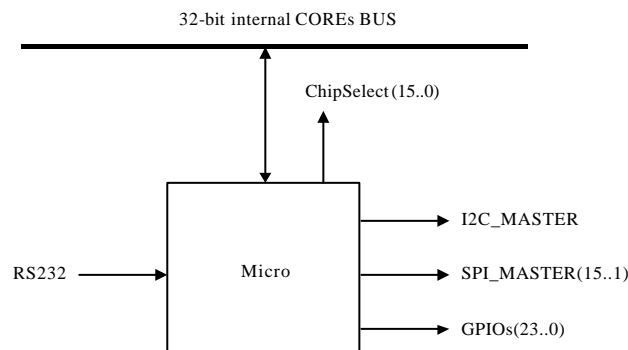
SERIAL\_INTF core can be used with any MVD modulator core when local CPU is not available.

## 3. SERIAL\_INTF overview

This core can directly drive the 32-bit CPU interface of the MVD modulator cores. It also delivers connection for master SPI, master I2C and GPIO interfaces.



**Figure 1- Overview diagram of the SERIAL\_INTF Core with I2C input**



**Figure 2- Overview diagram of the SERIAL\_INTF Core with UART input**

## SUMMARY

<b>1. FEATURES .....</b>	<b>1</b>
<b>2. APPLICATIONS .....</b>	<b>1</b>
<b>3. SERIAL_INTF OVERVIEW .....</b>	<b>1</b>
<b>4. I2C SLAVE INTERFACE.....</b>	<b>3</b>
4.1. INTERNAL BUS CORE INTERFACE.....	4
4.2. SPI MASTER INTERFACE.....	4
4.3. GPIO INTERFACE.....	5
<b>5. UART INTERFACE.....</b>	<b>6</b>
COMMAND FIELD : .....	6
SPI MASTER INTERFACE : .....	7
GPIO INTERFACE : .....	8
I2C MASTER INTERFACE : .....	9
<b>MULTI FPGA SPI CONFIGURATION .....</b>	<b>10</b>
ADDR FIELD : 7-BIT.....	10
DUMMY : .....	11
BYTE : .....	11
<b>6. DEFAULT INITIALIZATION PROVIDED AS OPTION .....</b>	<b>12</b>
<b>7. PORT DEFINITION .....</b>	<b>13</b>
<b>8. SERIAL_INTF CORE FILES AND DELIVERABLES .....</b>	<b>14</b>
8.1. CORE FILE.....	14
8.2. IMPLEMENTATION EXAMPLES .....	14
<b>9. RESOURCE UTILIZATION.....</b>	<b>15</b>
<b>10. USE WITH MVD CORES.....</b>	<b>15</b>
<b>11. REVISION HISTORY.....</b>	<b>16</b>

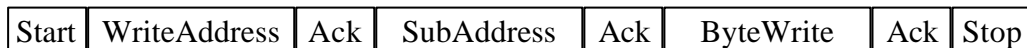
#### 4. I2C slave interface

I2C slave interface is made of S\_SCL input only signal for clock with 400 kHz maximum frequency, and S\_SDA signal for bidirectional data bit. The S\_SDA output is open collector. The default I2C chip address is 0x18 (WriteAddress) for write and 0x19 (ReadAddress) for read. This interface can drive the 32-bit internal bus, a SPI master output interface and a 24-bit GPIO port.

The I2C component address can be selected over 8 possibilities, a 3 bit address vector S\_ADR can be used to define this address.



I2C compliant 8-bit registers read and write are supported.



**Figure 3 – Byte write cycle**



**Figure 4 – Byte read cycle**

SubAddress is used for internal registers addressing. SubAddress from 0x00 to 0x7F are directed to MVD\_DVBx core registers (look at MVD\_DVBx specific core documentation for registers mapping). SubAddress from 0x80 to 0x8F are used for additional external IOs addressing.

