FP7 - Grant Agreement no. 283393 - Radionet3

Project name:	Advanced Radio Astronomy in Europe
Funding scheme:	Combination of CP & CSA
Start date:	01 January 2012
Duration:	48 months

Deliverable 8.13

Revised Firmware Design Document Beam Former

Due date of deliverable:	June 2015
Actual date of deliverable:	July 2015
Deliverable Leading Partner:	MPIfR Bonn

An European project supported within the 7th framework programme

SEVENTH FRAMEWORK PROGRAMME

1 DOCUMENT INFORMATION

Document name: Uniboard^2 Beam Former Firmware Design Document
Type: Document
Revision: 2.0
WP: 8
Authors: Guenter Knittel
MPIfR Report: MPIfR-UB2-07/2015

1.1 Dissemination level

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	

1.2 Document history

Revision	Date	Author	Modification / Change
1.0	31 Jan 2014	G. Knittel	Initial Version
2.0	31 Jul 2015	G. Knittel	Revised Version

1.3 Distribution list

ASTRON: Andre Gunst, Eric Kooistra, Sjouke Zwier, Daniel van der Schuur, Harm Jan Pepping
JIVE: Arpad Szomoru, Jonathan Hargreaves, Salvatore Pirruccio, Sergei Pogrebenko, Paul Boven, Harro Verkouter
UMAN: Aziz Ahmedsaid, Ben Stappers
INAF: Gianni Comoretto
BORD: Benjamin Quertier, Alain Baudry, Stephane Gauffre
UORL: Cedric Dumez-Viou, Rodolphe Weber, Nicolas Grespier
MPG: Guenter Knittel, Gundolf Wieching, Bernd Klein, Udo Beckmann

1.4 Terminology

10GBASE-KR: 10Gigabit/s Ethernet interface for backplanes, using one differential signal pair in each direction (four PCB traces)
ADC: Analog to Digital Converter
ALM: Adaptive Logic Module
BF: Beamformer
bps: Bits per second
BW: Bandwidth
Channel: Frequency band, output of the (polyphase) filterbank
CORDIC: (for COordinate Rotation DIgital Computer), algorithm for computing trigonometric functions
cplx: complex, denoting a complex number
CXM: Complex multiplication, complex multiplier
DSP: Digital Signal Processing
FD: Full-duplex, bi-directional link offering full data rate in each direction simultaneously
Firmware: Collection of control codes close to the hardware, less likely and easy to change than software
FFT: Fast Fourier Transform
FPGA: Field Programmable Gate Array
VHDL: Very high-speed integrated circuit Hardware Description Language
Im: Imaginary component of a complex number
I/O: Input/Output
IP: Intellectual Property
LAB: Logic Array Blocks
LUT: Look-up Table
MLAB: LAB that also can be used as RAM

QSFP+:	SFP for 40Gb Ethernet
RAM:	Random Access Memory
PAF:	Phased Array Feed
PFB:	Polyphase Filterbank
Re:	Real component of a complex number
SFP:	Small Form-factor Pluggable transceiver, an optical transceiver module
SFP+:	SFP for 10Gb Ethernet
SP-FP:	Single-precision floating-point (number)
Subband:	Frequency band, output of the (polyphase) filterbank
Transceiver:	High-speed I/O-circuit with bi-directional data transfer, using serial differential signaling
XGMII:	10Gigabit/s Media-Independent Interface

1.5 References

[1] W. C. Barott, O. Milgrome, M. Wright, D. MacMahon, T. Kilsdonk: „Real-Time Beamforming Using High-Speed FPGAs at the Allen Telescope Array“, (2011). Department of Electrical, Computer, Software, & Systems Engineering - Daytona Beach. Paper 2.

[2] G. Comoretto: „Deliverable 8.4 - Uniboard^2 Digital Receiver Firmware Design Document“, INAF report 01/2014

[3] G. Comoretto, G. Knittel, A. Russo: “Uniboard Pulsar Receiver Design document” (2012)

[4] P. E. Dewdney: „SKA1 System Baseline Design“, SKA-TEL-SKO-DD-001, 2013-03-12

[5] G. Schoonderbeek: “Deliverable 8.2 - Hardware Design Document”, ASTRON Doc. Nr. ASTRON-TN-040 1.0 (2014)

[6] A. Szomoru: “UniBoard^2 Work Package description”, RadioNet3 283393 (2011)

[7] A. Szomoru: “Beamforming specifications“, „ADC-specifications“, personal communication

[8] https://en.wikipedia.org/wiki/CORDIC

[9] https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html#family-table, visited on July 6 2015

[10] G. Schoonderbeek, ASTRON, email communication

[11] suggested by G. Wieching

[12] UniBoard2 Schematic, Sjouke Zwier / Gijs Schoonderbeek, 12/12/2014

[13] John Bunton, CSIRO, “PAF Beamformer”

Table of Contents

2 INTRODUCTION

The presented beamformer is a narrow-band beamformer operating on frequency channels of around 1MHz bandwidth. The specifications are oriented towards SKA phase 1 requirements [4], [7]: 384MHz observing bandwidth, 256 dual-polarization receivers (512 antenna signals), and 64 beams. The fundamental mathematical operation that it performs is

$$y(n) = \sum_{m=0}^{M-1} w_m \cdot x_m(n) \tag{1}$$

where $x_m(n)$ is the signal from the *m-th* antenna at sample time $n*T_0$, w_m the weightfactor for the *m-th* antenna, and $y(n)$ the beam sample at time $n*T_0$. All of the quantities y, x, and w are complex numbers. In (1), a single-beam system (many-to-one) is described.
A system computing several beams $b = 0 \ldots B-1$ (many-to-many) can then be described as

$$y_b(n) = \sum_{m=0}^{M-1} w_{m,b} \cdot x_m(n) \tag{2}$$

Finally, if the antenna signals have been split into a number of frequency channels $c = 0 \ldots C-1$ the system can be described as

$$y_{b,c}(n) = \sum_{m=0}^{M-1} w_{m,b,c} \cdot x_{m,c}(n) \tag{3}$$

The beamformer is polarization-agnostic, meaning, all 512 antenna signals can be used for the computation of a given beam. Polarization can be included by setting the weigths appropriately. For an estimate of the error arising from the channel bandwidth see [1].

The system must maintain a total of $M*B*C$ weightfactors. In this example design this amounts to 12,582,912 complex numbers. The precision is set to 19 bits per component, as this is the maximum width of the hard-wired multipliers on the Arria-10 FPGAs. Thus, a total of 478,150,656 bits must be provided for the weightfactors.

The total number of complex multiplications per second is given by $384*10^6*512*64 = 1.2582912*10^{13}$. The FPGAs which are planned to be used for a low-cost version of the Uniboard^2 provide a total of 759 hard-wired macros for complex multiplication. For a complex multiplication $(a + jb) * (c + jd)$, operand width for a and b can be up to 18 bits, whereas the width for c and d can be up to 19 bits.

For cost reasons this study uses the slowest production version of the FPGA, which allows the complex multipliers to be used at up to 360MHz. The complex multipliers are grouped into sub-arrays of 16 units, so that 7 complex multipliers remain unused per FPGA. In order to meet the computing requirements, a total of 48 FPGAs on 12 Uniboards are used. Then the minimum clock frequency is

$$1{,}2582912 \cdot 10^{13} / (48 \cdot 752) = 348{,}6 MHz \tag{4}$$

Thus, using a clock frequency of 360MHz in conjunction with shallow buffers will provide some headroom for compensating any data rate variations.

Each FPGA needs to process 384 / 48 = 8 frequency channels across the 512 antenna signals. We assume that channelization is done by polyphase filterbanks, and that the complex output samples have a width of 16 bits per component. Thus, data rate into each beamformer FPGA is $8*512*10^6*32 = 1.31072*10^{11}$bit/s. We assume conservatively that a net data rate of 10Gb/s can be achieved per backplane lane. Thus, each beamformer FPGA needs at least 14 such interfaces. Each beamformer-FPGA provides a total of 72 such backplane interfaces, so this is not a limitation. However, the actual number of interfaces depends on the architecture of the polyphase filterbanks, and on how the 512 antenna signals are connected to the filterbank-FPGAs.

These particular considerations shall now be used to derive architecture and size of the beamformer. Because of the multitude of hardware constraints it is very hard to give general rules for scaling the system. A change of the desired performance or availability of more powerful chips might very well require the design of a completely different architecture.

2.1 Design Targets

At this stage in the project the primary goal is to meet the peak performance requirements and to achieve a high resource utilization as well as a high compute efficiency. Aspects related to power consumption, and „green measures“ are currently not in the focus. These studies will be conducted during later stages.

2.2 Topics not considered in this Document

The document only describes the data paths and arithmetic units, without going into details regarding control units and control interfaces. This basic infrastructure is assumed to simply „be there“ and work in the expected way. It will only be described in general terms.

Also not covered in this document are considerations regarding minimum operand precision across the many buses in the design. For the time being, the maximum precision as provided by the available hardware resources (multipliers, RAM etc) is used for the operands. This also defines the width of register banks and other units. In many cases this kind of arithmetic precision might not be necessary.

Likewise, savings that arise from unused antennas, or weightfactors being zero or negligible (in a potential PAF application), are not considered in this architectural draft.

This again relates to power consumption, and to green measures for power reduction. These studies will be included in the final deliverable.

3 System Architecture

3.1 Filterbank

Since there are 512 antenna signals, the system needs to have 512 filterbanks. It is assumed that the actual implementation employs polyphase filterbanks with a channel bandwidth of about 1MHz. It might be the case that the observing bandwidth is larger than 384MHz [7], in which case it is assumed that 384 channels can be selected arbitrarily out of the larger set of filterbank output channels.

It is further assumed that the polyphase filterbank generates complex samples with 16 bits for both the real and imaginary components. This particular bit width is not a strict requirement, however, the components may not have more than 18 bits due to hardware limitations (multiplier port width).

The design of this filterbank is not part of this workpackage, so we cannot make any statements about the required hardware resources. From other projects [2], [3] we conservatively estimate that one FPGA can implement 32 filterbanks. Then, a total of 16 FPGAs are needed, or four Uniboards. In this case the signal processing system would take the shape of a standard midplane system with eight Uniboards to the left and eight to the right of a midplane (also called backplane).

In an SKA scenario it is most likely that the signal processing system is located in a central facility, and that the digitizers are located close to the receivers (dishes). Data is then transmitted in digital form over optical fibers. In the first instance the digitizer signals should be connected to those Uniboards that implement the filterbanks. In case there are not enough interface resources, digitizer signals can also be connected to the other Uniboards and passed on to the filterbank units via the backplane.

A block diagram showing the filterbank, its partitioning and the output distribution is given in Figure 1.

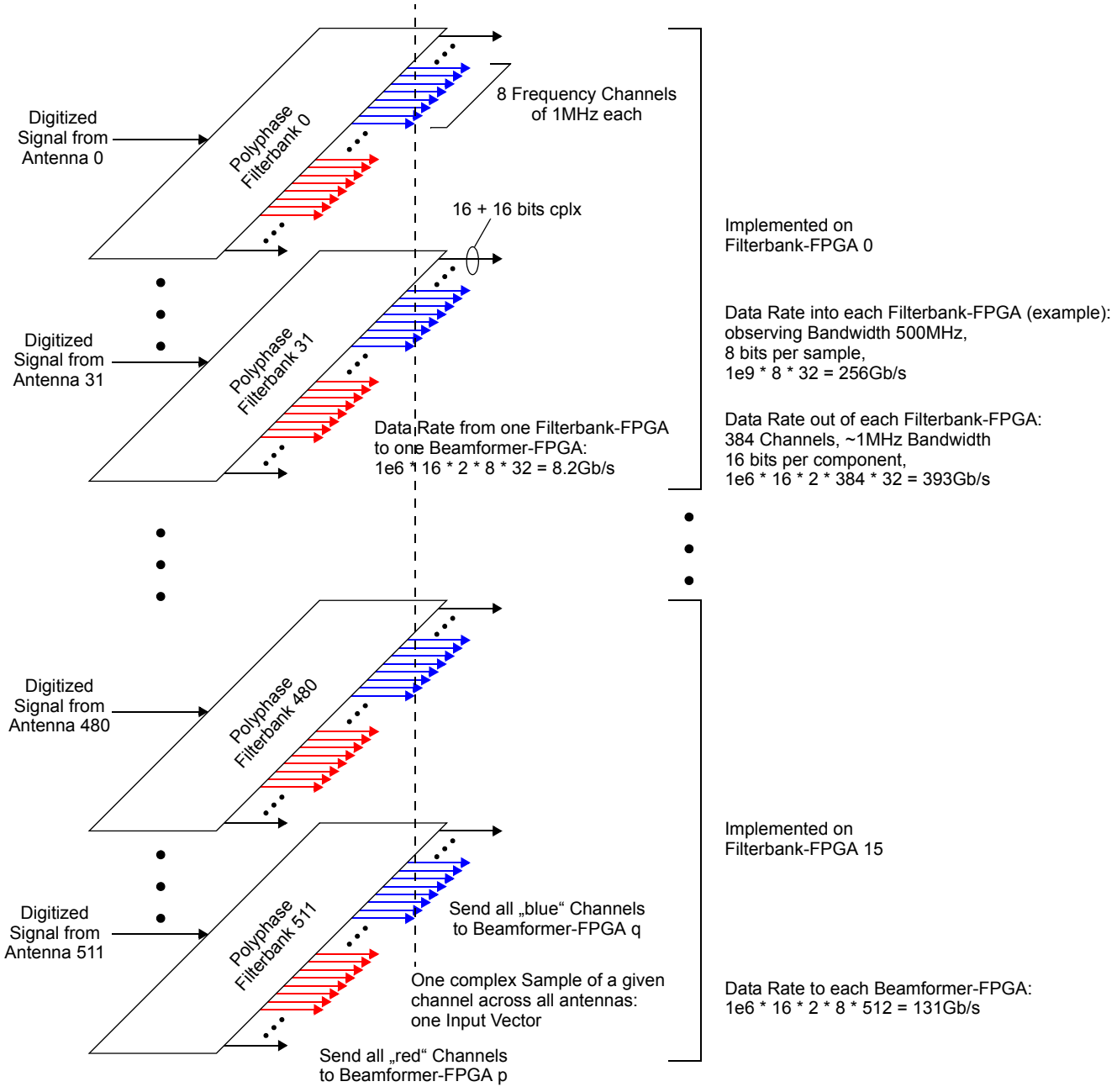

Figure 1: General Filterbank Architecture

3.2 Backplane

A decision was made to not include any local mesh on the Uniboard^2, but to provide all interconnects on a backplane. Again, design of the backplane is not part of this workpackage. Therefore we list only basic requirements about the interconnect structure in order to support the beamformer, assuming a polyphase filterbank as described above.
According to the data rates given in Figure 1, any filterbank-FPGA needs to connect to any beamformer-FPGA via one 10Gbit/s backplane lane (transceiver channel). This occupies 48 transceivers on any filterbank-FPGA. Conversely, any beamformer-FPGA is connected to any filterbank-FPGA, which occupies 16 transceivers on any beamformer-FPGA. This basically constitutes the required interconnect structure on the backplane. We assume that the results of the beamformer (beam samples) are output via optical links local to each beamformer-FPGA to a GPU-cluster or similar external device.
A sketch of the required interconnect structure is shown in Figure 2. See also [5] for planned interconnect capabilities.

