Selection and Reorder modules

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Harm Jan Pepping	ASTRON	1 August 2013
Controle / Checked: Eric Kooistra	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: **Handtekening / Signature** Andre Gunst	ASTRON	

Distribution list:

Group:	Others:
Andre Gunst Eric Kooistra Daniel van der Schuur	Gijs Schoonderbeek Sjouke Zwier Harro Verkouter (JIVE) Jonathan Hargreaves (JIVE) Salvatore Pirruccio (JIVE)

Document history:

Revision	Date	Author	Modification / Change
0.1	2013-08-01	Harm Jan Pepping	Creation
0.2	2014-01-22	Harm Jan Pepping	Added paragraph 3.5 explaining the create_settings method in the python driver.

Table of contents:

Terminology:

EOP	Start Of Packet
MM	Memory-Mapped
MOSI	Master Out Slave In
RAM	Random Access Memory
Signal Path	Time series signal
SISO	Source In Sink Out
SOP	Start Of Packet
SOSI	Source Out Sink In
Subband	Frequency signal

References:

1. $UNB/Firmware/modules/Lofar/ss/tb/vhd
2. Daniel van der Schuur, Uniboard Firmware Compilation Guide: Reference design unb_minimal, ASTRON-RP-1354, September 2012
3. $UNB/Firmware/modules/Lofar/ss/tb/python
4. $UPE/peripherals/ss_parallel.py

1 Introduction

The ss (subband select) library offers components to be used in applications that require on the fly selection and reordering of data. The name subband select suggests that the components only apply to a stream that contain subbands but that is not the case. In fact any time of data can be selected or reordered. The next paragraph shows the different operation types of the modules included in the ss library. Note that the modules presented in this document should be used as building blocks for selection and/or reorder solutions for any type of streaming application.

1.1 Mode of operation

1.1.1 ss_reorder

The ss_reorder unit composes output streams based on samples that are available on the input streams. A writable selection memory holds an output configuration for every clock sample within a frame. Figure 1 shows an example of ss_reorder instantiation, based on 4 inputs, 6 outputs and an input and output frame_size of 5. On the left the input frames are show and on the right the output frames are shown. The selection array in the symbol gives the output configuration (for all 6 outputs) for every clock cycle in the frame. The selection register must be interpreted from right to left.

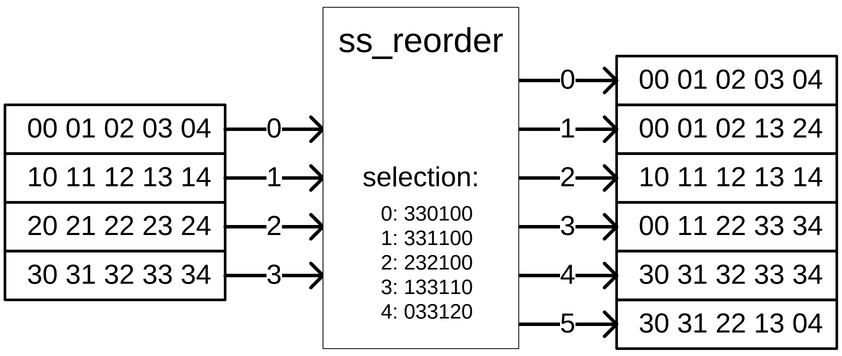

Figure 1 ss_reorder unit with 4 inputs and 6 outputs

1.1.2 ss

When considering the input frames and the output frames of Figure 1, it can be said that the ss_reorder unit facilitates the vertical movement from input to output. Horizontal movement however is facilitated by the ss unit. The ss unit uses a dual paged memory to store an incoming frame and simultaneously read the other page creating the output stream. The frame-size of the output stream can be different from the input frame_szie. The read order of the memory is defined by a writable and readable selection memory. Figure 2 shows an ss unit with an input frame and an output frame. The selection string in the symbol represents the content of the selection memory. Note that the ss unit works with a single stream.

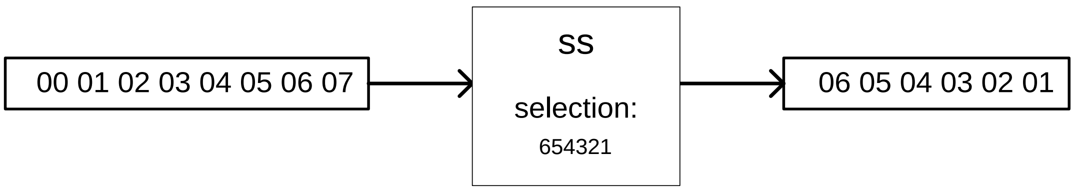

Figure 2 ss unit input frame_size = 8, output frame_size = 6

1.1.3 ss_wide

The ss_wide unit facilitates horizontal movement of samples within a frame on multiple streams. It can be considered as a set of stacked ss units. For each stream a unique selection memory is available. Note that it is not possible to move data from one stream to another, although each stream has it's own unique selection memory. Figure 3 shows a ss_wide unit with four inputs, input frame-size of 8 and output frame_size of 6. The content of the selection memory is given in the symbol.