#### **Exemple of programming a MVD CORE register :**

Programming 32 bits register with address 0x40 and values 0x12345678, following commands will be sent :

```
START / 0x18 / ACK / 0x40 / ACK / 0x78 / ACK / STOP
START / 0x18 / ACK / 0x41 / ACK / 0x56 / ACK / STOP
START / 0x18 / ACK / 0x42 / ACK / 0x34 / ACK / STOP
START / 0x18 / ACK / 0x43 / ACK / 0x12 / ACK / STOP
```

To read this register following dialog will be done :

```
START / 0x18 / ACK / 0x40 / ACK / START / 0x19 / ACK / 0x78 / NACK / STOP
START / 0x18 / ACK / 0x41 / ACK / START / 0x19 / ACK / 0x56 / NACK / STOP
START / 0x18 / ACK / 0x42 / ACK / START / 0x19 / ACK / 0x34 / NACK / STOP
START / 0x18 / ACK / 0x43 / ACK / START / 0x19 / ACK / 0x12 / NACK / STOP
```

### 4.1. Internal bus Core interface

SubAddress from 0x00 to 0x7F are directed to MVD\_DVBx core registers. Address 0xX0 is for lower data byte bit(7..0), address 0xX1 is for bit(15..8), address 0xX2 is for bit(23..16) and address 0xX3 is for upper data byte bit(31..24).

**Chip select Write/Read** : SubAddress 0x80

The MVD\_DVBx 32-bit core bus can drive 15 chip selects. This SubAddress is used to set and reset a particular chip select. The chip select must be set active prior to start register access.

The ByteWrite and ByteRead format for chip select is :

Bit(7..4)	Unused
Bit(3..0)	Chip select number from 0 to 15 (0x0 to 0xF)

By default chip select 0 is set active.

### 4.2. SPI master interface

The highest bit is sent first.

**Chip select write only** : SubAddress 0x84

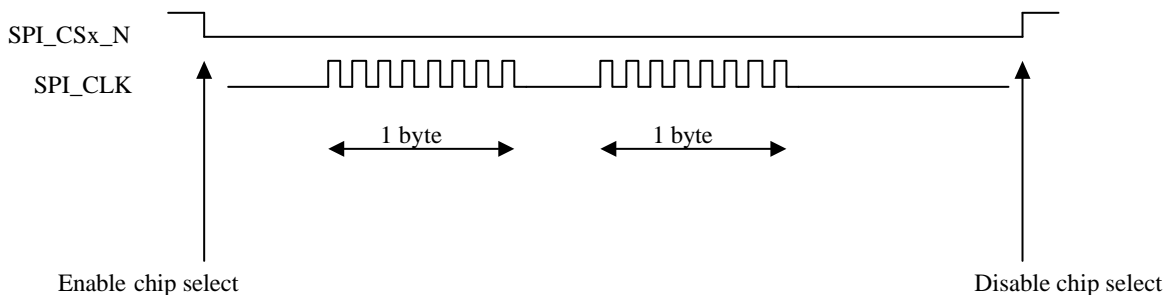
SPI master serial interface can drive 15 chip selects. One chip select must be set active before starting a data transfer. Only write accesses are supported.

The ByteWrite format is :

Bit(7..4)	Unused
Bit(3..0)	Chip select number from 1 to 15 (0x1 to 0xF, 0x0 all chip select are inactive)

By default all chip select are in inactive state.

**Data register write only** : SubAddress 0x85



**Figure 5 – SPI master example**

### 4.3. GPIO interface

GPIO is a 24-bit general purpose parallel port. Each bit can be configured as output or input with a direction register. During write access only outputs are driven, during read access data is coming from output data register for output ports and from chip pad for input ports.

#### Direction registers Write/Read :

SubAddress 0x88	direction for GPIO(7..0)	1= output	0=input
SubAddress 0x89	direction for GPIO(15..8)	1= output	0=input
SubAddress 0x8A	direction for GPIO(23..16)	1= output	0=input

By default GPIO(7..0) are outputs , others are inputs.