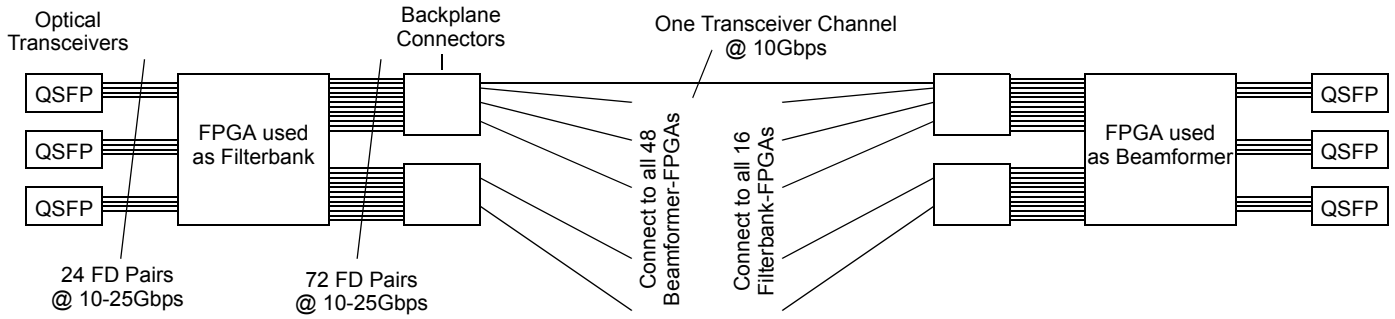

Figure 2: Interconnect Structure

3.2.1 Revised Architecture

The Uniboard^2 architecture has been modified for the production version as shown in Figure 3 [12]. Each FPGA now has only 48 FD pairs towards the backplane because of routing constraints. However, this number is just sufficient for the beamformer architecture presented here. The remaining 24 FD pairs are used to construct a ring network, which might be used for sample distribution (as an example see Figure 21).

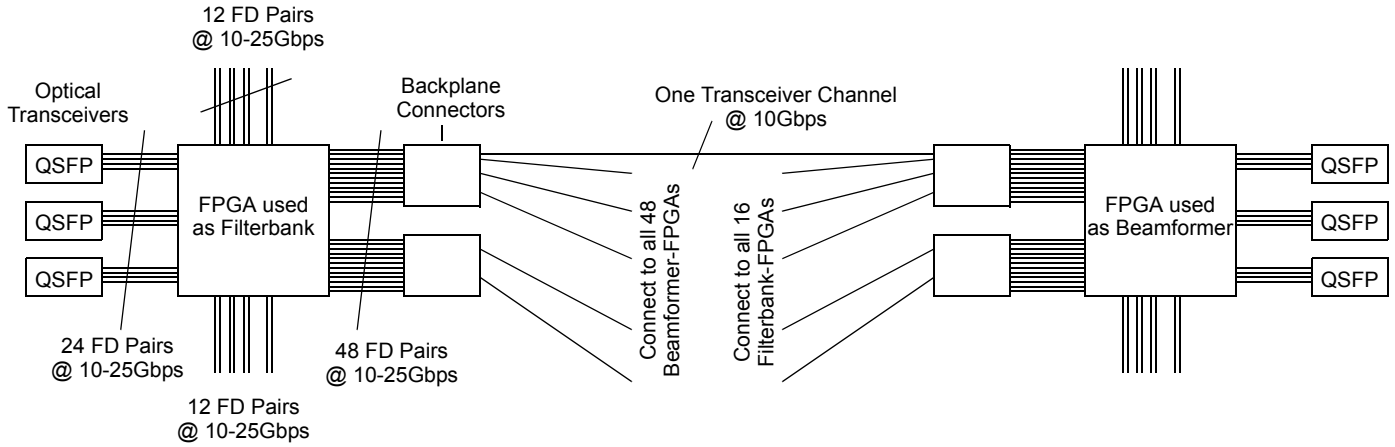

Figure 3: Interconnect Structure (Revised Version)

4 ARCHITECTURE OF THE BEAMFORMER FPGA

All 48 beamformer-FPGAs have an identical design. The specific role of an individual FPGA is defined by its place on the backplane, the programmable set of weight factors, and by further programmable parameters such as output destination.
The beamformer-FPGA consists mainly of six parts:

- sample input stage,
- input fan-out register tree,
- beamformer unit,
- weightfactor memory,
- weightfactor fan-out register tree, and
- output stage.

In addition to these units there are control and synchronization units at various places, whose functionality will be described only in general terms. Also, the general infrastructure for writing all programmable parameters including weightfactors will be described in a later document.

4.1 Chip Resources

For a better understanding of the following sections we shall give a coarse overview of available hardware resources. The specific FPGA that is planned to be used for the production version is an Altera Arria-10 GX 1150 (order code Altera 10AX115U4F45I3SG).

4.1.1 Logic

Configurable logic elements on an Arria-10 FPGA include so-called LABs (Logic Array Blocks) and MLABs (Logic Array Blocks that also can be used as RAM). Both are further divided into so-called ALMs (Adaptive Logic Modules). Each ALM provides two LUTs, a two-bit carry-chain for building adders, and 4 flipflops (registers). The LUTs allow computation of any boolean expression of up to 7 variables. The FPGA provides a total of 854,400 LUTs, and 1,708,800 single-bit registers.

4.1.2 Arithmetic

Each FPGA provides a total of 1,518 so-called „DSP Blocks“, each providing two 18×19 bit multipliers among other things. These multipliers are implemented as specialized hardware units. The design tools allow two DSP Blocks to be used as one complex multiplier (CXM). Thus, there are 759 CXMs on the chip.

4.1.3 Memory

Each FPGA provides a total of 2,713 memory blocks of 20Kbits each. These specialized hardware units are called „M20K“. They can be organized as 512×40 bits. Thus, for example, one entry can hold one complex weightfactor with 19 bits for both the real and imaginary component.
M20K blocks can be used as dual-port memory with one read and one write port. Reading from an entry that is simultaneously being written to has some implications, so it is best to avoid this situation.

4.2 Block Diagram

The block diagram of a beamformer-FPGA is shown in Figure 4.

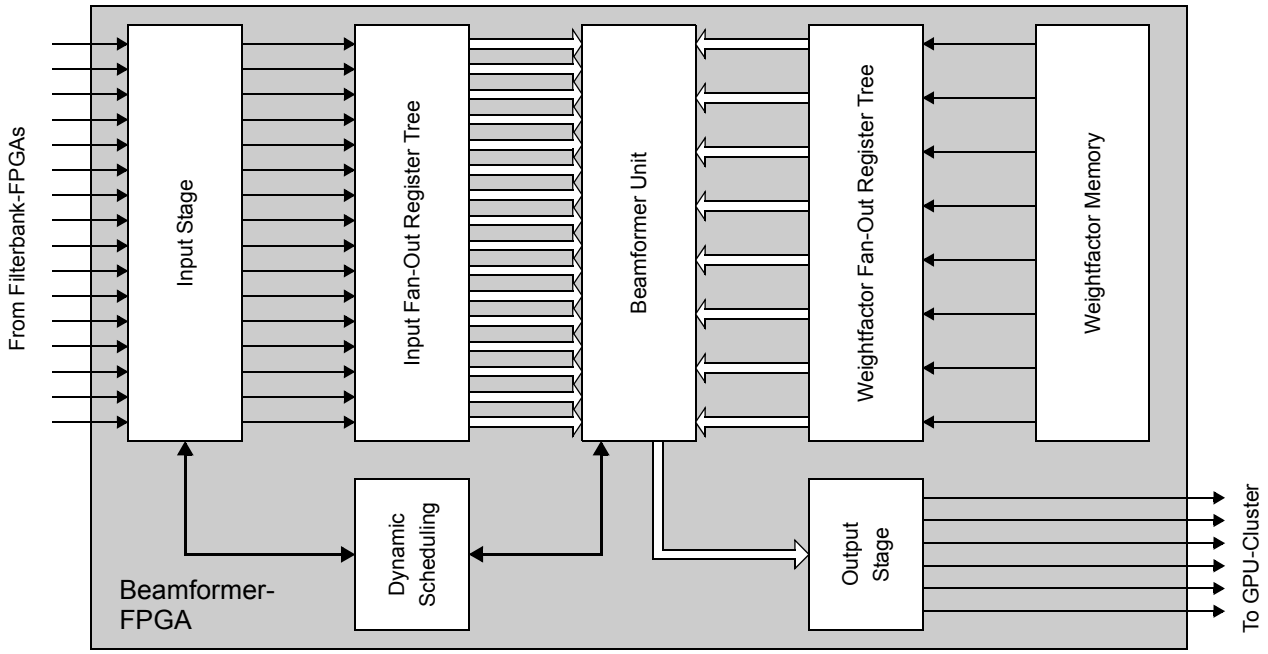

Figure 4: Beamformer-FPGA Block Diagram

4.3 Input Stage

The input stage consists of 16 identical interfaces, each connecting to one filterbank-FPGA. Each interface can be implemented as a 10GBASE-KR transceiver channel (10Gbps Ethernet for backplanes), and should provide a data rate of about 10Gbps. In case of 10GBASE-KR, the interface towards the FPGA-logic is called XGMII (10 Gigabit Media Independent Interface) and is implemented as a parallel interface transfering 8 bytes in parallel. The clock rate at this stage is 156.25MHz.

The samples enter a FIFO memory which should be large enough to provide buffer space for a number of data packets. For safety purposes we assume a buffer capacity of 16KByte per interface, for a total of 256KByte across the input stage. FIFO memories are typically implemented using the Altera design tools. These tools report a memory consumption of 7 M20K blocks per FIFO, for a total of 112 such blocks across the input stage. This represents about 4% of the available resources.

Towards the remaining logic on the beamformer-FPGA, a 32-bit data bus is needed. Thus, the read-port of each FIFO is carried out as 32-bit interface, and data width reduction is handled by internal control logic. Besides this, the FIFOs also provide translation between the I/O- and the system clock domains (156.25MHz to 360MHz).

Accordingly, one complex sample (16 + 16 bits) is read per read cycle (clock) from each FIFO, or 16 complex samples are read per clock across the entire input stage.

The samples need to be transmitted by the filterbank-FPGAs in a specific order, which is shown in Figure 5. See also Figure 1 for sample arrangement. Sample origin is given by [Antenna Number; Frequency Channel Number].

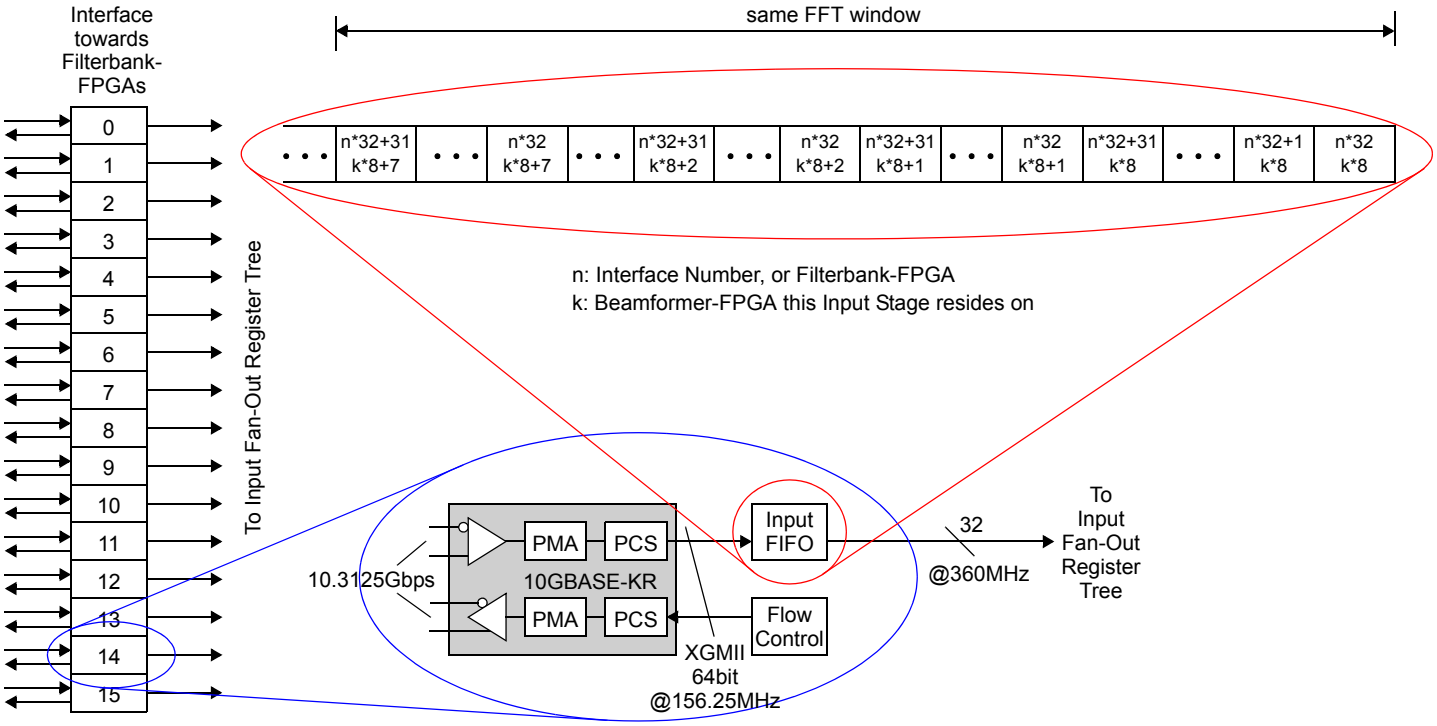

Figure 5: Input Stage Block Diagram

4.4 Input Fan-Out Register Tree

The FIFO of interface *n* feeds the data to CXM unit *n* in each beamformer core (see section 4.5.4). There are 47 beamformer cores. Thus, the fan-out is 47 as well, and to reduce wiring delays and meet the clock rate the input data is distributed by a fan-out register tree. We use a two-stage 6×8 register arrangement. Thus, this distribution stage consumes 27,648 flipflops. This is 1.6% out of 1,708,800 available flipflops. The arrangement is sketched out in Figure 6. Note that the latency introduced by this structure is of no concern.