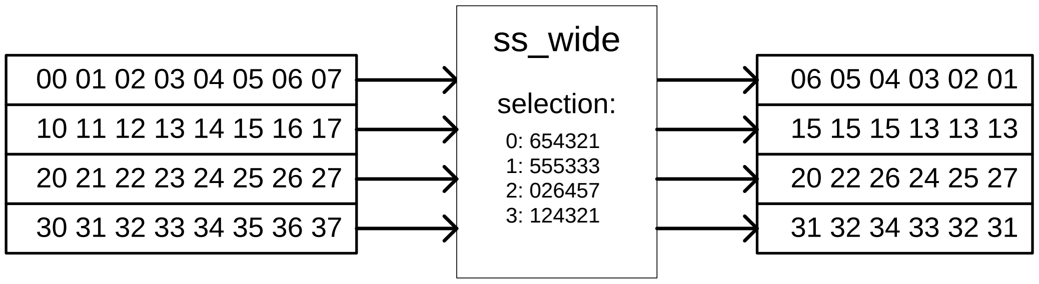

Figure 3 ss_wide unit with 4 inputs

1.1.4 ss_parallel

When ss_wide is combined with one or two instances of ss_reorder is becomes possible to make both horizontal and vertical movements in one go, which provides the ability to transform any input matrix to any desired output matrix. In this context a matrix is defined as a number of input streams (nof_inputs) by a number of samples in a frame (frame_size). This combination of ss_wide and ss_reorder units is the basic principal of the ss_parallel unit. An example is shown in Figure 4 that converts a 4x2 matrix to a 2x4 matrix.

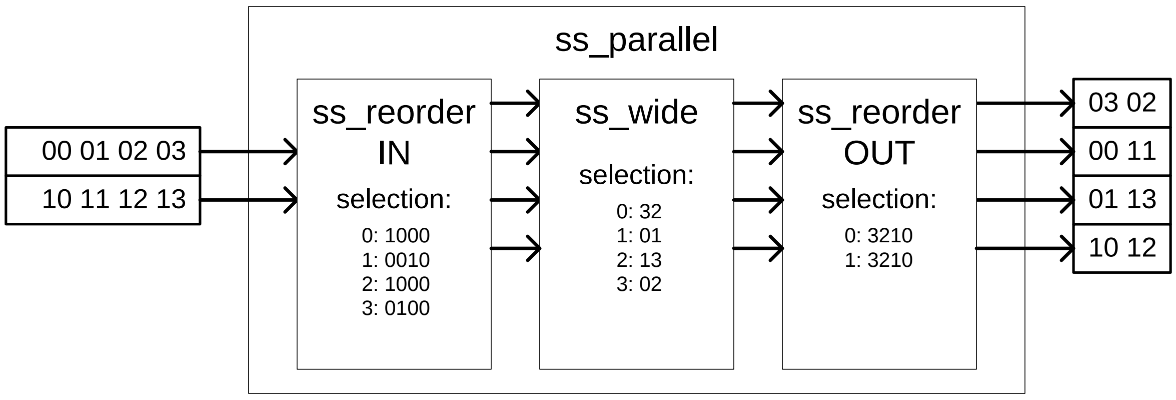

Figure 4 ss_parallel unit

2 Firmware interface

This chapter covers all firmware interface related topics for the modules of interest in the ss library. Every module is described in terms of parameters and interface signals.

2.1 Clock domains

All modules described have two clock domains: the mm_clk and the dp_clk domain. Where the mm_clk domain is used for interfacing with the host/control application and the dp_clk is the clock that drives the datapath.

2.2 ss_reorder

The parameters that define an instantiation of the ss_reorder module are listed in Table 1.

Generic	Type	Value	Description
dsp_data_w	NATURAL	16	Specifies the width of the input data for both real and imaginary part.
frame_size	NATURAL	256	The size of the incoming frames.
nof_inputs	NATURAL	8	The number of parallel streaming inputs.
nof_outputs	NATURAL	16	The number of parallel streaming outputs.
ram_init_file	STRING	“UNUSED”	String that holds the path and filename to an initialization file for the selection ram.
pipeline_in	NATURAL	1	Specifies the number of pipeline input registers.
pipeline_in_m	NATURAL	1	Pipeline registers rfor M-fold fan out.
pipeline_out	NATURAL	1	Specifies the number of pipeline output registers.

Table 1: ss_reorder parameters

The interface signals of the ss_reorder module are shown in Figure 5 and Table 2 lists the general specifications of these interfaces.

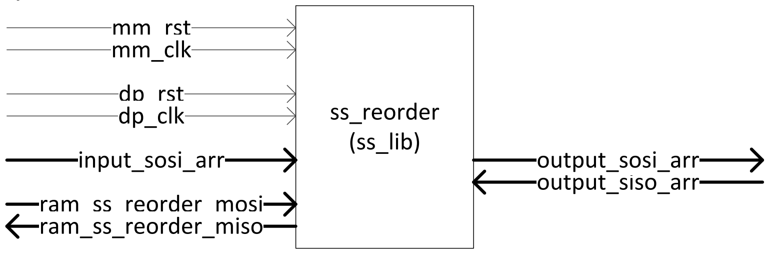

Figure 5: interface signals ss_reorder

Interface	Type	Size or Span	Description
input_sosi_arr	t_dp_sosi_arr	nof_inputs	Array of streaming inputs.
output_sosi_arr	t_dp_sosi_arr	nof_outputs	Array of streaming outputs.
output_siso_arr	t_dp_siso_arr	nof_outputs	Array of streaming outputs.
ram_ss_reorder_mosi	t_mem_mosi	frame_size	A mosi interface to read and write the selection settings.
ram_ss_reorder_miso	t_mem_miso	frame_size	A miso interface to read and write the selection settings.
dp_clk	std_logic	na	Datapath clock
dp_rst	std_logic	na	Datapath reset
mm_clk	std_logic	na	Memory mapped interface clock
mm_rst	std_logic	na	Memory mapped interface reset

Table 2: interface signals ss_reorder

2.3 ss

The parameters that define an instantiation of the ss module are listed in Table 3.