#### Data registers Write/Read :

SubAddress 0x8C	data for GPIO(7..0)
SubAddress 0x8D	data for GPIO(15..8)
SubAddress 0x8E	data for GPIO(23..16)

By default GPIO(7..0) are set to 0 level.

## 5. UART interface

UART interface can be configured as 9600 Bauds or 115200 Bauds, 8bit data, NO parity and 1 STOP bit. No hardware flow control is available. The B115K pin when set at 0 level selects 9600 Bauds rate, setting it to level 1 selects 115200 Bauds rate. Transfers are defined as 6 bytes frame for write and 4 bytes frame for read. This interface can drive 32-bit internal bus, SPI master output interface, I2C master 8/16-bit output interface, and 24-bit GPIO port.



**Figure 6 – UART write frame**

STX	start of frame code 0x55
CMD	command field
BYTE(3)	upper data byte bit(31..24)
BYTE(2)	data byte bit(23..16)
BYTE(1)	data byte bit(15..8)
BYTE(0)	lower data byte bit(7..0)

Any read command returns a four byte frame.



**Figure 7 – UART read frame**

### Command field :

Bit(7..6) Two bits command

'00'	Chip select number on 32-bit MVD_DVBx core bus
'01'	IOs commands
'10'	Write 32-bit MVD_DVBx core bus
'11'	Read 32-bit MVD_DVBx core bus

Bit(5..0) For '00' command

Bit(5..4)	unused
Bit(3..0)	Chip select number from 0 to 15 (By default chip select 0 is active)

Bit(5..0) For '1x' commands

Bit(5)	must be 0
Bit(4..0)	32 registers address (Address is for 32-bit register size) shifted on the right of 2 bits (example : <b>0x04</b> register address of MVD_DVBx core is <b>0x01</b> register address of UART)

Bit(5..0) For '01' command

Bit(5..4) '00'	I2C master
'01'	SPI master
'10'	GPIO
'11'	Reserved
Bit(3..1)	IO parameters (Look at specific IO mode I2C, SPI, GPIO)
Bit(0)	'1' for read access, '0' for write access

### SPI master interface :

The command field header must be :  $CMD(7..4) = '0101'$  (0x5)

Notice that only write accesses are supported ( $CMD(0)='0'$ ).

The highest bit is sent first.

#### Chip select write only :

SPI master serial interface can drive 15 chip selects. One chip select must be set active before starting a data transfer.

The IO parameters field  $CMD(3..1)$  format is :

'000'            Enable/Disable chip select number (Number is coded in data Byte(0) )

The Byte(0) format is :

Bit(7..4)        Unused

Bit(3..0)        Chip select number from 1 to 15 (0x1 to 0xF, 0x0 all chip select are inactive)

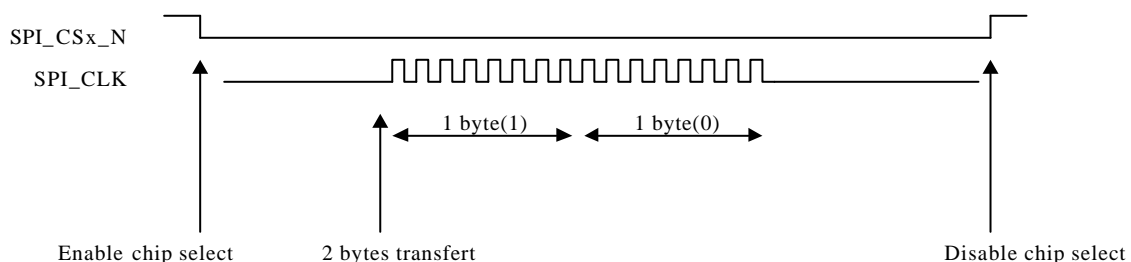
By default all chip select are in inactive state.