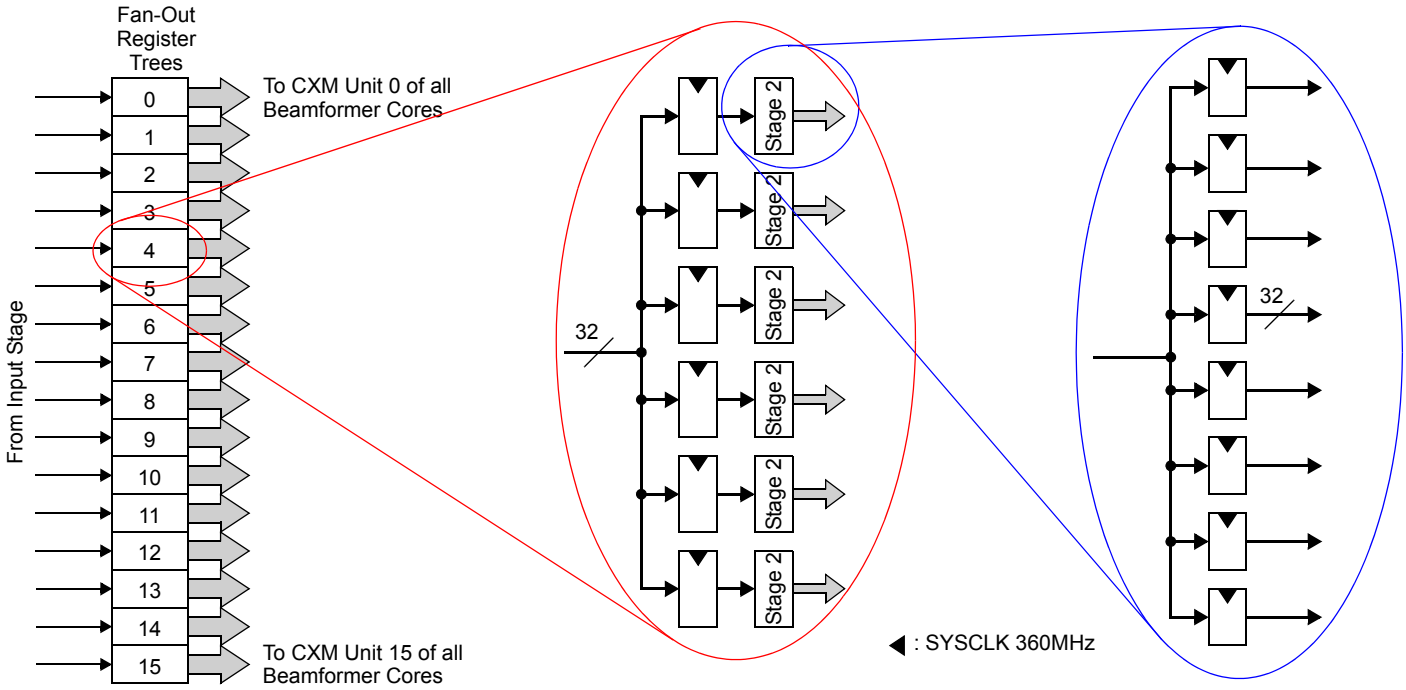

Figure 6: Input Fan-Out Register Tree

4.5 Beamformer Unit

The beamformer unit is divided into an array of independent beamformer cores in order to make better use of the available hardware resources. Each beamformer core operates on one sample across all 512 antennas in one frequency channel and generates all 64 beam samples from this input data set. We call this collection of input samples an *input vector* (see Figure 1).

A beamformer core uses 16 complex multipliers (CXM) in parallel. Accordingly, there are 47 beamformer cores on the chip, with a total of 7 CXMs, or 14 DSP Blocks going unused. This represents a resource utilization of 99.1%.

A beamformer core will only start operating when all 512 complex samples are available. Thus, we need one M20K memory block in front of each CXM for data buffering. An input vector will only occupy 32 out of 512 entries of each M20K block, so there is ample room for buffering and data rate adjustments. A total of 752 M20K blocks are needed for this purpose, or about 27.7% of the available memory blocks. The weightfactors, on the other hand, are streamed in real-time from the weightfactor memory and don‘t need deep buffers. This will be explained in detail in the following section.

4.5.1 Theory of Operation

Whenever a beamformer core has enough free storage space for an input vector it will issue a work package request to the scheduler (see Figure 4). The scheduler will prioritize all requests and send an input vector to the selected beamformer core. This input vector is simply the next input vector in the input FIFOs and can be from any of the eight frequency channels. A read across all input FIFOs will yield 16 complex samples, which are written into the 16 memory blocks of the beamformer core in parallel. 32 such transfers are needed to move a complete input vector to the selected beamformer core.

Using such scheme it cannot be predicted which frequency channel any given beamformer core will receive, nor the exact point in time the complete input vector will be available. Still the beamformer core is required to run at 100% efficiency, i.e., no single cycle may be spent idle when input data is available.

For this purpose the weightfactor memory (see section 4.6) issues a constant stream of weightfactors for all eight frequency channels. The beamformer core utilizes a set of multiplexers to select the proper frequency channel. Most importantly, however, the stream of weightfactors also include the index of the current set of weightfactors, which consists of antenna and beam number. Using this index, a beamformer core can select the corresponding time samples from the memories and start processing immediately, even if it tapped the weightfactor stream in mid-sequence. It is obvious that the sum of products in (3) can be computed in any order.

A control unit local to each beamformer core only needs to determine when an input vector has been completed, which happens after 2048 clocks. It will then tag the corresponding partial sums as being the last ones, and switch to the next input vector in the sample memories, if present.

The „inner loop“ iterates over beams, which reliefs timing requirements of the accumulator at the output of a beamformer core.

4.5.2 Beamformer Core Block Diagram

The beamformer unit consists of 47 identical beamformer cores. As explained above, this uneven number is given by the available chip resources and might be different for other FPGA variants. Nevertheless, the architecture should allow any number of beamformer cores to be operated at peak performance. A beamformer core consists of the following parts:

- weightfactor and index multiplexer,
- memory and CXM stage,
- adder tree, and
- accumulator.

These units are detailed in the following sections. The block diagram is shown in Figure 7.

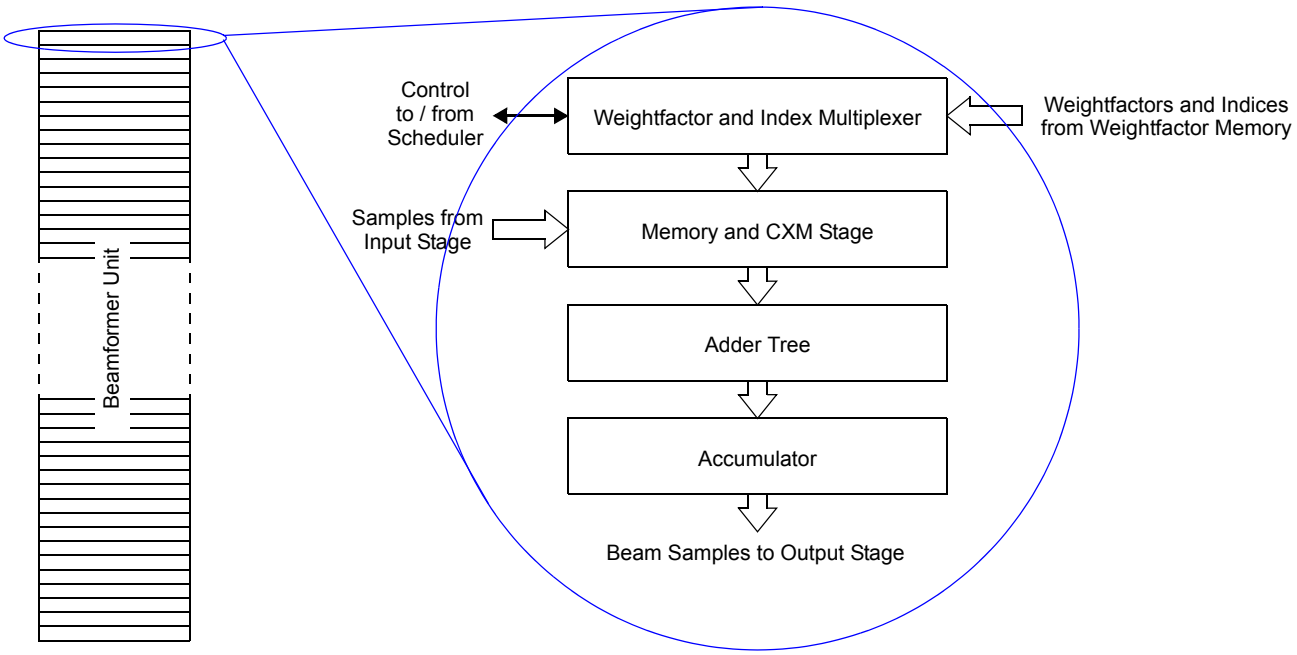

Figure 7: Beamformer Core Block Diagram

4.5.3 Weightfactor and Index Multiplexer

The scheduler has written an input vector into the sample memories, and also the frequency channel number (0 .. 7) into a small FIFO. The frequency channel number is used to select one of eight weightfactor streams from the weightfactor memory system. Thus, this stage mainly consists of a very wide MUX and a set of registers. A control unit reads the channel FIFO and counts the number of cycles. In cycle 2047 it issues the flag „LAST“ to the adder tree, to be passed on to the accumulator stage.

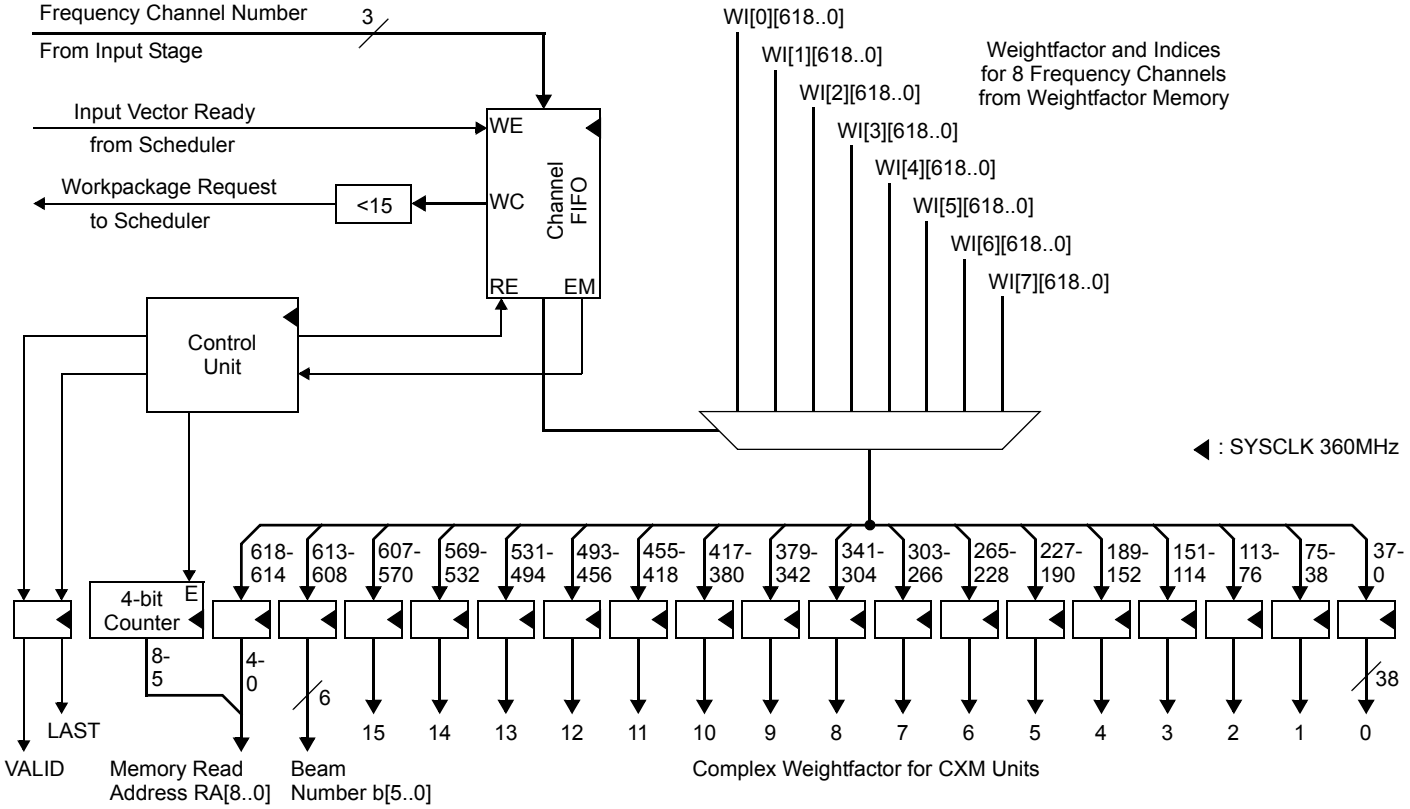

Figure 8: Weightfactor and Index Multiplexer

4.5.4 Memory and CXM Stage

This unit receives a set of 16 complex weightfactors from the multiplexer stage each clock. In addition to that, it receives a memory address of the corresponding set of samples, and the beam number. Its sole task is to read the sample memory, and to pass the 16 complex samples together with the weightfactors to the complex multiplier (CXM). Therefore this unit is divided into 16 Memory and CXM units, as shown in Figure 9.