Generic	Type	Value	Description
dsp_data_w	NATURAL	16	Specifies the width of the input data for both real and imaginary part.
use_output_rl_adapter	BOOLEAN	FALSE	When TRUE adapt output Ready Latency to 1 else the output Ready Latency is equal to 2 which is fine if no flow control is needed.
nof_ch_in	NATURAL	512	The frame_size of the incoming stream.
nof_ch_sel	NATURAL	384	The frame_size of the outgoing stream. Value must be smaller than nof_c_in.
select_file_prefix	STRING	"UNUSED"	String containing path to initialization files for the selection ram.

Table 3: ss_reorder parameters

The interface signals of the ss module are shown in Figure 6 and Table 4 lists the general specifications of these interfaces.

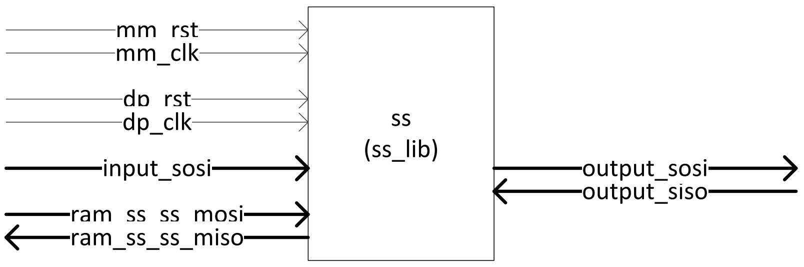

Figure 6: interface signals ss

Interface	Type	Size or Span	Description
input_sosi	t_dp_sosi	na	Streaming input.
output_sosi	t_dp_sosi	na	Streaming output.
output_siso	t_dp_siso	na	Streaming output.
ram_ss_ss_mosi	t_mem_mosi	nof_ch_sel	A mosi interface to read and write the selection settings.
ram_ss_ss_miso	t_mem_miso	nof_ch_sel	A miso interface to read and write the selection settings.
dp_clk	std_logic	na	Datapath clock
dp_rst	std_logic	na	Datapath reset
mm_clk	std_logic	na	Memory mapped interface clock
mm_rst	std_logic	na	Memory mapped interface reset

Table 4: interface signals ss

2.4 ss_wide

The parameters that define an instantiation of the ss_wide module are listed in Table 5.

Generic	Type	Value	Description
wb_factor	NATURAL	4	The wideband factor, defining the number of parallel input and output streams.
dsp_data_w	NATURAL	16	Specifies the width of the input data for both real and imaginary part.
nof_ch_in	NATURAL	512	The frame_size of the incoming streams.
nof_ch_sel	NATURAL	384	The frame_size of the outgoing streams. Value must be smaller than nof_c_in.

select_file_prefix	STRING	"UNUSED"	String containing path to initialization files for the selection rams.

Table 5: ss_wide parameters

The interface signals of the ss module are shown in Figure 7 and Table 6 lists the general specifications of these interfaces.

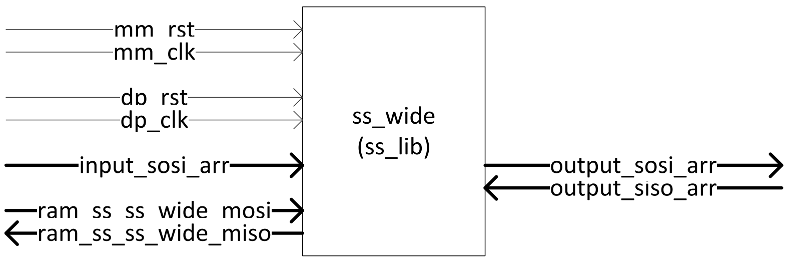

Figure 7: interface signals ss_wide

Interface	Type	Size or Span	Description
input_sosi_arr	t_dp_sosi	na	Array of streaming inputs.
output_sosi_arr	t_dp_sosi	na	Array of streaming outputs.
output_siso_arr	t_dp_siso	na	Array of streaming outputs.
ram_ss_ss_wide_mosi	t_mem_mosi	nof_ch_sel	A mosi interface to read and write the selection settings for all ss units within ss_wide.
ram_ss_ss_wide_miso	t_mem_miso	nof_ch_sel	A miso interface to read and write the selection settings for all ss units within ss_wide.
dp_clk	std_logic	na	Datapath clock
dp_rst	std_logic	na	Datapath reset
mm_clk	std_logic	na	Memory mapped interface clock
mm_rst	std_logic	na	Memory mapped interface reset

Table 6: interface signals ss_wide

2.5 ss_parallel

The parameters that define an instantiation of the ss_parallel module are listed in Table 7.

Generic	Type	Value	Description
dsp_data_w	NATURAL	16	Specifies the width of the input data for both real and imaginary part.
frame_size_in	NATURAL	256	The size of the incoming frames.
frame_size_out	NATURAL	192	The size of the outgoing frames.
nof_inputs	NATURAL	8	The number of parallel streaming inputs.
nof_outputs	NATURAL	16	The number of parallel streaming outputs.
nof_internals	NATURAL	16	String that holds the path and filename to an initialization file for the selection ram.