#### Data register write only :

SPI master serial interface support 1, 2, 3 or 4 bytes transfer in a single command.

The IO parameters field  $CMD(3..1)$  format is :

'001'	One byte transfer	(Byte(0))
'010'	Two bytes transfer	(Byte(1) & Byte(0))
'011'	Three bytes transfer	(Byte(2) & Byte(1) & Byte(0))
'100'	Four bytes transfer	(Byte(3) & Byte(2) & Byte(1) & Byte(0))



**Figure 8 – SPI master example**

**GPIO interface :**

The command field header must be :  $CMD(7..4) = '0110'$  (0x6)

GPIO is a 24-bit general purpose parallel port. Each bit can be configured as output or input with a direction register. During write access only outputs are driven, during read access data is coming from output data register for output ports and from chip pin for input ports.

**Direction registers Write/Read :**

The IO parameters field  $CMD(3..1)$  format is : '000'

Byte(0) direction for GPIO(7..0) 1= output 0=input  
Byte(1) direction for GPIO(15..8) 1= output 0=input  
Byte(2) direction for GPIO(23..16) 1= output 0=input

By default GPIO(7..0) are outputs , others are inputs.

**Data registers Write/Read :**

The IO parameters field  $CMD(3..1)$  format is : '001'

Byte(0) data for GPIO(7..0)  
Byte(1) data for GPIO(15..8)  
Byte(2) data for GPIO(23..16)

By default GPIO(7..0) are set to 0 level.

## I2C master interface :

The command field header must be : CMD(7..4) = '0100' (0x4)

The I2C master serial port can generate 8-bit or 16-bit read and write transfers.

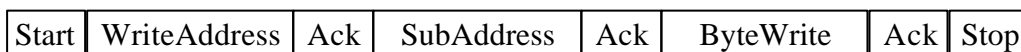
The IO parameters field CMD(3..1) format is :

'000'	for 1 byte transfer	8-bit
'001'	for 2 bytes transfer	16-bit

CMD(0)'1' for read access, '0' for write access

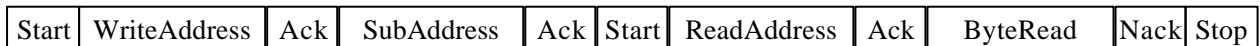
The data format is :

BYTE(3)	Write Address of slave component on I2C bus
BYTE(2)	Register SubAddress of slave component on I2C bus
BYTE(1)	Upper data byte in 16-bit transfer (unused in 1 byte transfer)
BYTE(0)	Lower data byte in 16-bit transfer or data byte in 8-bit transfer



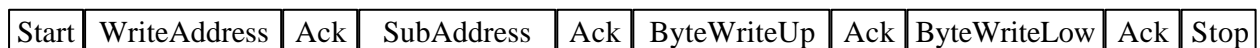
**Figure 9 – I2C master 8-bit write transfer**

Ex : 0x55 0x40 0x18 0x40 0x00 0x34 (UART COMMAND)



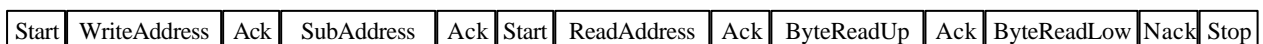
**Figure 10 – I2C master 8-bit read transfer**

Ex : 0x55 0x41 0x18 0x40 0x00 0x00 (UART COMMAND)