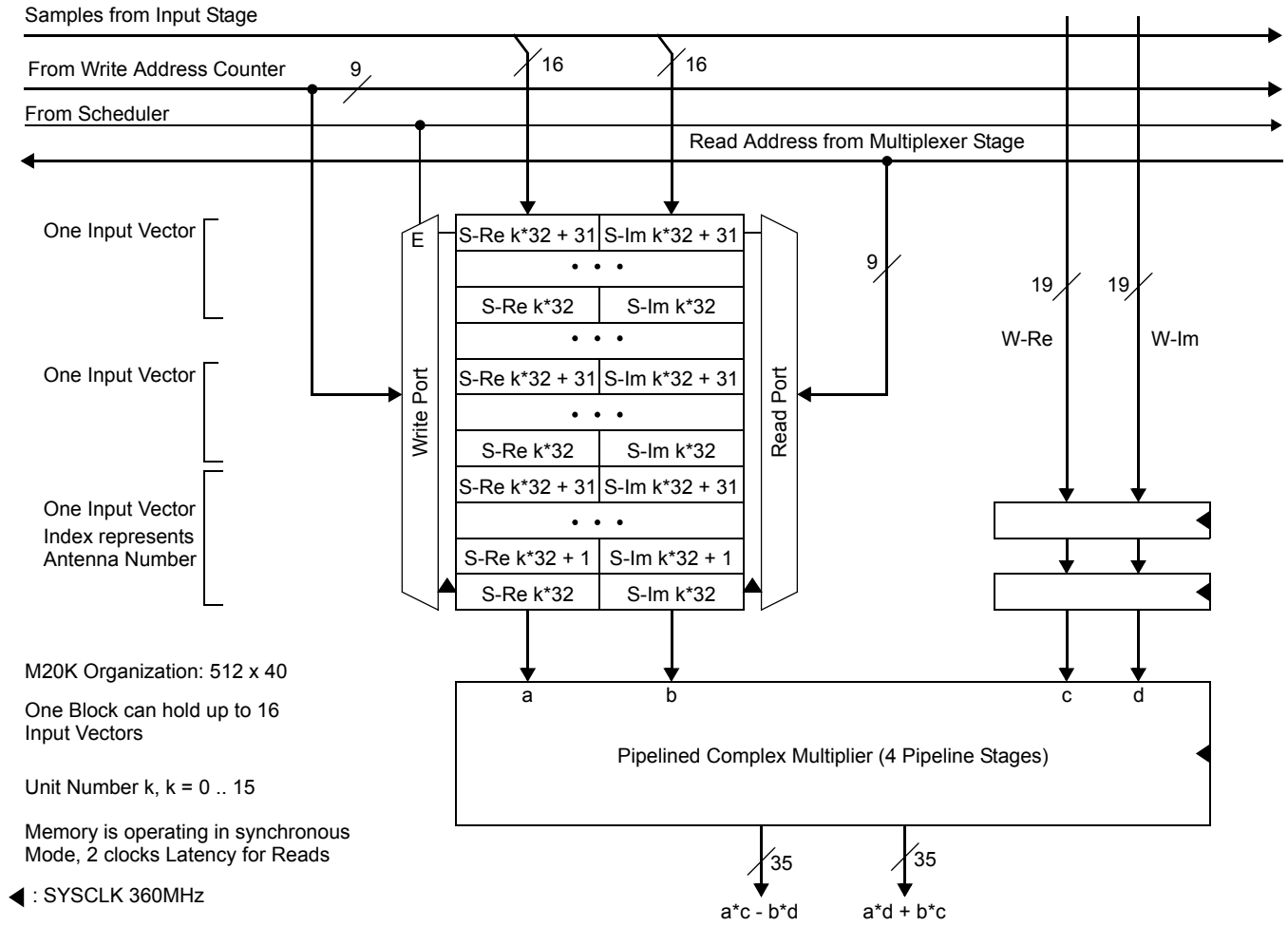

Figure 9: One Sample Memory and associated CXM Unit

The entire stage is shown in Figure 10. Products and beam number are passed on to the adder tree. Note that all 16 memory blocks, which consist of one M20K block each, receive the same read address. Writing samples from the input stage is always done in units of 16 complex samples, one from each input interface, per clock. Also, a transfer always includes one complete input vector. Therefore, we need only one write address counter for all 16 memories, which is incremented upon the Write Enable signal from the scheduler. The counter will endlessly wrap around.

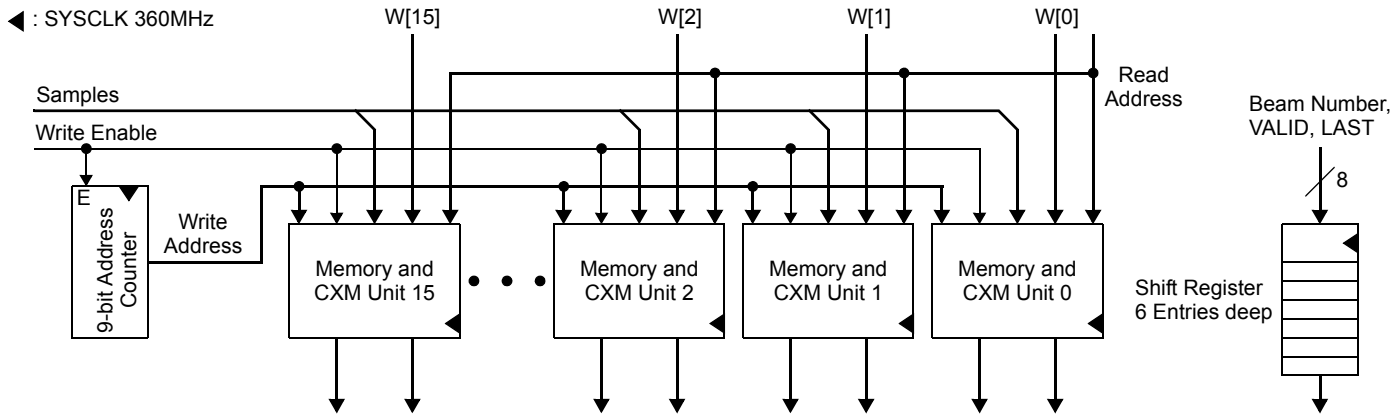

Figure 10: Memory and CXM Stage

4.5.5 *Adder Tree*

The adder tree is a pipelined binary tree of adders, which compute one partial sum of 16 complex products each clock. Each adder is pipelined in itself for high performance. Operand precision starts at 35 bits and increases by one bit per stage. The tree has 4 stages. After the last stage the operands are truncated to 35 bits again.

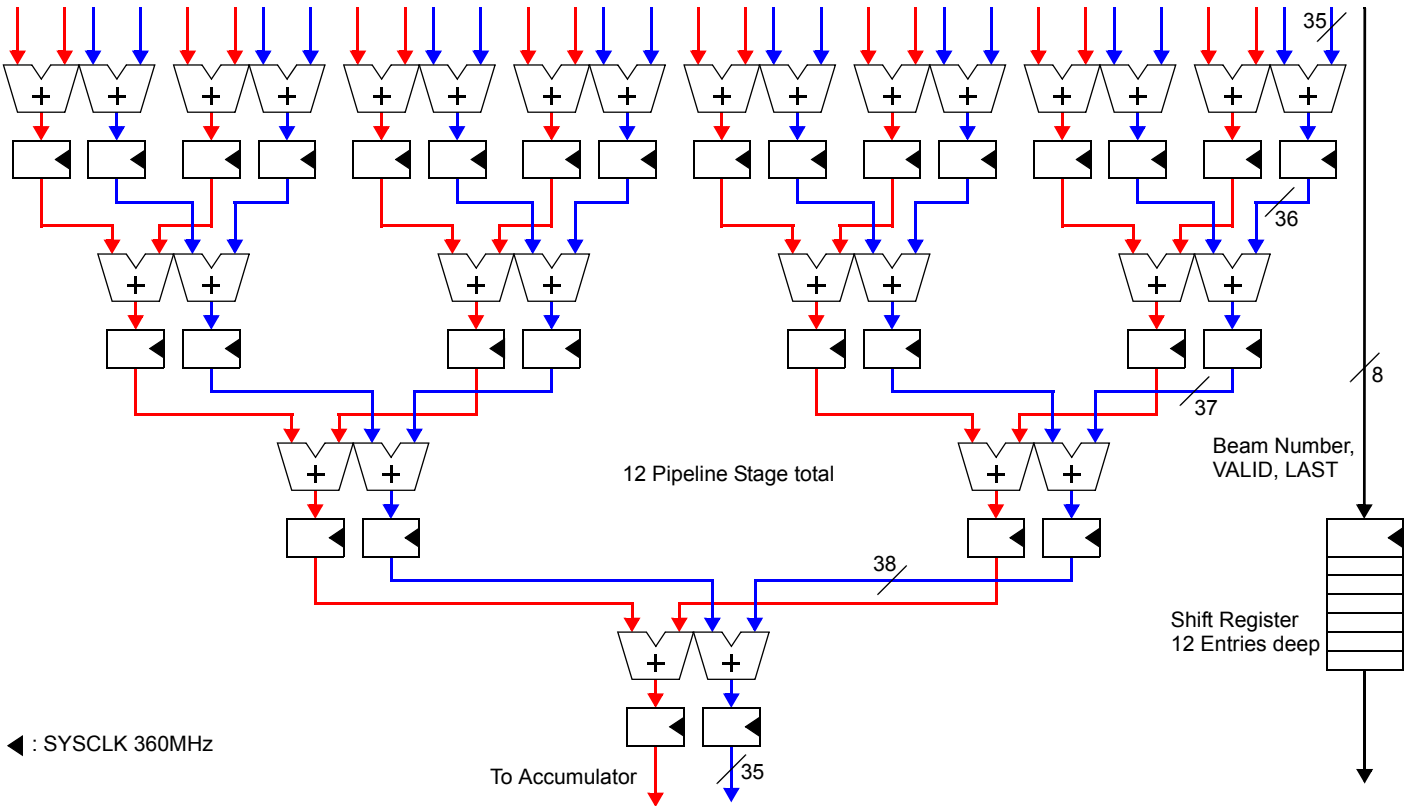

Figure 11: Adder Tree

A further modification of this unit could involve a fixed-point to floating-point conversion of the partial sum. This would also require the accumulator stage to be changed. Studies on potential accuracy and power trade-offs will be conducted at a later time during the project.

4.5.6 Accumulator Stage

The accumulator stage accumulates one complex sample for each of the 64 beams. Each component is maintained as a 40-bit number. Format conversions or rounding can be done after the beam samples are complete.
The partial sum of 16 complex multiplies that has been produced by the adder tree arrives together with the beam number at this stage. The beam number is used as the read address, and later as the write address, for a memory system holding the beam sample as it has been computed so far. In a pipelined fashion the new partial sum will be added to the current value, and the updated value will be written back. Since the „inner loop“ iterates over the beam number, read and write address will always be different and so read/write-hazards will be avoided.
For each beam, 32 partial products need to be accumulated.
The accumulator stage in its basic form is shown in Figure 12.

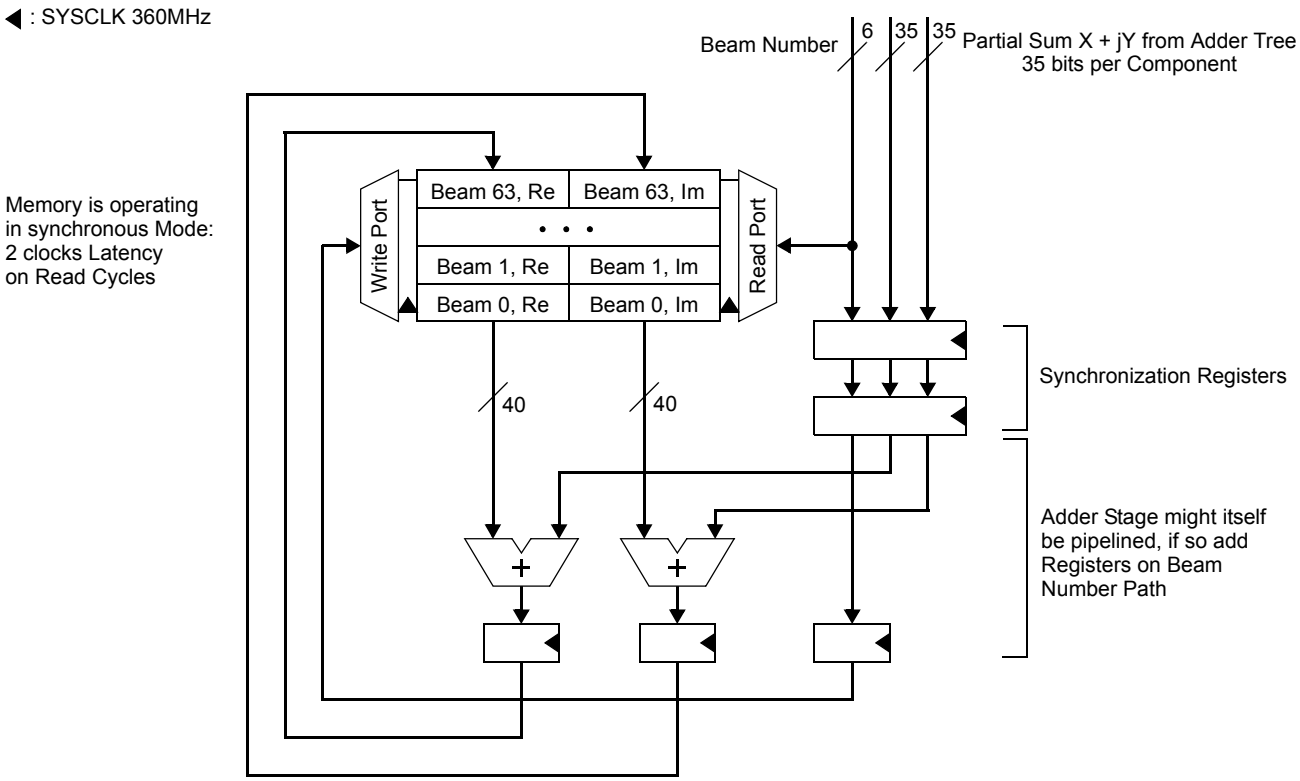

Figure 12: Accumulator Stage (Basic Form)

However, at some point in time the beam samples need to be read by the output stage, and the beam memory must be cleared for the next input vector. This might interrupt the operation of the beamformer core. In order to avoid this, the beam memory is implemented as a double-buffer. While one memory system is used for accumulation, the other is read and cleared. Buffer switching must be controlled by the beamformer core control unit, and synchronized with the output stage.

A block diagram of the double-buffered accumulator stage is shown in Figure 13. In this form the accumulator stage consumes four M20K blocks. For the entire beamformer unit this sums up to 188 M20K blocks, or about 7% of the available resources.