Table 7: ss_parallel parameters

The interface signals of the ss_parallel module are shown in Figure 8 and Table 8 lists the general specifications of these interfaces.

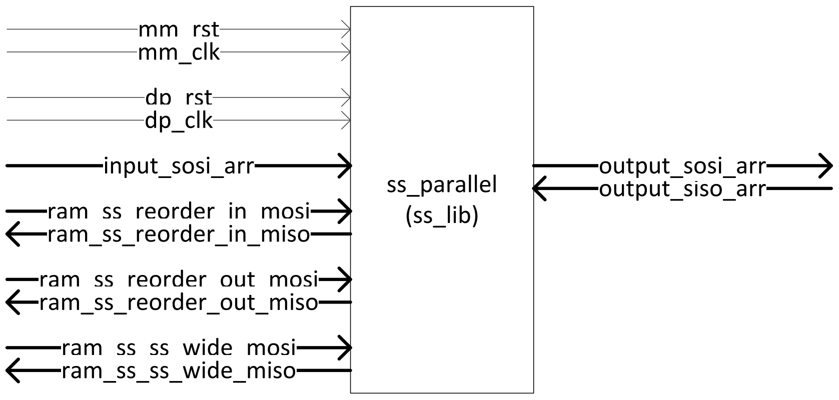

Figure 8: interface signals ss_parallel

Interface	Type	Size or Span	Description
input_sosi_arr	t_dp_sosi_arr	nof_inputs	Array of streaming inputs.
output_sosi_arr	t_dp_sosi_arr	nof_outputs	Array of streaming outputs.
output_siso_arr	t_dp_siso_arr	nof_outputs	Array of streaming outputs.
ram_ss_reorder_in_mosi	t_mem_mosi	frame_size_in	A mosi interface to read and write the selection settings for the input reorder unit.
ram_ss_reorder_in_miso	t_mem_miso	frame_size_in	A miso interface to read and write the selection settings for the input reorder unit.
ram_ss_reorder_out_mosi	t_mem_mosi	frame_size_out	A mosi interface to read and write the selection settings for the output reorder unit.
ram_ss_reorder_out_miso	t_mem_miso	frame_size_out	A miso interface to read and write the selection settings for the output reorder unit.
ram_ss_ss_wide_out_mosi	t_mem_mosi	nof_internals* frame_size_out	A mosi interface to read and write the selection settings for the ss unit.
ram_ss_ss_wide_out_miso	t_mem_miso	nof_internals* frame_size_out	A miso interface to read and write the selection settings for the ss unit.
dp_clk	std_logic	na	Datapath clock
dp_rst	std_logic	na	Datapath reset
mm_clk	std_logic	na	Memory mapped interface clock
mm_rst	std_logic	na	Memory mapped interface reset

Table 8: interface signals ss_parallel

3 Software interface

This chapter describes the software interface for all ss modules.

3.1 Register span ss_reorder

The register span of the ss reorder module consists of a number of 32-bit registers. The exact number of registers depends of the input frame size, the nof_inputs and the nof_outputs. The bitwidth of a selection word is defined as follows: bitwidth_selection_word = ceil_log2(nof_inputs)*nof_outputs. In case it is larger than 32-bit the selection word is divided over multiple 32-registers. The register definition for an ss_reorder unit with nof_input = 8 and nof_output = 4 is shown in Figure 9. The register consists of 4 times 3 bits that define the connection between input ports and output ports.

11 .. 9	8 .. 6	5 .. 3	2 .. 0
Output 3	Output 2	Output 1	Output 0

Figure 9 Register definition ss_reorder

Table 9 shows the register map for a ss_reorder with input frame_size = 4, nof_inputs = 8 and nof_outputs = 12. So each selection_word is 36-bit and therefore the number of registers per selection word is 2.

Name	Address (words)	Size (bits)	Read/ Write	Description
selection_0_lsb	0x0	32	r/w	Register that holds the 32 least significant bits of the selection word for sample 0 from input frame.
selection_0_msb	0x1	4	r/w	Register that holds the 4 most significant bits of the selection word for sample 0 from input frame.
selection_1_lsb	0x2	32	r/w	Register that holds the 32 least significant bits of the selection word for sample 1 from input frame.
selection_1_msb	0x3	4	r/w	Register that holds the 4 most significant bits of the selection word for sample 1 from input frame.
selection_2_lsb	0x4	32	r/w	Register that holds the 32 least significant bits of the selection word for sample 2 from input frame.
selection_2_msb	0x5	4	r/w	Register that holds the 4 most significant bits of the selection word for sample 2 from input frame.
selection_3_lsb	0x6	32	r/w	Register that holds the 32 least significant bits of the selection word for sample 3 from input frame.
selection_3_msb	0x7	4	r/w	Register that holds the 4 most significant bits of the selection word for sample 3 from input frame.

Table 9 ss_reorder register (ram) span

3.2 Register span ss

Table 10 shows the register span for the ss module with the following parameters: nof_chan_in=256 and nof_ch_sel=8. The size of the register is defined by ceil_log2(nof_chan_in). The number of registers is defined by nof_ch_sel.