**Figure 11 – I2C master 16-bit write transfer**

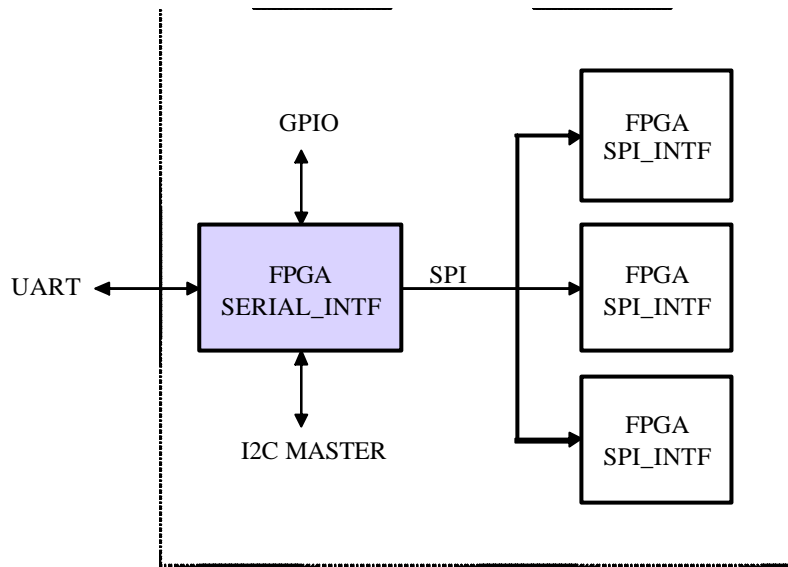
Ex : 0x55 0x42 0x18 0x40 0x12 0x34 (UART COMMAND)



**Figure 11 – I2C master 16-bit read transfer**

Ex : 0x55 0x43 0x18 0x40 0x00 0x00 (UART COMMAND)

## Multi FPGA SPI configuration



The SPI master interface of SERIAL\_INTF can be used to read and write 32-bit registers inside SPI slave FPGAs containing MVD SPI\_INTF core.

The UART serial line source only is supported (No transfer from I2C slave interface are available).

The SPI frame format is the following (With high bit sent first) :



### ADDR field : 7-bit

Bit(6..5)	'00'	Internal 32-bit bus
	'01'	Chip select register (Byte_0 for chip select number from 1 to 15)
	'1x'	Reserved

Bit(4..0) 32-bit register address shifted on the right of 2 bits (example : **0x04** register address of MVD\_DVBx core is **0x01** register address of UART)  
(Look at specific core documentation for registers mapping)

**DUMMY :**

8-bit clock time data is unused, the master must send the clock signal.

**BYTE :**

Byte\_3 is for register bit(31..24)

Byte\_2 is for register bit(23..16)

Byte\_1 is for register bit(15..8)

Byte\_0 is for register bit(7..0)

The UART master can use the following sequence to generate the transfers.

Write sequence example :

0x55, 0x50, 0x00, 0x00, 0x00, 0x01	to activate SPI chip select 1
0x55, 0x52, 0x00, 0x00, 0x00, 0x05	one byte transfer for write address
0x55, 0x58, 0x12, 0x34, 0x56, 0x78	four bytes transfer for write data
0x55, 0x52, 0x00, 0x00, 0x00, 0x00	one byte transfer for dummy byte
0x55, 0x50, 0x00, 0x00, 0x00, 0x00	to deactivate SPI all chip select

Read sequence example :

0x55, 0x50, 0x00, 0x00, 0x00, 0x02	to activate SPI chip select 2
0x55, 0x54, 0x00, 0x00, 0x85, 0x00	two bytes transfer for read address and dummy byte
0x55, 0x58, 0x00, 0x00, 0x00, 0x00	four bytes transfer for read data
0x55, 0x50, 0x00, 0x00, 0x00, 0x00	to deactivate SPI all chip select

Notice that the SPI chip select number is coded in Byte\_0 on bit(3..0) and can be from 1 to 15. The 0 value deselect all SPI slaves.

The address byte contains the transfer direction on bit 7 (0 for a write or 1 for a read).

## **6. Default initialization provided as option**

The SERIAL\_INTF core can be upgraded to implement static I2C or SPI configuration on external components.