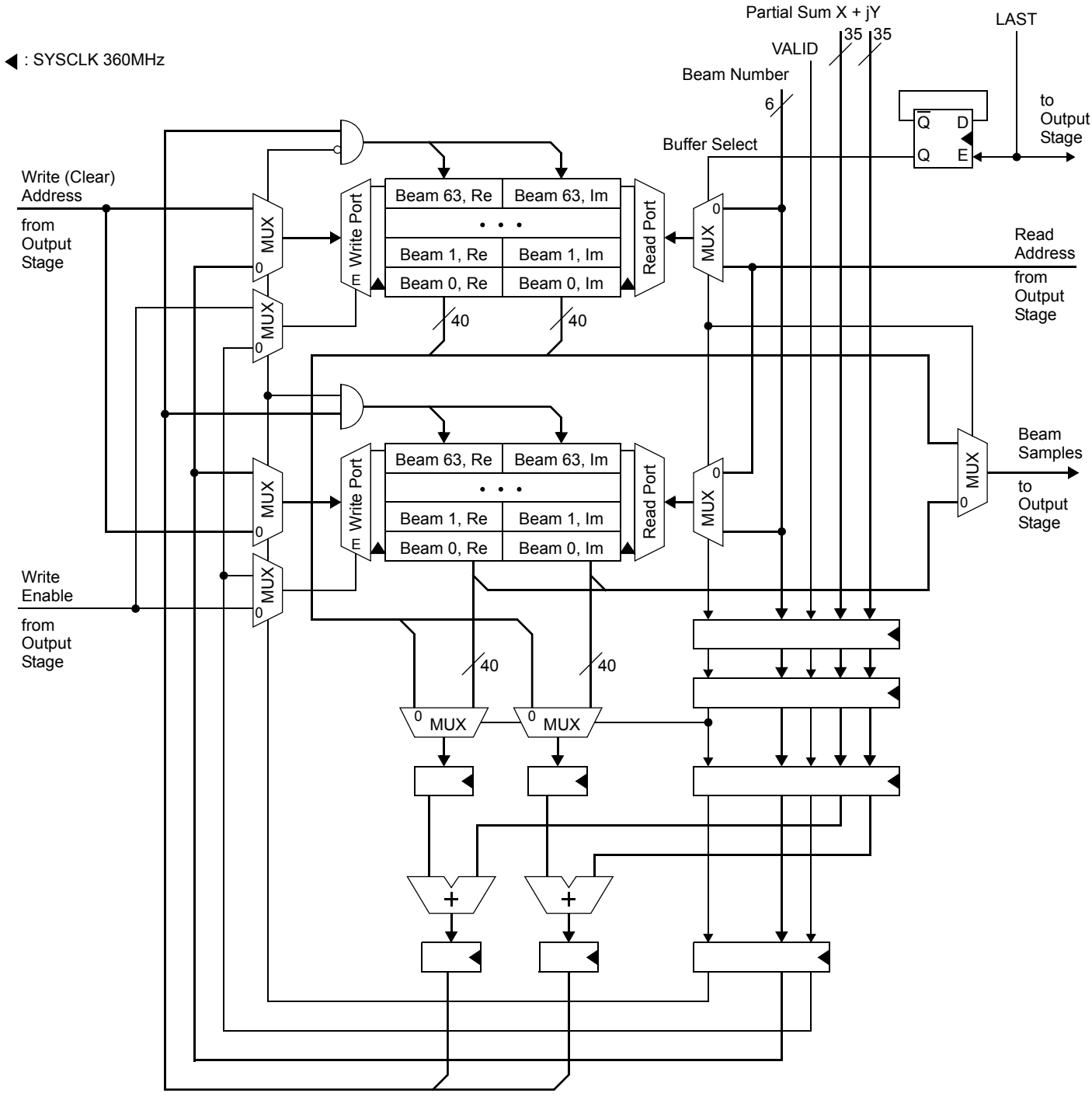

Figure 13: Double-Buffered Accumulator Stage

In Figure 13, note the flag „LAST“, which identifies the last partial sum of an input vector, which in turn completes the last beam sample. The next partial sum will belong to the next input vector, and must therefore be accumulated into the other buffer. This buffer in turn must have been read and cleared by the output stage by then. Computing all beam samples for a given input vector will take 2048 clock cycles, reading and clearing the buffer will take at most 128 clock cycles. The control unit of the output stage also receives the LAST-flag.

The LAST-flag switches all memory ports as the data travels through the pipeline, so there is not a single cycle overhead when switching to a new input vector.

4.6 Weightfactor Memory

Each beamformer-FPGA processes 8 frequency channels from 512 antennas and produces 64 beams. Therefore it needs to store 8*512*64 = 262,144 complex weights. Both real and imaginary components have a width of 19 bits and are two‘s-complement fixed-point numbers.
The weightfactor memory is organized as eight independent banks, one for each frequency channel. Thus, a bank holds 512*64 = 32,768 weightfactors and occupies 64 M20K blocks. The entire weightfactor memory therefore occupies 512 M20K blocks (out of 2,713, or about 19%).
In order to keep the beamformer core operating at maximum speed, each weightfactor memory bank outputs 16 weightfactors in parallel. Thus, each memory bank is physically organized as a 2048×640 bit memory, consisting of a 4×16 array of M20K blocks.
See Figure 14 for a block diagram of one weightfactor memory bank.

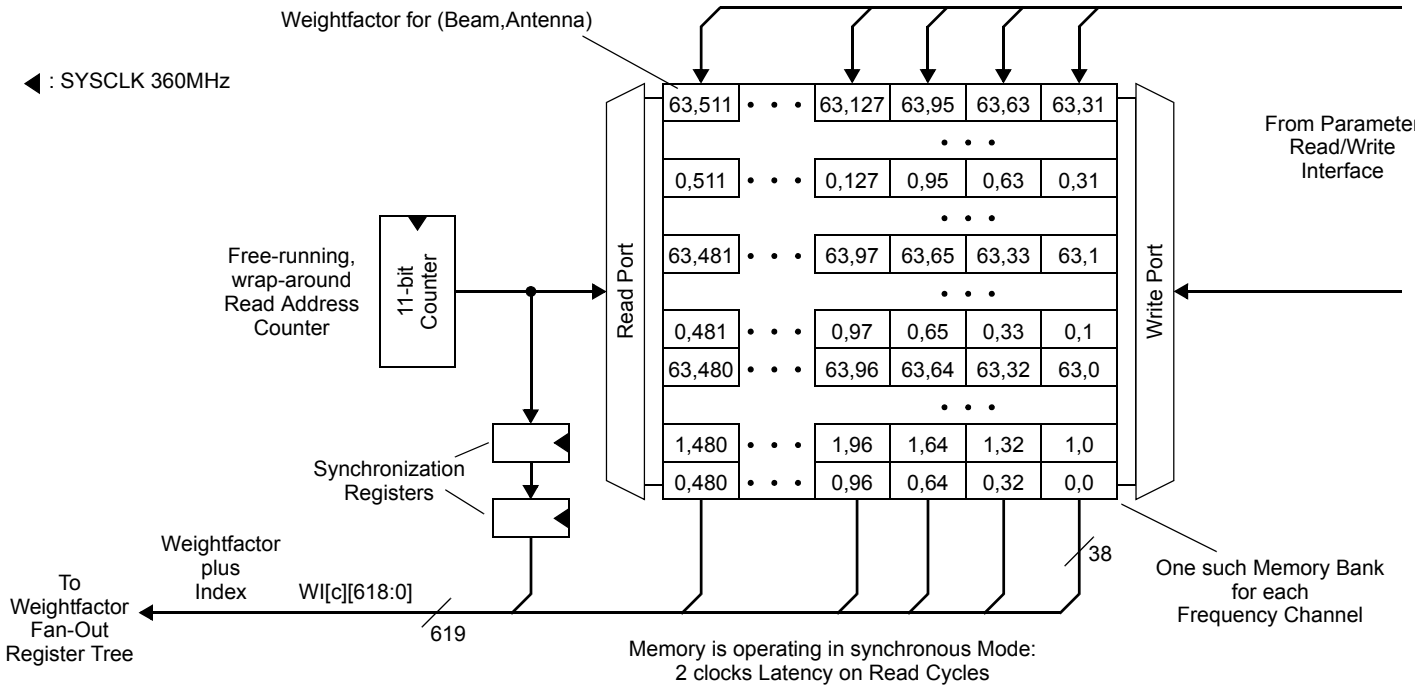

Figure 14: One Weightfactor Memory Bank

Note: a more flexible index management can be obtained by storing the indices along with the weightfactors in the weightfactor memory.

4.7 Weightfactor Fan-Out Register Tree

This distribution network is similar to and serves the same purpose as the register tree in Figure 6. It also uses a two-stage 6×8 register arrangement. Flipflop-consumption is 262,456, or 15.4% of the available registers. Note that again the latency introduced by this structure is of no concern.

4.8 Output Stage

The task of the output stage is to monitor the „LAST“-flags from all beamformer cores, collect the beam samples, convert the operands to single-precision floating-point numbers and send the data to the final, in this scenario external destination.

Reading the beam samples sequentially from all accumulator memories takes 47*64 = 3008 clocks, this is more than what is needed to compute a new set of beam samples on a given beamformer core (2048 clocks). Thus, the output stage needs to be implemented as several independent units, each serving only a subset of beamformer cores. For this example design we assume six such units, each serving 8 (7) beamformer cores. Clearing the beam sample buffers can be done mostly overlapping with the read-out.

The data rate produced by each individual output unit is as follows. One complex beam sample is computed in 32 clocks on each beamformer core. At 360MHz, this amounts to 88.9ns. Each beam sample is output as two SP-FP numbers, or 64 bits in total. The aggregate data stream from 8 beamformer cores adds up to 8 * 64 bits / 88.9ns = 5.76Gb/s. Thus, each output unit connects to one 10GbE interface. This can either connect to an optical transceiver, or to the backplane. As for the input stage there is one output FIFO, translating between the clock domains. Input and output data width is the same in this case.

The schematic diagram of one output unit is shown in Figure 15. The control unit receives the LAST-flags from the eight beamformer cores, prioritizes the signals and reads out the beam samples. A tree of multiplexers is used to pass the proper samples to the fixed-point to floating-point converters. The control unit will also have all required information to assemble a valid Ethernet packet including destination address.

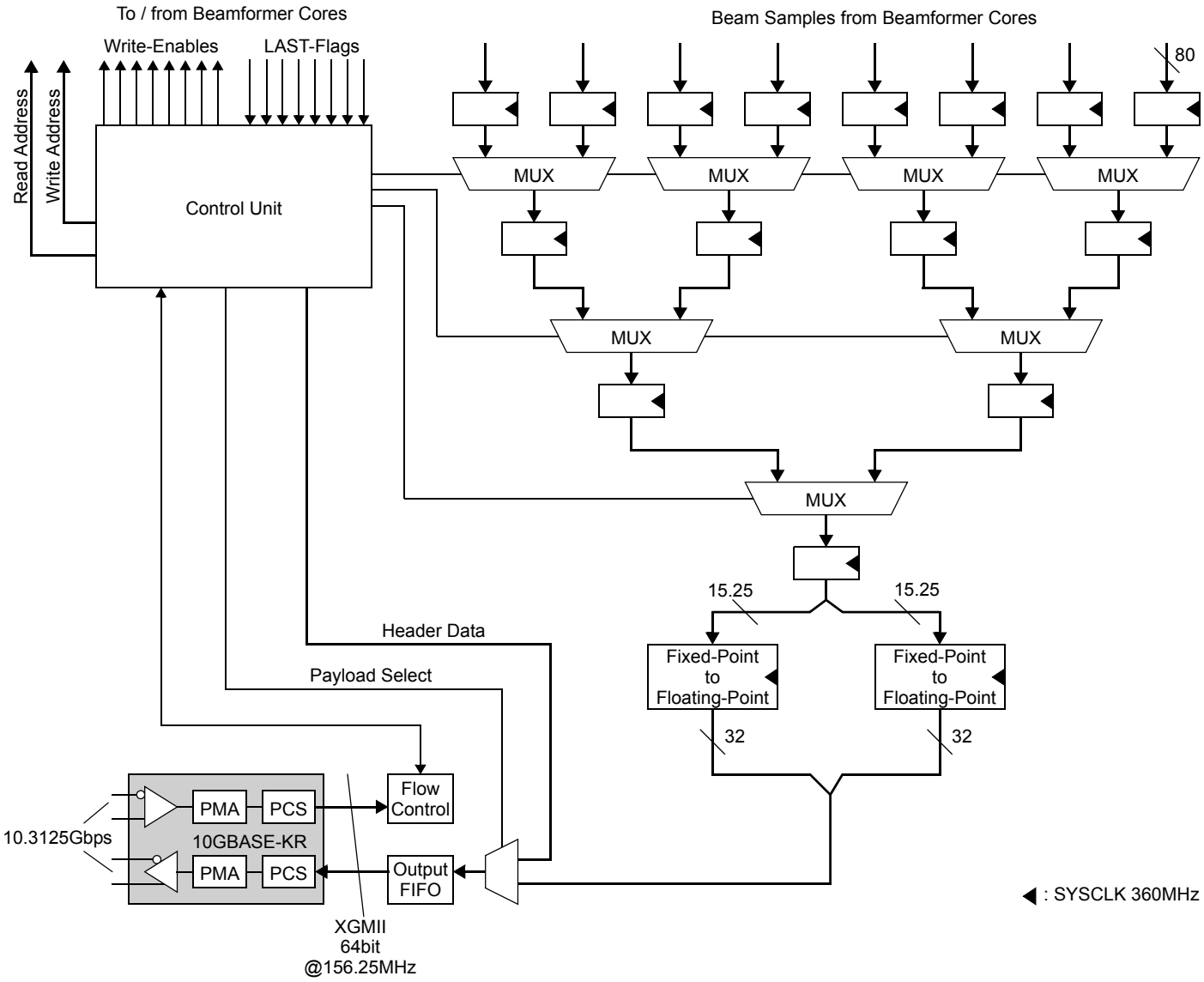

Figure 15: Output Unit

4.9 System Outline

Figure 16 shows a sketch of the Uniboard and a 3D rendering of how the complete beamformer system could look like. The dimensions of the Uniboard are 280mm x 366.7mm. The midplane is designed for a 19” rack and is of size 428mm x 387mm. Thickness is 3.3mm.