Name	Address (words)	Size (bits)	Read/ Write	Description
selection_0	0x0	8	r/w	Address that specifies sample 0 in the output frame.
selection_1	0x1	8	r/w	Address that specifies sample 1 in the output frame.
selection_2	0x2	8	r/w	Address that specifies sample 2 in the output frame.
selection_3	0x3	8	r/w	Address that specifies sample 3 in the output frame.
selection_4	0x4	8	r/w	Address that specifies sample 4 in the output frame.
selection_5	0x5	8	r/w	Address that specifies sample 5 in the output frame.
selection_6	0x6	8	r/w	Address that specifies sample 6 in the output frame.
selection_7	0x7	8	r/w	Address that specifies sample 7 in the output frame.

Table 10 ss register (ram) span

3.3 Register span ss_wide

The register span of an ss_wide module is simply a concatenation of a series of ss modules register spans. The span of each individual ss module is always a power of 2. Table 11 shows the register span for an ss_wide unit with wb_factor=4, nof_ch_in =4, nof_ch_sel=3. Note the jumps in the addresses when going from one channel to the next channel.

Name	Address (words)	Size (bits)	Read/ Write	Description

selection_0_0	0x0	2	r/w	Address that specifies sample 0 in the output frame of output channel 0.
selection_0_1	0x1	2	r/w	Address that specifies sample 1 in the output frame of output channel 0.
selection_0_2	0x2	2	r/w	Address that specifies sample 2 in the output frame of output channel 0.
selection_1_0	0x4	2	r/w	Address that specifies sample 0 in the output frame of output channel 1.
selection_1_1	0x5	2	r/w	Address that specifies sample 1 in the output frame of output channel 1.
selection_1_2	0x6	2	r/w	Address that specifies sample 2 in the output frame of output channel 1.
selection_2_0	0x8	2	r/w	Address that specifies sample 0 in the output frame of output channel 2.
selection_2_1	0x9	2	r/w	Address that specifies sample 1 in the output frame of output channel 2.
selection_2_2	0xA	2	r/w	Address that specifies sample 2 in the output frame of output channel 2.
selection_3_0	0xC	2	r/w	Address that specifies sample 0 in the output frame of output channel 3.
selection_3_1	0xD	2	r/w	Address that specifies sample 1 in the output frame of output channel 3.
selection_3_2	0xE	2	r/w	Address that specifies sample 2 in the output frame of output channel 3.

Table 11 ss_wide register (ram) span

3.4 Register span ss_parallel

The register span for an ss_parallel unit is a compilation of two ss_reorder spans and one ss_wide span. Each register span has its own memory interface and therefor the span format of the ss_reorder_in and the ss_reorder_out span can be found in paragraph 3.1 and the span definition for the ss_wide part is in paragraph 3.3.

3.4.1 Span width ss_reorder_in

The register span of the ss_reorder_in part consists of a number of 32-bit registers. The exact number of registers depends of the frame_size_in, the nof_inputs and the nof_internals. The bitwidth of a selection word is defined as follows: bitwidth_selection_word = ceil_log2(nof_inputs)*nof_internals. In case it is larger than 32-bit the selection word is divided over multiple 32-registers.

3.4.2 Span width ss_reorder_out

The register span of the ss_reorder_out part consists of a number of 32-bit registers. The exact number of registers depends of the frame_size_in, the nof_internals and the nof_outputs. The bitwidth of a selection word is defined as follows: bitwidth_selection_word = ceil_log2(nof_internals)*nof_outputs. In case it is larger than 32-bit the selection word is divided over multiple 32-registers.

3.4.3 Span width of ss_wide

The total span of the ss_wide module is defined as nof_internals*ceil_log2(frame_size_out).

3.5 Python control for ss_parallel

In order to obtain the desired data at the output of the ss_parallel unit an algorhythm is developed that determines the settings for ss_parallel(ss_reorder_in, ss_wide and ss_reorder_out) based on the desired output. The python peripheral used to control the ss_parallel unit has a method called create_settings. The create_settings method uses two arguments to generate the return values result, Rin, Dram, Dsel, Rout and Errout. The function of ss_parallel is to create an output matrix based on a given input matrix. Both in- and output matrixes have ‘time’ and ‘number of streams’ as dimensions as shown in Figure 4.

3.5.1 Din and Dout

Argument Din represents the input matrix and it consists of elements that represent the coordinates of each element. Every element has to have a unique coordinate. From the example in Figure 4 the following Din is determined:

[[[0,0],[0,1],[0,2],[0,3]],
[[1,0],[1,1],[1,2],[1,3]]]

The Dout argument represents the desired output matrix based on the available elements from Din. From the same example it follows that the Dout argument should be as follows:

[[[0,3],[0,2]],
[[0,0],[1,1]],
[[0,1],[1,3]],
[[1,0],[1,2]]]

Note that Dout can only contain elements (coordinates) that appear in Din. It is also possible to select a certain element multiple times.

3.5.2 Method create_settings

Based on the provided Din and Dout the create_settings method will try to find the settings for the ss_parallel unit. This includes settings for ss_reorder_in, ss_wide and ss_reorder_out. Create_settings uses four internal matrixes to keep track of the progress: Rin, Rout, Dram and Dsel. Rin and Rout hold the settings for both the ss_reorder_in and ss_reorder_out unit. Dram reflects the content of the dataram in the ss_wide unit and Dsel contains the settings for the selection ram in the ss_wide units. This is displayed in Figure 10.

