LLRF controller implementation

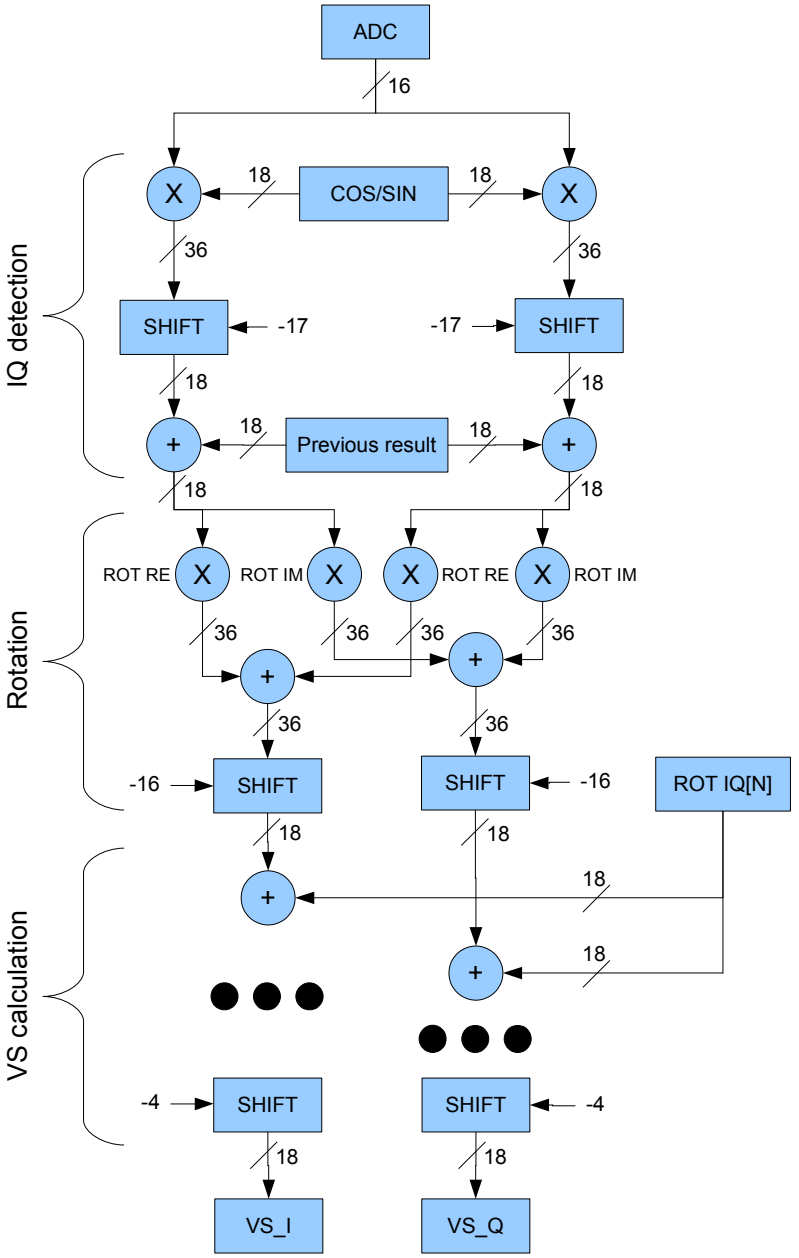
Wojciech Jalmuzna

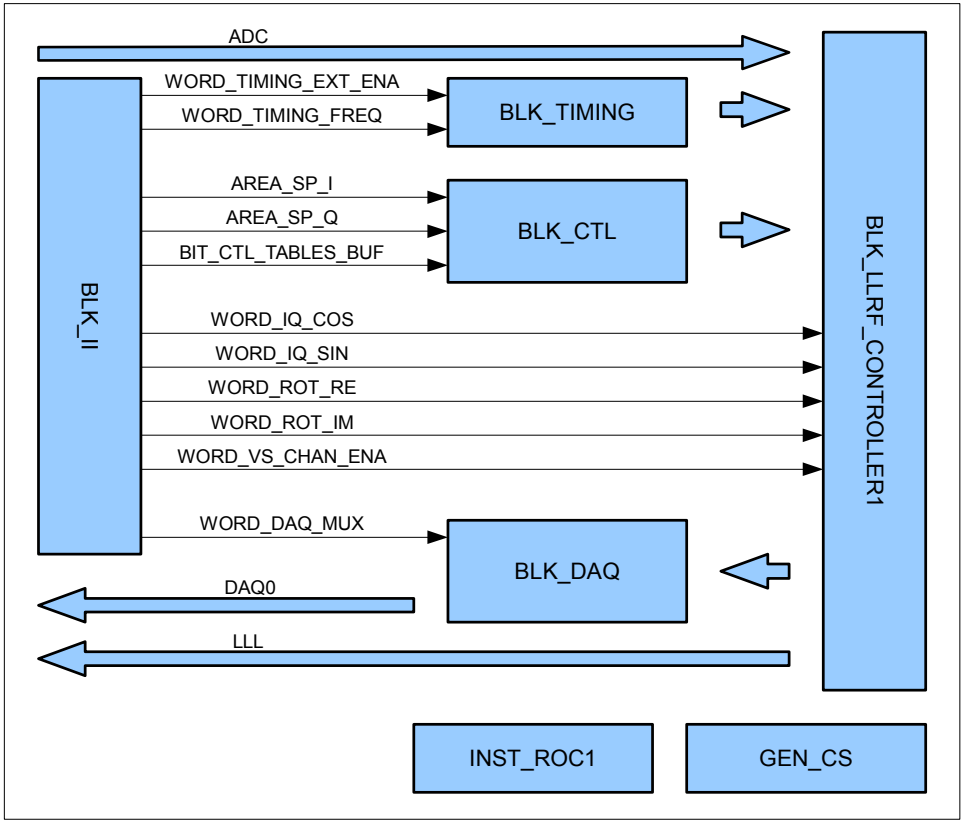
Source code documentation

- Evaluated doxygen
- Prepared custom scripts for doc generation

- --! - doxygen style comments used
- Special script for address space documentation

Block diagrams





Element name	Width	Count	Type	Description
WORD_FIRMWARE	32	1	unsigned	Firmware Identification Register. It contains hexadecimal string “52454741”, which converted to ASCII gives “REGA”
WORD_COMPILATION	32	1	unsigned	Compilation Identification Register, which is incremented every time firmware modification is done.
WORD_TIMING_FREQ	32	2	unsigned	Frequency vector for internal event generation. This register configures timing subsystem to generate internal events. Element 0 corresponds to PUSLE event. Element 1 is currently not used. Values are given in clock cycles.
WORD_TIMING_EXT_ENA	2	1	unsigned	Selection bits for internal/external events. Value '1' indiocates that external event should be used. '0' switches to internal event generation. As for FREQ registers, bit '0' corresponds to PULSE event etc.
WORD_IQ_COS	18	5	signed	Table of cosine values for IQ detectors. The same table is used for every channel.
WORD_IQ_SIN	18	5	signed	Table of cosine values for IQ detectors. The same table is used for every channel.
WORD_ROT_RE	18	10	signed	Real part of rotation matrix coefficients. Each element in a vector corresponds to single channel.
WORD_ROT_IM	18	10	signed	Imaginary part of rotation matrix coefficients. Each element in a vector corresponds to single channel.
WORD_VS_CHAN_ENA	10	1	unsigned	Enable bits for vector sum components
WORD_DAQ_MUX	8	6	signed	Multiplexer for DAQ channels (see DAQ TYPE 2 manual [1])
AREA_SP_I	18	8192	signed	Setpoint Table I. Array generated by control tables module. Each entry corresponds to single clock cycle after PULSE event. Table index 0.
AREA_SP_Q	18	8192	signed	Setpoint Table Q. Array generated by control tables module. Each entry corresponds to single clock cycle after PULSE event. Table index 1.
BIT_CTL_TABLES_BUF	2	1	unsigned	Double buffering vector is used while uploading new set of tables. Each bit switches buffers one corresponding table according to its position (bit 0 – index 0, bit 1 – index 1)