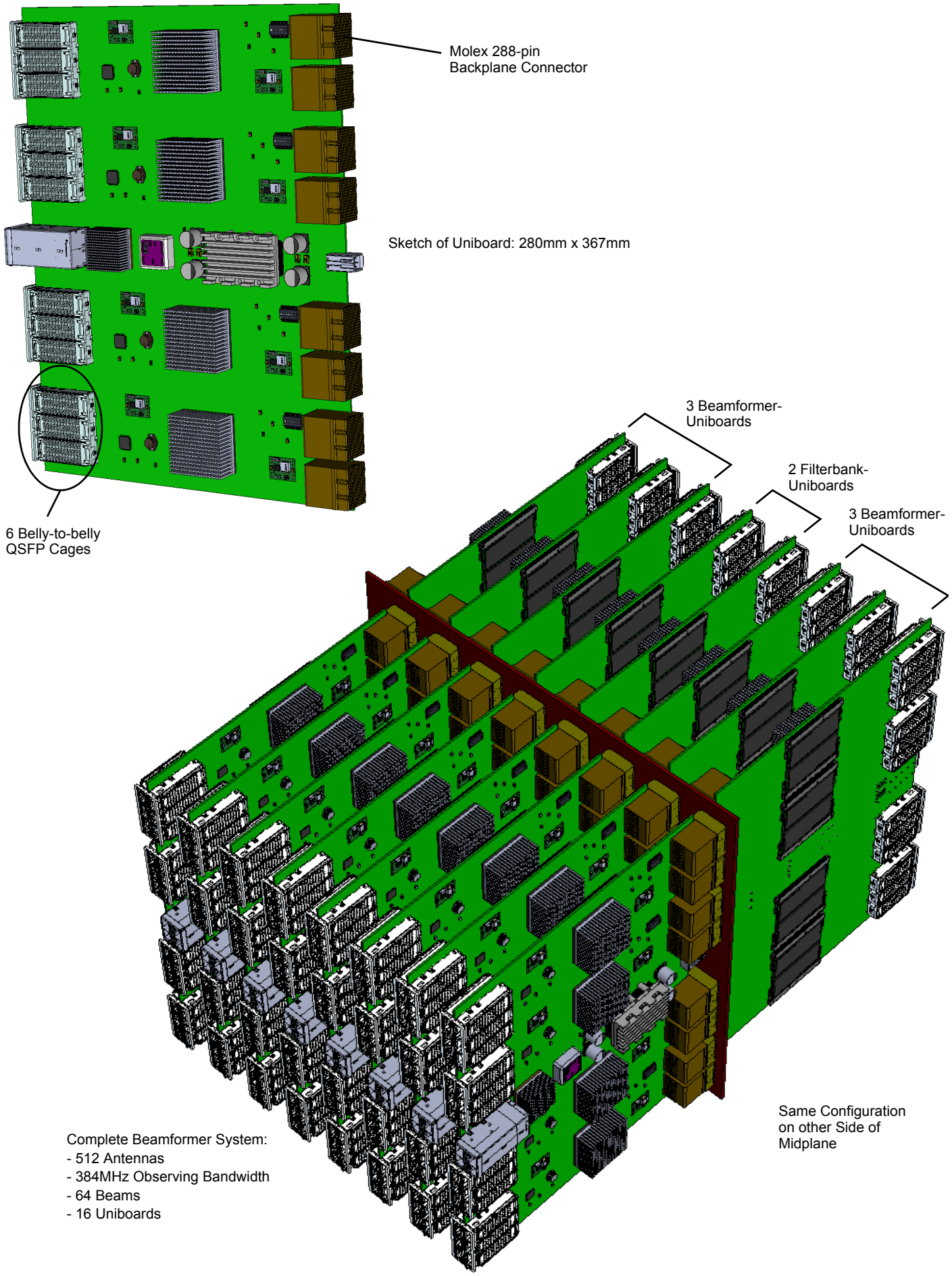

Figure 16: Sketches of the Components

4.10 Design Summary

We have presented system and chip architecture for a beamformer based on the Uniboard^2 signal processing platform. The performance figures are oriented towards the SKA Phase 1 requirements: 384MHz observing bandwidth, 256 dual-polarization receivers (512 antenna signals), and 64 beams. The design target was to achieve a high resource utilization and a high compute efficiency. This has been achieved by a special data flow architecture, in which the stream of weightfactors, rather than the stream of samples, controls the sequence of operations. This allows a high percentage of chip resources such as hardware-multipliers, even if being an uneven number, to be employed. Also, any expensive redundant storage of weightfactors is avoided. An overall pipeline structure together with redundant buffers allows uninterrupted operation without a single-cycle loss.

The system is of moderate size (16 Uniboards and a backplane) and uses FPGAs in the slowest speed grade. This should help to keep the hardware and power supply costs low. Further studies in this direction („green measures“) will be done during the further course of the project.

5 SCALABILITY

The multidimensional design space that needs to be considered for the specific requirements of radio astronomical facilities is spanned by the following parameters:

- desired observing bandwidth,
- number of antennas,
- number of beams,
- required numerical precision,
- data rate of the surrounding infrastructure,
- recurring costs (power consumption, cooling, building costs, maintenance etc.)
- non-recurring costs (design effort, component costs, manufacturing, deployment etc.)

In the remainder of this section we will explore possibilities for increasing the performance of the system with respect to observing bandwidth, number of antennas and number of beams, necessarily trading in arithmetical precision and costs. We'll start on the micro-architectural level (FPGA circuitry) and work our way up to the system level.

5.1 Alternative Arithmetic

Recalling (1) we can see that the dominant operation is a complex multiply, which when performed in Cartesian coordinates requires four multiplies and two adds. Therefore, the number of multiplier hardmacros on the FPGA defines the upper limit in performance. The obvious idea is to perform this operation in polar coordinates, which then takes the form

$$r_1 \cdot r_2 \qquad \angle(\varphi_1 + \varphi_2) \tag{5}$$

The expenses are then reduced to one multiply and one add.

However, the complex multiply is immediately followed by an add, which cannot be done (so easily) in polar coordinates. Thus, the result (scaled and phase-shifted phasor) must be transformed into Cartesian coordinates immediately after the complex multiply. It is obvious that neither any further multiply nor any trigonometric functions can be employed for this operation.

The method of choice is therefore the well-known CORDIC algorithm [8]. CORDIC can be used to transform a complex number from Cartesian to polar coordinates and back. It is a multiplier-free, iterative shift-and-add operation that can be implemented as a pipeline for high throughput. In principle, the architectural changes to the system would be as shown in Figure 17. Note that all other aspects of the beamformer as presented so far, such as data paths, memory systems and flow control, remain unaffected.

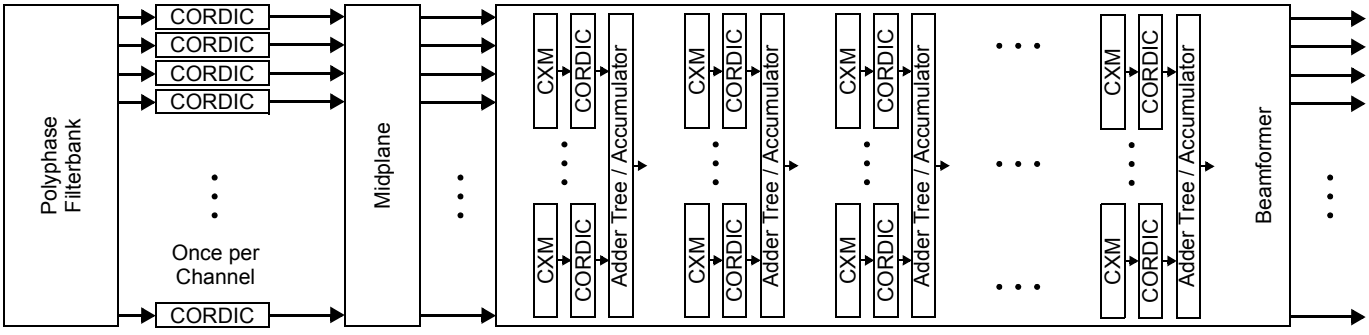

Figure 17: Beamformer System using Polar Coordinates

So, the general idea is to implement all CORDIC units using the regular FPGA fabric and to pair each multiplier hardmacro with one CORDIC unit. In this way, the number of CXM-macros (see section 4.5.4) would be increased by a factor of four compared to the previous approach. Likewise, the overall chip performance would be increased by four, which could be used, for example, to compute four times the number of beams.

However, the CORDIC unit requires a certain amount of logic resources, depending on the desired precision. Available chip resources may put an upper limit on this approach. This is detailed in the following section.

5.1.1 The CORDIC Unit

Figure 18 shows the pipeline for transforming a complex number from polar coordinates to Cartesian coordinates (real and imaginary part). The necessary scaling operation at the end is not accounted for. Instead, it can be pre-multiplied into the weightfactor magnitude (5) or be deferred to later stages in the processing pipeline. Also, it is assumed that the phase β of the operand has been normalized to a range of $-\pi/2 \le \beta \le \pi/2$. All operands are processed as two's complement fixed-point numbers.

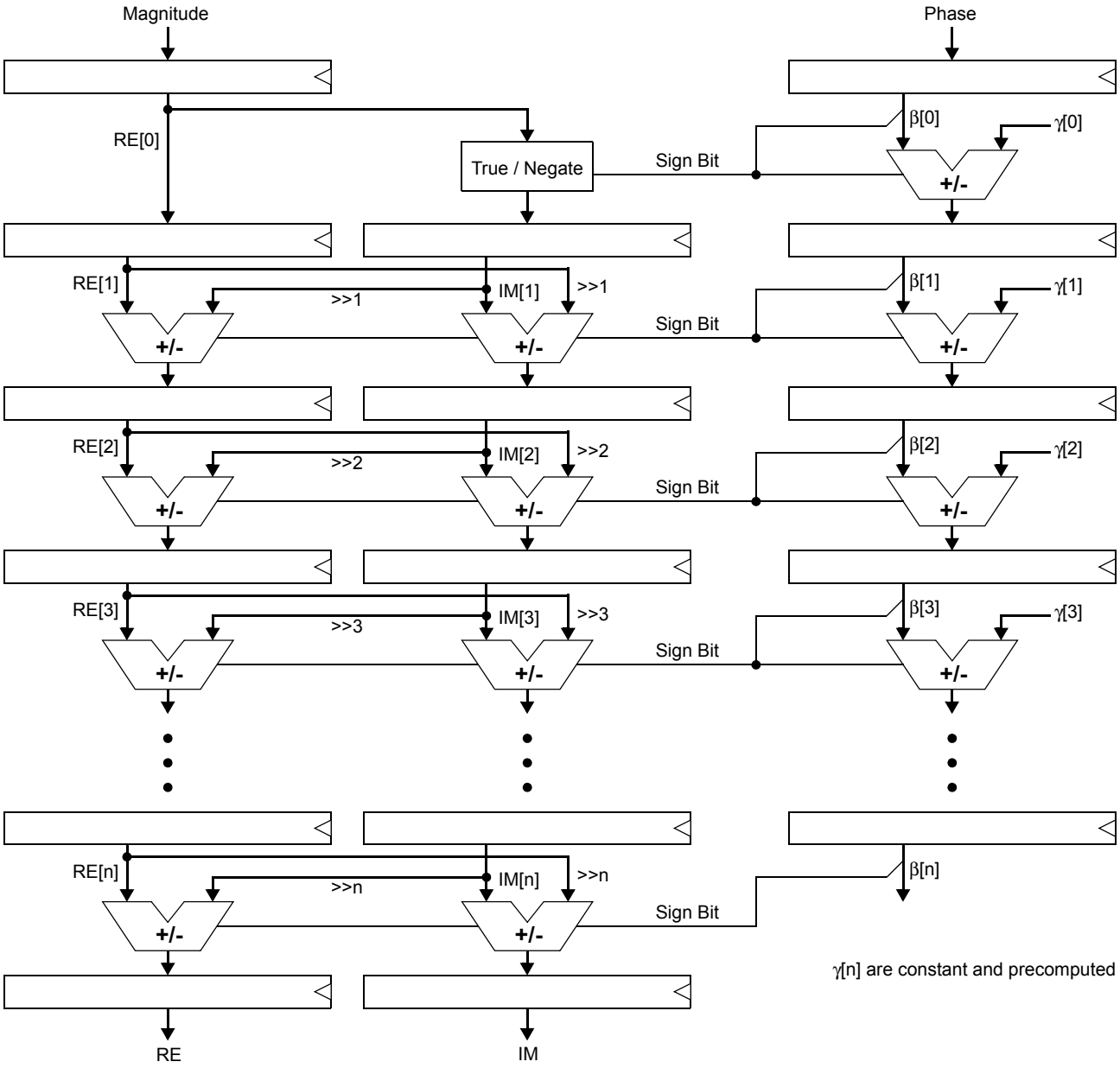

Figure 18: CORDIC Unit

As can be seen in Figure 18, both precision and expenses depend on the following quantities:

- the bitwidth *K* of the phase angle,
- the bitwidth *L* of real and imaginary components, and
- the number *N* of pipeline stages (iterations).

To a certain degree these quantities are correlated with each other. A high number of stages requires a certain bitwidth of the real and imaginary components or otherwise all bits will be shifted out. Likewise a certain angular resolution must be provided or otherwise the comparisons will be meaningless.

In [8] we find: "*For most ordinary purposes, 40 iterations (n = 40) is sufficient to obtain the correct result to the 10th decimal place.*" This would roughly correspond to IEEE754 SP-FP accuracy. Other than this it is hard to give an analytic error function over the above quantities. Therefore a bit-accurate simulator was written in C to give visual representations of the error distribution. This simulator compares the outputs of the CORDIC pipeline to DP-FP results of the C library.

Figure 19 shows the relative error in percent of four selected configurations. The parameters K, L and N are shown next to the plots. In general, the error is pronounced towards small magnitudes. With decreasing parameter values the error becomes larger, but more importantly, it is pronounced for angles close to 0° or 90° irrespective of magnitude.