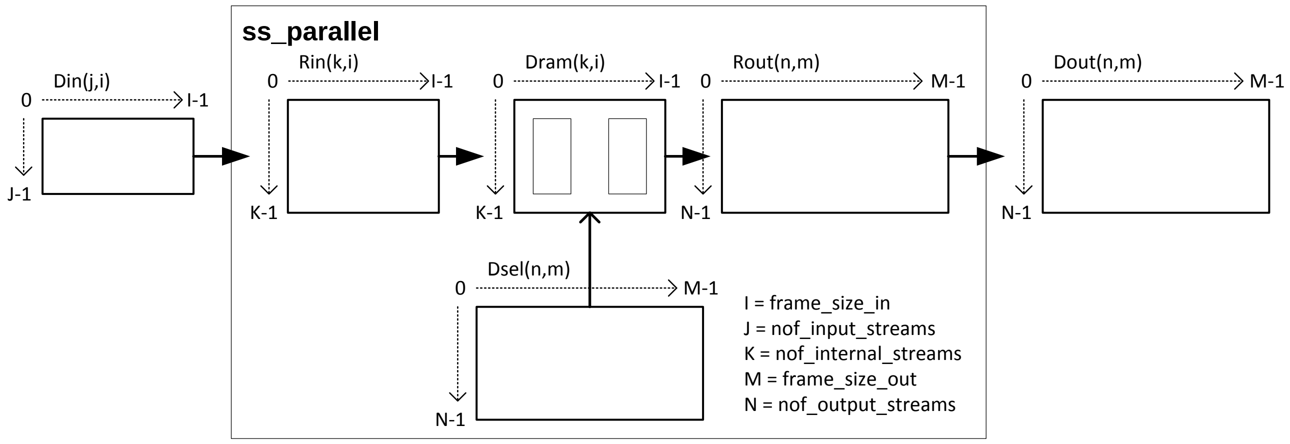

Figure 10 Matrixes in create_settings

The create_settings initializes the matrixes Rin and Rout with all zeros and matrixes Dram and Dsel with the value -1. Then it iterates over the columns (M) and within each column it iterates over the number of outputs (N). Within every iteration it first checks if the required Dout element is already placed in one of the rows of the Dram. This is done with the method “locate_value”. If the value already exists in the Dram the settings for Dsel and Rout can be derived from the found Dram location. The Dsel location must hold the column of the found location and the Rout setting is based on the found row number. This is displayed in the upper row of Figure 11.

In case the current value is not found in the Dram ii must be found in Din. When the value is found in Din a free and valid row in the Dram is needed. When this is found the settings for Rin, Dsel and Rout can be derived as shown in the second row in Figure 11

.

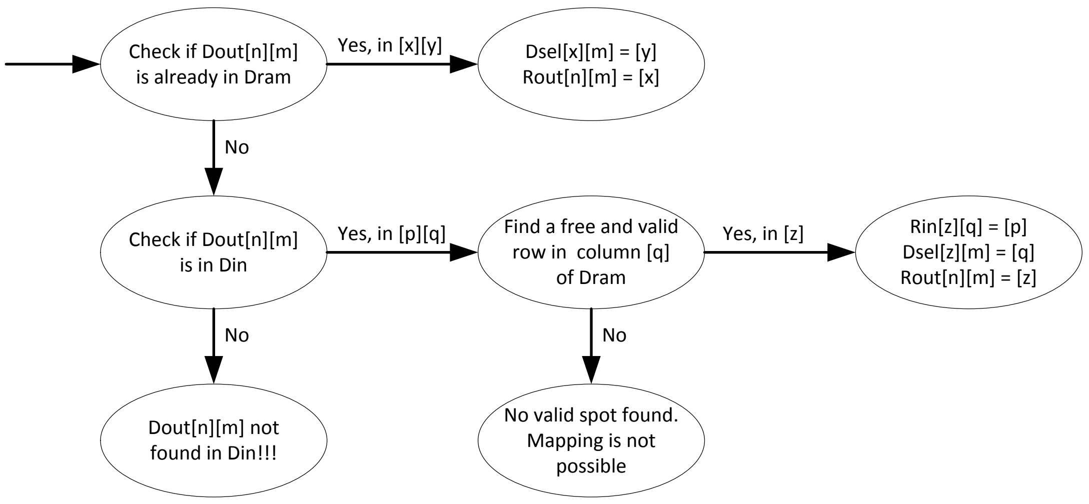
Check if Dout[n][m] is already in Dram
Yes, in [x][y]
Dsel[x][m] = [y]
Rout[n][m] = [x]
No
Check if Dout[n][m] is in Din
Yes, in [p][q]
Find a free and valid row in column [q] of Dram
Yes, in [z]
Rin[z][q] = [p]
Dsel[z][m] = [q]
Rout[n][m] = [z]
No
No
Dout[n][m] not found in Din!!!
No valid spot found. Mapping is not possible

Figure 11 create_settings iteration

The python code looks like this and can be found in [4].

```
Rin  = self.init_array(self.nofInternals, self.frameSizeIn,  0)
Rout = self.init_array(self.nofOutputs,   self.frameSizeOut, 0)
Dram = self.init_array(self.nofInternals, self.frameSizeIn,  [-1,-1])
Dsel = self.init_array(self.nofInternals, self.frameSizeOut, -1)

for m in range(self.frameSizeOut):
    for n in range(self.nofOutputs):
        location = self.locate_value(Dout[n][m], Dram, 0, 0)
        if(location != [-1, -1]):
            Dsel[location[0]][m] = location[1]
            Rout[n][m] = location[0]
        else:
            location = self.locate_value(Dout[n][m], Din, 0, 0)
            if(location == [-1, -1]):
                print "Value " + str(Dout[n][m]) + " not found in Din..."
            else:
                mem_row = self.find_free_spot(Dram, Dout[n][m], location[1], Dout, m, 0)
                if mem_row != -1 :
                    Rin[mem_row][location[1]] = location[0]
                    Dsel[mem_row][m] = location[1]
                    Rout[n][m] = mem_row
                 else:
                    print "Not Found, not able to map"
```