## 7. Port definition

entity SERIAL\_INTF is

```
port (
  RST           : in  std_logic; -- reset
  RST_CPU      : in  std_logic; -- Reset pin for CPU
  CLK           : in  std_logic; -- CPU clock
  FREQUENCY    : in  std_logic_vector(15 downto 0);
  B115K        : in  std_logic;

  -- Internal Bus
  BUSADR       : out std_logic_vector(6 downto 0);
  BUSIN        : in  std_logic_vector(31 downto 0);
  BUSOUT       : out std_logic_vector(31 downto 0);
  RD           : out std_logic;
  WR           : out std_logic_vector(3 downto 0);
  CS           : out std_logic_vector(15 downto 0);

  -- GPIOs
  GPIO        : inout std_logic_vector(23 downto 0);

  -- SPI master serial bus
  I_CLK       : out std_logic;
  I_MOSI      : out std_logic;
  I_MISO      : in  std_logic;
  I_CS_N      : out std_logic_vector(15 downto 1);

  -- I2C master serial bus
  SCL         : out std_logic;
  SDA_IN      : in  std_logic;
  SDA_OUT     : out std_logic;
  SDA_EN      : out std_logic;

  -- I2C slave serial bus
  S_ADR       : in  std_logic_vector(2 downto 0);
  S_SCL       : in  std_logic;
  S_SDA_I     : in  std_logic;
  S_SDA_O     : out std_logic;
  S_SDA_EN    : out std_logic;

  -- UART
  RXD         : in  std_logic;
  TXD         : out std_logic
)
end SERIAL_INTF;
```

RST and RST\_CPU are asynchronous reset inputs. They can be connected together.  
The clock frequency must be in the range from 20MHz to 50MHz.

The FREQUENCY input vector must be fixed at :

$$\text{FREQUENCY} = \text{Integer value}(\text{CLK\_frequency(in Hertz)} / 1024)$$

Example :

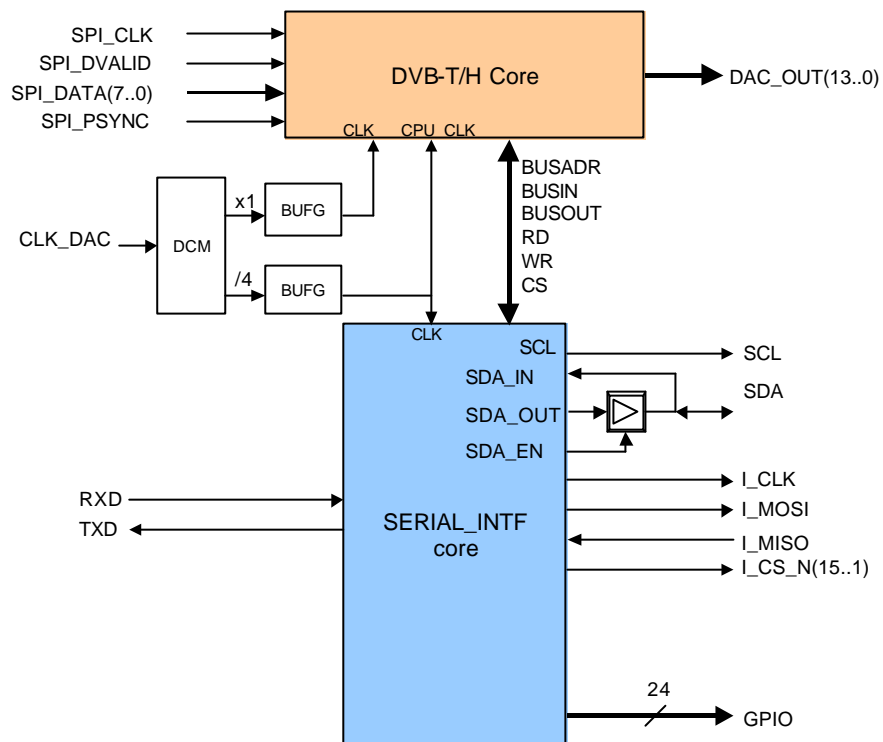
```
FREQUENCY <= conv_std_logic_vector(27000000/1024,16);
```