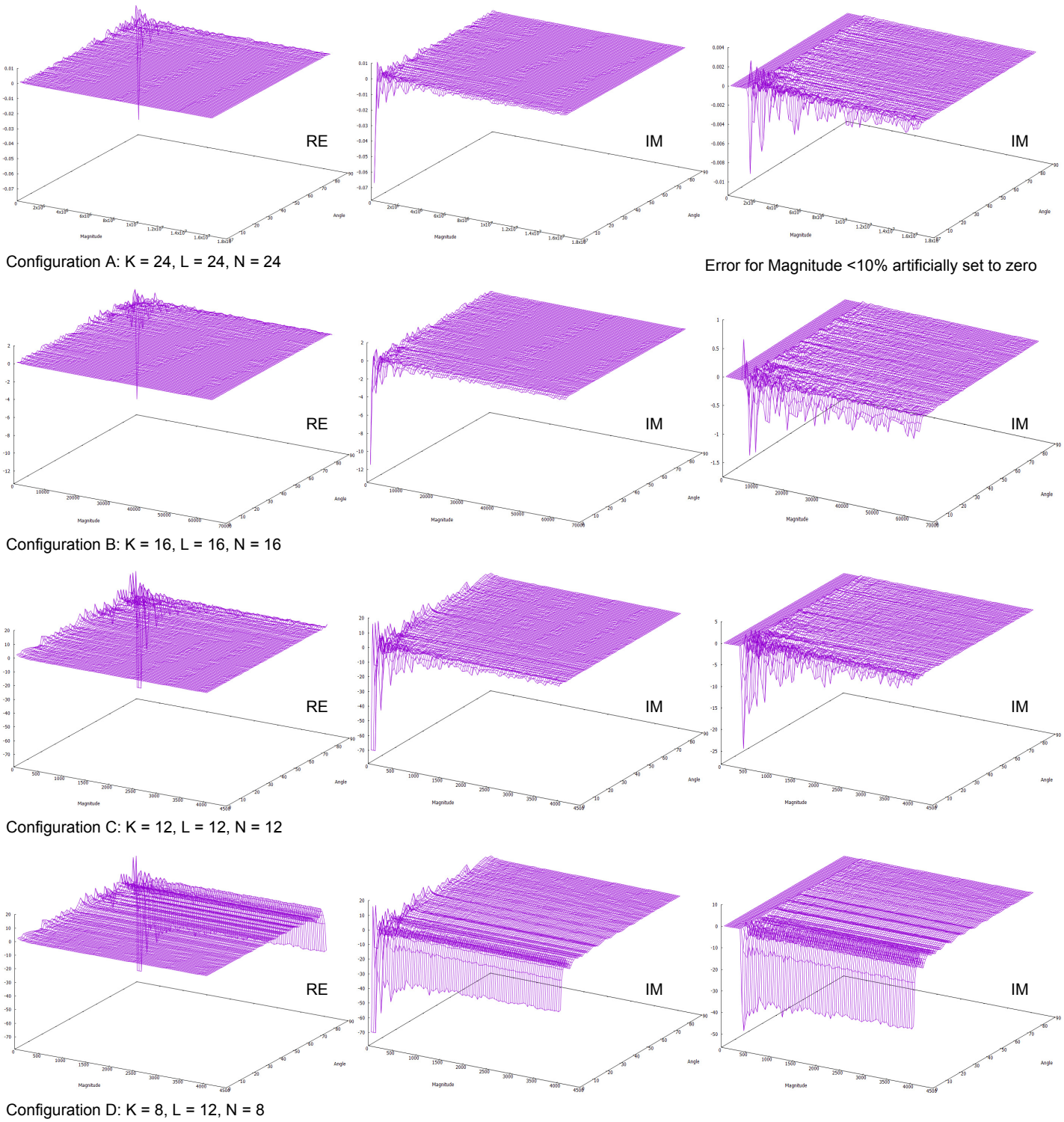

Figure 19: CORDIC Relative Error Distribution

Resource consumption is most critical for the adders in the CORDIC pipeline. In the FPGA-fabric, adders are implemented using LUTs and the fast carry chains. Altera FPGAs provide a certain number of *ALMs* (Adaptive Logic Module). Each ALM can be configured to provide two full-adders. Thus, a 16-bit adder consumes 8 ALMs.

However, as we recall from Figure 9, each CXM is also paired with one M20K memory block. Thus, either ALMs or M20K blocks may become the limiting factor for this approach. Table I lists the available resources for both the low-cost Arria-10 GX 1150 and the high-end Stratix-10 GX 2800.

TABLE I: AVAILABLE FPGA RESOURCES

FPGA	Adaptive Logic Modules ALM	18x19 Multiplier Hardmacros	M20K Memory Blocks
Arria-10 GX 1150	427,700	3,036	2,713
Stratix-10 GX 2800	933,120	11,520	11,721

Table II and Table III list the FPGA resources required for different configurations. Fundamental quantity is the number of implemented beamformer cores, which each contain 16 CXMs and 20 M20K memory blocks (see section 4.5.2). In our traditional approach an Arria-10 GX 1150 allows 47 beamformer cores to be implemented. So we start the table with four times this number, or 188 beamformer cores.

TABLE II: FPGA RESOURCE REQUIREMENTS (ARRIA-10 GX 1150)

Beamformer Cores	18x19 Mult. Hardmacros	CORDIC Config.	ALMs	% of Chip	M20K Mem. Blocks	% of Chip
188	3,008	A	2,526,720	591	3,760	139
188	3,008	B	1,106,944	259	3,760	139
188	3,008	C	613,632	143	3,760	139
188	3,008	D	354,944	83	3,760	139
94	1,504	C	306,816	72	1,880	69
47	752	B	276,736	65	940	35

The bottom row in Table II represents the same performance as the traditional approach, however, with a potentially reduced accuracy. It might still be of interest if power consumption of the FPGA can be lowered using this kind of circuitry. This can be subject of further studies for deliverable D8.16, („green measures“).
The configuration using 94 beamformer cores is of higher interest since it would double the performance (see sections 5.3ff), again at a somewhat reduced precision. All other configurations cannot be implemented. So, the optimistic goal of pairing each multiplier hardmacro with a CORDIC unit cannot be reached without architectural changes.

For the Stratix-10 GX 2800 FPGA we start the table with the theoretical maximum number of beamformer cores, that is 11,520 / 16 = 720. Since the number of ALMs only scales by a factor of 2.2 for the Stratix device, this method appears to be of little use on this plattform.

TABLE III: FPGA RESOURCE REQUIREMENTS (STRATIX-10 GX 2800)

Beamformer Cores	18x19 Mult. Hardmacros	CORDIC Config.	ALMs	% of Chip	M20K Mem. Blocks	% of Chip
720	11,520	A	9,676,800	1037	14,400	123
720	11,520	B	4,239,360	454	14,400	123
720	11,520	C	2,350,080	252	14,400	123
720	11,520	D	1,359,360	146	14,400	123
360	5,760	C	1,175,040	126	7,200	61
180	2,880	C	587,520	63	3,600	31

5.1.2 Conclusion

Performing the complex multiply in polar coordinates suffers from a high resource consumption and low precision. Still there might be configurations and scenarios where this method can be used advantageously. This depends on application, chip architecture and downstream processing requirements and is therefore simply an engineering decision. The problems largely stem from the inefficient adder-logic on FPGAs. Therefore, especially for low-power ASIC-designs it might be worth to re-evaluate the method.

5.2 Alternative FPGAs

As advertised by Altera, it was planned from the start of the Uniboard^2 project to initially use the low-cost Arria-10 device and later upgrade to a more powerful, pin-compatible Stratix-10 device. As of now, however, there is no such device on the market [9]. Accordingly, there is very little information about performance (maximum core or DSP clock frequency) available. We therefore assume that the device in question will be a Stratix-10 GX 2800 [10] and conservatively estimate that the core clock can be as high as 500MHz. As listed in Table I, there are 11,520 multiplier hardmacros and 11,721 M20K memory blocks on the chip. We further assume that all multipliers are used to construct beamformer cores and that other arithmetic units such as format converters are made from fabric logic. Then we will have 180 beamformer cores.

5.3 Increasing the Number of Beams

5.3.1 Circuit-Level

In the following we assume that the number of beamformer cores is doubled to 94 (Arria-10 using CORDIC, trading in accuracy) or approximately quadrupled to 180 (Stratix-10, at much higher chip costs).
In this section we assume further that the increase in processing power shall solely be used to double or quadruple the number of computed beams, while keeping the observing bandwidth (384MHz), number of channnels (384) and number of antennas (512) constant. The obvious consequences are:

- the input data rate (per clock) remains the same,
- both the Input Fan-Out Register Tree (see Figure 6) and the Weightfactor Fan-Out Register Tree (see section 4.7) need to service twice or four times the number of beamformer cores and might need to be adapted,
- the size of the Weightfactor Memory (see Figure 14) increases in size,
- the output data rate per clock increases by a factor of two or four.

While for the traditional Stratix-10 architecture all units scale linearly, we can reduce precision at several places in the Arria-10 CORDIC system to the precision that the CORDIC units provide, in this case 12 bits. Each of the eight weightfactor memory banks therefore has a width of 2*12*16 = 384 bits and a depth of 4096 words. It can be constructed from 80 M20K memory blocks, for a total of 640 M20K memory blocks.
Since the number of antennas has not changed each beamformer core still produces one beam sample every 32 clocks. A beam sample consists of two SP-FP numbers of 32 bits each. Combined output data rate of 8 beamformer cores is therefore 5.76Gb/s (Arria-10) or 8Gb/s (Stratix-10). Again we use one Output Unit (see Figure 15) for 7 or 8 beamformer cores.
Hardware requirements are summarized in Table IV.

TABLE IV: INCREASING THE NUMBER OF BEAMS

Architecture	Observing Bandwidth	# of Antennas	# of Beams	# of 18x19 Multipliers	# of M20K	# of Output Units	Output Data Rate (Gb/s)
Arria-10 CORDIC	384MHz	512	128	1,504	2,656	12	67.68
Stratix-10	384MHz	512	256	11,520	5,808	24	180.0

Looking at Table IV it appears to be possible to compute twice or four times the number of beams without changes to the architecture if we accept reduced accuracy or higher chip costs.

5.3.2 System-Level

By simply replicating the entire system of 16 Uniboards (and the midplane), the number of beams can also be multiplied provided each system receives the proper set of weightfactors [11]. The Uniboards performing the polyphase filterbanks are redundant, though.

5.4 Increasing the Observing Bandwidth

5.4.1 Circuit-Level

For this study we assume that the observing bandwidth shall be doubled (768MHz, Arria-10 CORDIC) or quadrupled (1536MHz, Stratix-10). Channel bandwidth shall remain at 1MHz. That is, the increase in beamformer cores shall be used to compute the same number of beams from the same number of antennas, but on twice or four times the number of channels.
Since the filterbank design is not part of this workpackage we simply assume that it can be implemented on the respective plattform. However, we have to examine the bitrates at the input and outputs, i. e., at the optical interfaces and on the midplane.
If we assume 8 bits per sample from the ADC, the input data rate for each filterbank-FPGA is 1,536e6*8*32 = 393Gb/s (Arria-10) or 786Gb/s (Stratix-10). If we restrict ourselves to using current 40GbE technology, these kinds of bandwidth are not available at the optical connectors of the filterbank-FPGAs alone. Thus, antenna connections must be distributed across the entire system, and routed via the midplane to the Uniboards that implement the filterbanks. This is shown for the Stratix-10 case. We assume, however, a collection of external switches that aggregate the data streams from the ADCs. An example configuration is shown in Figure 20.

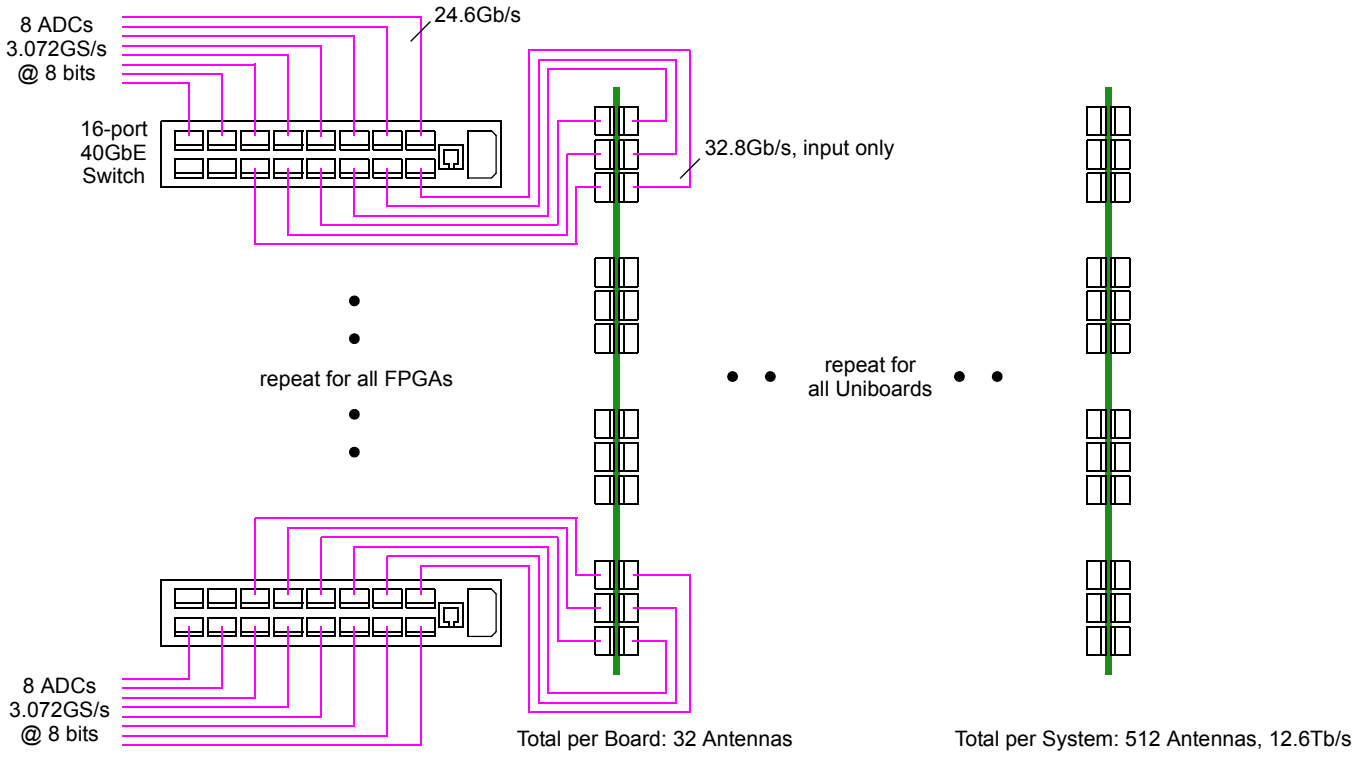

Figure 20: Aggregation Switches and Uniboards

As can be seen, each filterbank-FPGA receives 8 antenna signals directly via its own optical interfaces. The remaining 24 signals must be routed through the midplane from the beamformer-FPGAs. As we recall from Figure 3, there is exactly one FD lane between any given pair of beamformer-FPGA and filterbank-FPGA. One direction is used to distribute the ADC samples, as shown in Figure 21.