SIS8300 Resources

- More than 65 multipliers – 48 available
- Up to 125 MHz operation
- DAQ limitations (200 MHz memory clock – 128 bit bus)

Server changes (in respect to VME)

- Additional status registers (TRG, CLK)
- LLL status – each broadcasts its id so we know interconnections
- Register “polarity” changes – more intuitive way.

Server changes

- Flexible DAQ configuration
- Event select (each system has trigger on different line)
- DAQ test signal

Server cannot be limitation !

Additional modifications

- CORDIC iterations for AP calc. (currently 12)
- One Cordic module for all channels

Discussion Items

- How to time-align daq on different boards
- Latency measurement
- How to decimate data to 9 MHz
- Do we want to stick to P.Gessler ideas ? (LLL packages, standard register set, etc.)

UTC resource estimation

Input Stage Resource Estimation

Name	Number of multipliers	Registers
Virtual probe calculation	4x16 channels = 64	16x36 = 576
Low latency rotation	4x32 channels = 128	32x36 = 1152
Partial vector sum rotation	4x1	36
Vector sum rotation	4x1	36
Cavity limit detection	4x1	36
Drift calibration	4x16 channels	576
Sum	**263**	**2412**

Feedback Stage Resource Estimation

Name	Number of multipliers	Registers
P-Controller	2	36
MIMO	20	360
BBF	16	72
Sum	**38**	**468**

Output Stage Resource Estimation

Name	Number of multipliers	Registers
Output rotation matrix	4	36
Linearization	8	36
Various corrections	16	72
Sum	**28**	**144**

Overall resource estimation

	Number of multipliers	Registers
Sum	329	3024
Intermediate stages		~5000
Input parameters		~5000
Available [occupied %]	**640 [51%]**	**58800 [22%] (under est.)**
Driver part	+0%	+10%
Total	**51%**	**33%**

Additionally almost 60% of Internal Block RAM is occupied by driver part. Control tables should be implemented in external DDR memory.

What is ready and what needs to be done ?

- Functionality as for ACC1 demo is implemented

<u>Additional work</u>

- PCIe DMA transfer to memory
- MIMO integration
- DSP module integration
- Forward - reflected connection
- Detuning calc ?
- What else ??

Thank you