4 Implementation

4.1 ss_reorder

The ss_reorder unit is based on the common_select_m_symbols unit and a dual port selection memory as shown in Figure 10. A counter is used to generate the read addresses for the selection words. The common_select_m_symbols unit requires a concatenated input format where real and imaginary data of all inputs is concatenated to one wide input word. The output is similar. The address counter increases as long as valid data enters and at the end of a frame the counter is cleared using the eop signal. At the output the data of the common_select_m_symbols unit is merged with the pipelined delayed sosi signals form the input. The selection memory can

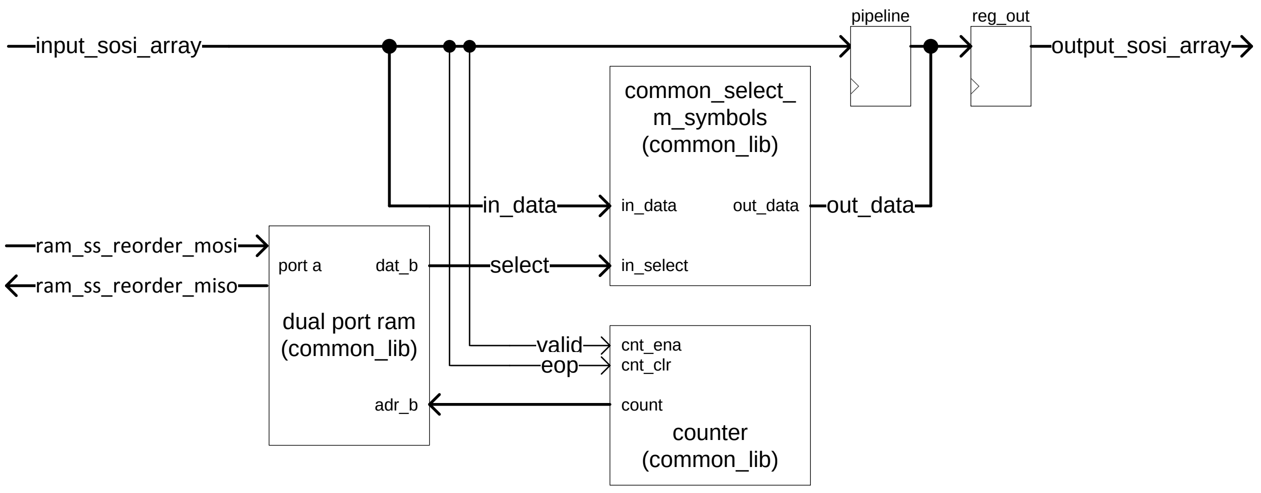

Figure 12 ss_reorder block diagram

4.2 ss

The ss unit was originally designed for the Lofar project and later on adapted to the Uniboard firmware design environment. A block diagram is shown in Figure 11. The ss_store unit converts the incoming sosi stream to a memory mapped mosi interface in order to store the data in the first page of the data memory. When the last word of a frame is written the ss_retreive unit is notified by assertion of the store_done signal. The ss_store unit will then continue writing in the second page. Meanwhile the ss_retreive unit reads out the

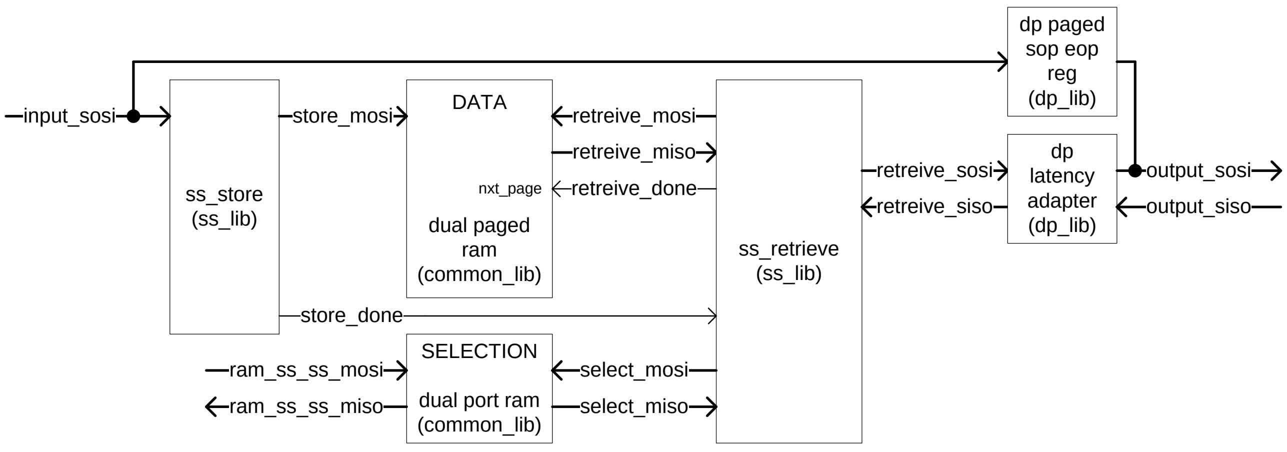

Figure 13 ss unit block diagram

data from the first page based on the addresses in the selection memory. The retrieved data is composed into a streaming output and then merged with the other sosi fields (eop, sop, bsn). The dp_latency adapter is used to adapt the ready latency in case that is required. The selection memory holds the addresses of the data to be read from the data memory. Note that the order of the addresses in the selection memory determines the order of the data in the outgoing frame. The selection memory can be read and written via the ram_ss_ss_mosi interface.