## 8. SERIAL\_INTF core files and deliverables

### 8.1. Core file

SERIAL\_INTF.ngc : Core netlist

### 8.2. Implementation examples

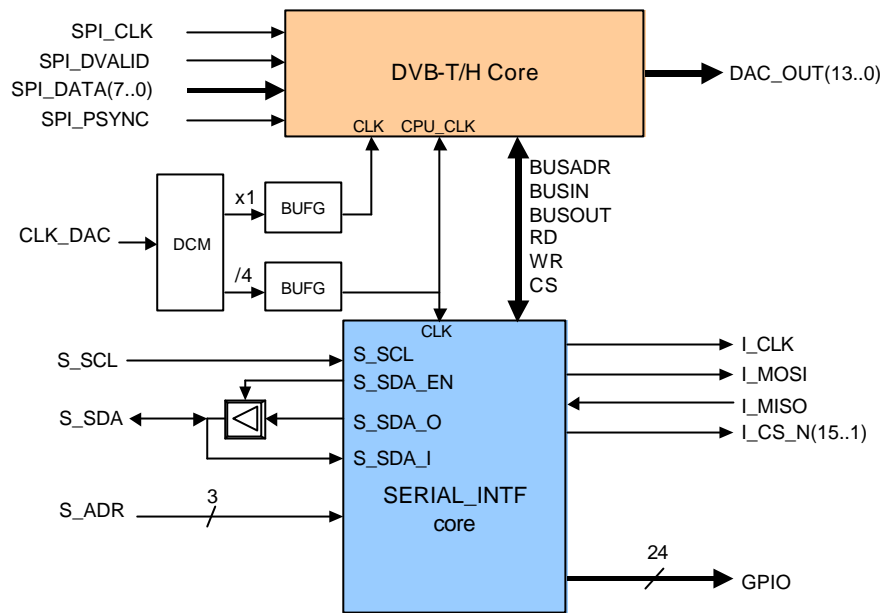


**Figure 12 – SERIAL\_INTF example with MVD DVB-T core configured via UART**

In this case, SDA implementation is as follow :

```
SDA <= SDA_OUT when SDA_EN = '1' else 'Z';
SDA_IN <= SDA;
```

Where SDA is the FPGA output pin.



**Figure 13 – SERIAL\_INTF example with MVD DVB-T core configured via I2C**

In this case, SDA implementation is as follow :

```
S_SDA <= S_SDA_O when S_SDA_EN = '1' else 'Z';
S_SDA_I <= S_SDA;
```

Where S\_SDA is the FPGA output pin.

## 9. Resource Utilization

	Slices	BRAMs	MULT or DSP48	BUFG
Spartan 3™	600	1	0	1
Virtex 4™	600	1	0	1
Virtex 5™	500	1	0	1

## 10. Use with MVD Cores

MVD cores : J83B, J83B 4-Channel, DVB-C, DVB-S, DVB-T/H.

## 11. Revision History

Edition/ Révision	Date	Modified Pages	Observations
2.0 A	12/2008	All	
2.0 B	06/2009	All 10 15	Changes of MVD Logo Add 2 bits shifting address requirement for MVD Core Address field Add virtex 5 ressources information Add example of MVD's companion core
2.0 C	03/2011	3 13 14 15	Add example of programming commands modify I <sup>2</sup> C interface for netlist compatibility Show I <sup>2</sup> C interface implementation example for master mode Show I <sup>2</sup> C interface implementation example for slave mode
2.0 D	11/2011	All 16	Removing of <a href="mailto:support_cores@mvd-fpga.com">support_cores@mvd-fpga.com</a> address Add this history revision paragraph