4.3 ss_wide

Two or more ss units placed in parallel are considered an ss_wide unit. Figure 12 shows an ss_wide unit where two ss units are combined. A common_mem_mux unit is used to combine the memory mapped interface to one single memory mapped interface.

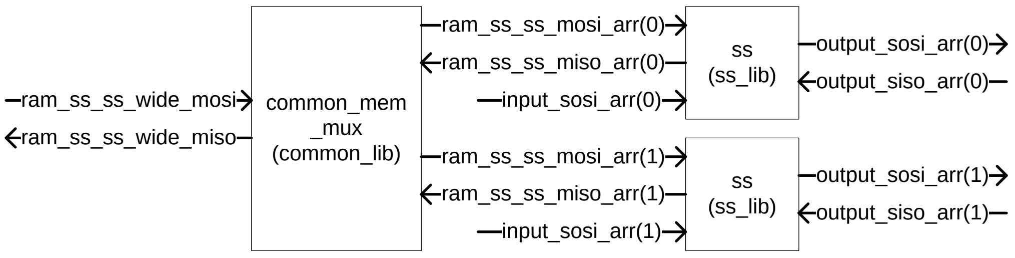

Figure 14 ss_wide unit block diagram

4.4 ss_parallel

The ss_parallel unit combines two ss_reorder blocks and an ss_wide unit as show in in Figure 13.

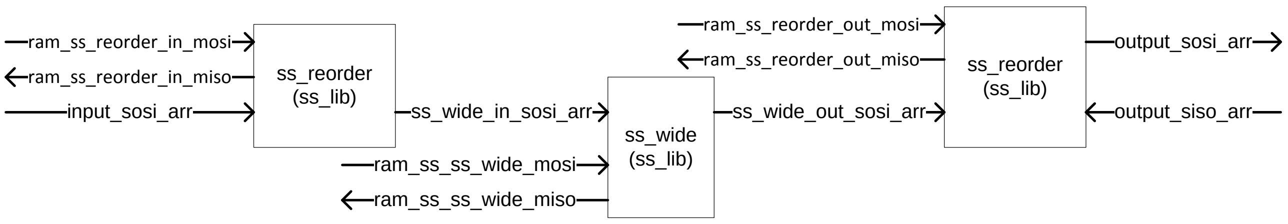

Figure 15 ss_parallel block diagram

5 Verification

5.1 ss_reorder

Verification of the ss_reorder module is performed using the co-simulation approach that is also used in the unb_minimal design. See [2] for more detail. The VHDL testbench can be found in [1] and is called tb_mmf_ss_reorder.vhd. The accompanying python test case can be found in [3] and is called tc_ss_reorder.py. The simulation is started by running ss_reorder.py that is also located in [3]. The script automatically starts Modelsim. The script should be called as follows:

```
python ss_reorder.py --hold
```

The --hold arguments ensures that the konsole that outputs the ModelSim is not closed when the simulation has finished.

5.2 ss

Simulation of the ss module is done using a “normal” VHDL testbench. The testbench can befound in [1] and is called tb_ss. The tb_ss for the ss module verifies the data. It is also possible to simulate multiple instances with different settings for the ss module. This is done using the tb_tb_ss testbench that can also be found in [1]. The header of the vhd file contains methods for proper usage.

5.3 ss_wide

The ss_wide module is simulated with a VHDL testbench called tb_ss_wide that can be found in [1]. The header of the vhd file contains methods for proper usage.

5.4 ss_parallel

Verification of the ss_parallel module is performed using the co-simulation approach that is also used in the unb_minimal design. See [2] for more detail. The VHDL testbench can be found in [1] and is called tb_mmf_ss_parallel.vhd. The accompanying python test case can be found in [3] and is called tc_ss_parallel.py. The simulation is started by running ss_parallel.py that is also located in [3]. The script automatically starts Modelsim. The script should be called as follows:

```
python ss_parallel.py --hold
```

The script can be run with default settings. The default settings are the settings that are applied to the bn_filterbank design for the Apertif project.

6 Validation

Validation of the selection and reorder modules is performed in several designs. In the bn_filterbank image the ss_parallel module is instantiated.

7 Appendix – list of files

7.1 Firmware VHDL

All VHDL source files that are used for the st module can be found in the following directory:

$UNB/Firmware/modules/Lofar/ss/src/vhdl

The next table gives an overview of the VHDL source files:

VHDL File	Description
ss.vhd	Subband select toplevel file for a single stream. Contains ss_store, ss_retreive, selection memory and data memory.
ss_parallel.vhd	Toplevel for a parallel subband select on multiple streams, based on ss_wide and ss_reorder.
ss_reorder.vhd	Contains reorder function for a parallel stream.
ss_retreive.vhd	Building block for the ss unit for reading selected data back from the data memory.
ss_store.vhd	Building block for the ss unit for writing incoming data to the data memory.
ss_wide.vhd	A parallel configuration based on multiple ss units.

7.2 Testbench

The testbench files for simulation are in the following directory:

$UNB/Firmware/modules/Lofar/ss/tb/