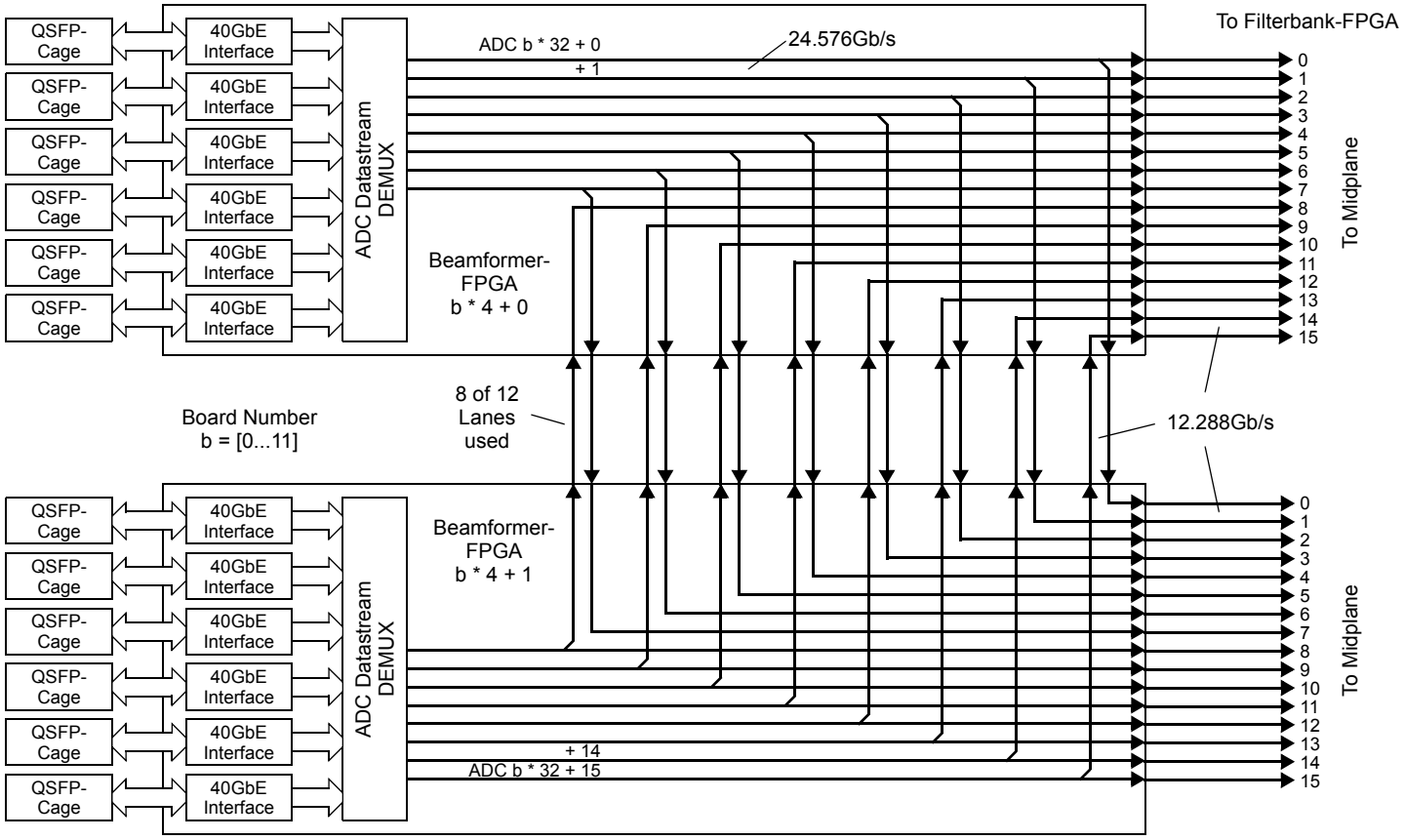

Figure 21: ADC Sample Distribution (½ Uniboard shown, other Half identical for remaining Antenna Signals)

As can be seen, the 512 antenna signals can nicely be routed to the filterbank-FPGAs provided that the link speed across the midplane can be set to a net data rate of 12.288Gb/s.

In reverse direction, depending on precision, the required data rate can exceed physical capabilities of the transceivers and wires. Each filterbank-FPGA needs to send 16 or 32 channels from 32 antennas to a given beamformer-FPGA. Required data rates are shown in Table V.

TABLE V: REQUIRED DATA RATE FROM FILTERBANK-FPGA TO BEAMFORMER-FPGA

Architecture	# of Channels	Data Rate @ 16;16	Data Rate @ 12;12	Data Rate @ 8;8
Arria-10 CORDIC	16	16.384Gb/s	12.288Gb/s	8.192Gb/s
Stratix-10	32	32.768Gb/s	24.576Gb/s	16.384Gb/s

Thus, one can be optimistic about using 12-bit components for Arria-10 and 8-bit components for Stratix-10. Better accuracy for Stratix-10 will not be feasible for a midplane of this size.

The Input Stage on each beamformer-FPGA architecturally remains the same, however, the rate at which complex samples arrive is increased to 512MS/s or 1GS/s. In the current architecture, consecutive samples in the input FIFOs (see Figure 5) go to the same memory block in the Memory and CXM Stage (see Figure 9). Thus, one would need to increase the transfer rate (write cycles per second) beyond the set limits.

A solution to this problem could be to rearrange the chip resources. In particular, the beamformer core would need to be extended to include 32 instead of 16 Memory and CXM Units. Obviously, the number of beamformer cores on each FPGA would then be halved. Data distribution for a given input vector would then be as shown in Figure 22.

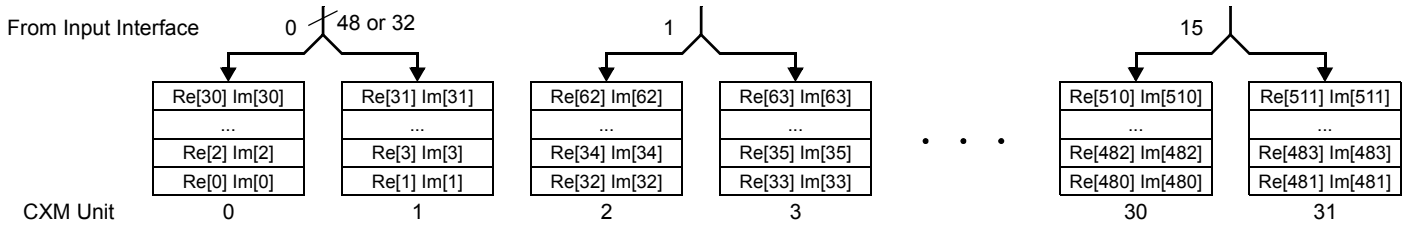

Figure 22: Data Distribution in Modified Beamformer Core

In this way two complex samples are written per clock, bringing bus and memory clock rates back into the legal range. However, there are a number of system implications that need to be considered:

- Depending on sample precision, the width of the input FIFOs increases. Also, register consumption of the Input Fan-Out Register Tree increases to 41,472 flipflops (2.4% on Arria-10).
- The memory in a beamformer core should be consolidated. For 12-bit components the total width of the memory is 768 bits. Thus the memory can be constructed using 20 M20K blocks. For the Arria-10 version, the beamformer unit would then consume a total of 1,128 M20K blocks including accumulators. For the Stratix-10 version, M20K blocks are not a critical resource.
- Width, depth and capacity of the Weightfactor Memory change significantly. The Weightfactor Memory has to provide 32 weights on 16 or 32 channels in parallel. This amounts to a data bus of 12,288 bits (Arria-10) or 16,384 bits (Stratix-10). The depth is 1,024 such words. The Weightfactor Memory can be built using 616 (Arria-10) or 820 (Stratix-10) M20K blocks.
- The Weightfactor Fan-Out Register Tree consumes 663,552 flipflops on the Arria-10 FPGA, or 39% of available resources. The CORDIC-pipelines consume approximately 650,000 flipflops. At such a high usage percentage, routing might become a problem. For Stratix-10 this is not an issue because of the HyperFlex-feature (registers embedded in the routing channels).
- Each beamformer core now produces one beam sample every 16 clocks. This amounts to a bit rate of 1.44Gb/s (Arria-10) or 2Gb/s (Stratix-10) per beamformer core. Thus we use one Output Unit (see Figure 15) per four beamformer cores, for a total of 12 (Arria-10) or 24 (Stratix-10) interfaces.

Table VI shows a summary of performance and resource consumption of this approach.

TABLE VI: INCREASING THE OBSERVING BANDWIDTH

Architecture	Observing Bandwidth	# of Antennas	# of Beams	# of Beamformer Cores	# of 18x19 Multipliers	# of M20K	# of Output Units	Output Data Rate (Gb/s)
Arria-10 CORDIC	768MHz	512	64	47	1,504	2,052	12	67.68
Stratix-10	1,536MHz	512	64	90	11,520	4,060	24	180.0

5.4.2 System-Level

Provided that

- suitable external switches for the purpose as shown in Figure 20 are available,
- the digitized antenna signals are fed into the system as shown in Figure 21, and
- the necessary filterbanks can be implemented on the FPGAs

a collection of the original beamformer systems can be operated in parallel for processing larger observing bandwidths. Each system would then process / discard a disjoint set of frequency channels. Again all but one of the Uniboards that are used for filtering are redundant.

In case implementing these large-bandwidth filterbanks is a problem a low-cost compromise could be to use Stratix-10 for the filterbank-FPGAs and Arria-10 otherwise.

5.5 Increasing the Number of Antennas

5.5.1 Circuit-Level

In this section we examine if and how the number of antennas can be increased to 1,024 and 2,048, respectively. That is, the increased processing power shall be used to process more antennas, but over the same bandwidth and for the same number of beams.

The obvious consequence is that the external aggregation switches need more ports, most of them with lower bandwidth. Per ADC we have 768e6 * 8 = 6.144Gb/s. For 2,048 antennas, the switches in Figure 20 should be replaced by switches that provide 32 10GbE ports and 6 40GbE ports, or equivalent. Data distribution is then identical to Figure 21 except that the data streams across the midplane carry signals from multiple antennas.

On the filterbank-FPGAs twice or four times the number of polyphase filterbanks need to be implemented. Again we simply assume that this can be done. For the data rate between any given pair of filterbank-FPGA and beamformer-FPGA the same considerations apply as listed in Table V, in this case, however, depending on the number of antennas.

TABLE VII: REQUIRED DATA RATE FROM FILTERBANK-FPGA TO BEAMFORMER-FPGA

Architecture	# of Antennas	Data Rate @ 16;16	Data Rate @ 12;12	Data Rate @ 8;8
Arria-10 CORDIC	1,024	16.384Gb/s	12.288Gb/s	8.192Gb/s
Stratix-10	2,048	32.768Gb/s	24.576Gb/s	16.384Gb/s

For the Input Stage the sample rate is increased similar to section 5.4.1. Again a solution could be to build beamformer cores from 32 Memory and CXM Units. The layout of one input vector from 2,048 antennas across 32 memory systems is shown in Figure 23. As in section 5.4.1, a beamformer core memory can be built from 20 M20K blocks, for a total of 1,128 M20K blocks (Arria-10).

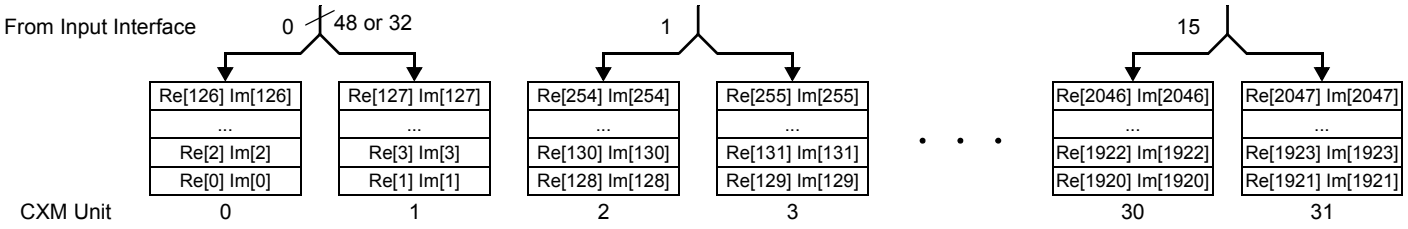

Figure 23: Data Distribution for 2,048 Antennas

The Weightfactor Memory again has to provide 32 weightfactors in parallel, however, in this configuration only on 8 channels. Thus, for the Arria-10 CORDIC system this bus is 6,144 bits wide, for the Stratix-10 system 4,096 bits. The depth is 64*32 (Arria-10) or 64*64 (Stratix-10) such words. The Weightfactor Memory can be built using 616 (Arria-10) or 824 (Stratix-10) M20K blocks.
The Weightfactor Fan-Out Register Tree consumes 331,776 flipflops on the Arria-10 FPGA, or 19% of available resources.
Each beamformer core produces one beam sample every 32 (Arria-10) or 64 (Stratix-10) clocks. This amounts to an average bit rate of 720Mb/s (Arria-10) or 500Mb/s (Stratix-10). Thus we can use one Output Unit (see Figure 15) per 8 (Arria-10) or per 16 (Stratix-10) beamformer cores.
Table VIII shows a summary of performance and resource consumption of this approach.

TABLE VIII: INCREASING THE NUMBER OF ANTENNAS

Architecture	Observing Bandwidth	# of Antennas	# of Beams	# of Beamformer Cores	# of 18x19 Multipliers	# of M20K	# of Output Units	Output Data Rate (Gb/s)
Arria-10 CORDIC	384MHz	1,024	64	47	1,504	2,010	6	33.84
Stratix-10	384MHz	2,048	64	90	11,520	4,024	6	45.0

5.5.2 System-Level

Multiple of the original beamformer systems can be operated in parallel to process more antennas. Provided, however, that there is a downstream device that performs the final accumulation. As was detailled in section 4.8 there are six output stages on each beamformer-FPGA, each one connecting to one 10GbE-line. Net data rate per line is 5.76Gb/s.
An example configuration could be four systems processing 2,048 antennas. The accumulation device shall also be built from Uniboards.
For this configuration, the outputs of four corresponding beamformer-FPGAs need to be summed up and sent off to the central compute facility. Thus, the accumulating device needs 24 input-channels for each set of four beamformer-FPGAs. The output data rate is of course equal to that of a single beamformer-FPGA, and requires again six 10GbE-channels.
As can be seen in Figure 3, each FPGA exposes 24 FD lines towards the optical transceivers, and 48 FD lines towards the midplane. By using breakout-boards that simply connect 24 FD lines from the midplane to QSFP-cages, one Uniboard can accumulate the outputs of 3 beamformer-Uniboards. Since a beamformer system contains 12 beamformer-Uniboards, we would need 4 accumulator-Uniboards, 8 breakout-boards and a special midplane for this kind of post-processing.
Processing requirements are moderate. Each beamformer-FPGA outputs SP-FP complex beam samples at a rate of 360MHz / 32 * 47 = 528.75e6 numbers per second. Each summation requires 6 FP operations, corresponding to 3.1725GFlops, or roughly 10GFlops for each accumulator-FPGA. This is well within reach.
Figure 24 shows a 3D sketch of a breakout board and the accumulation unit. The midplane has the same dimensions as in Figure 16.

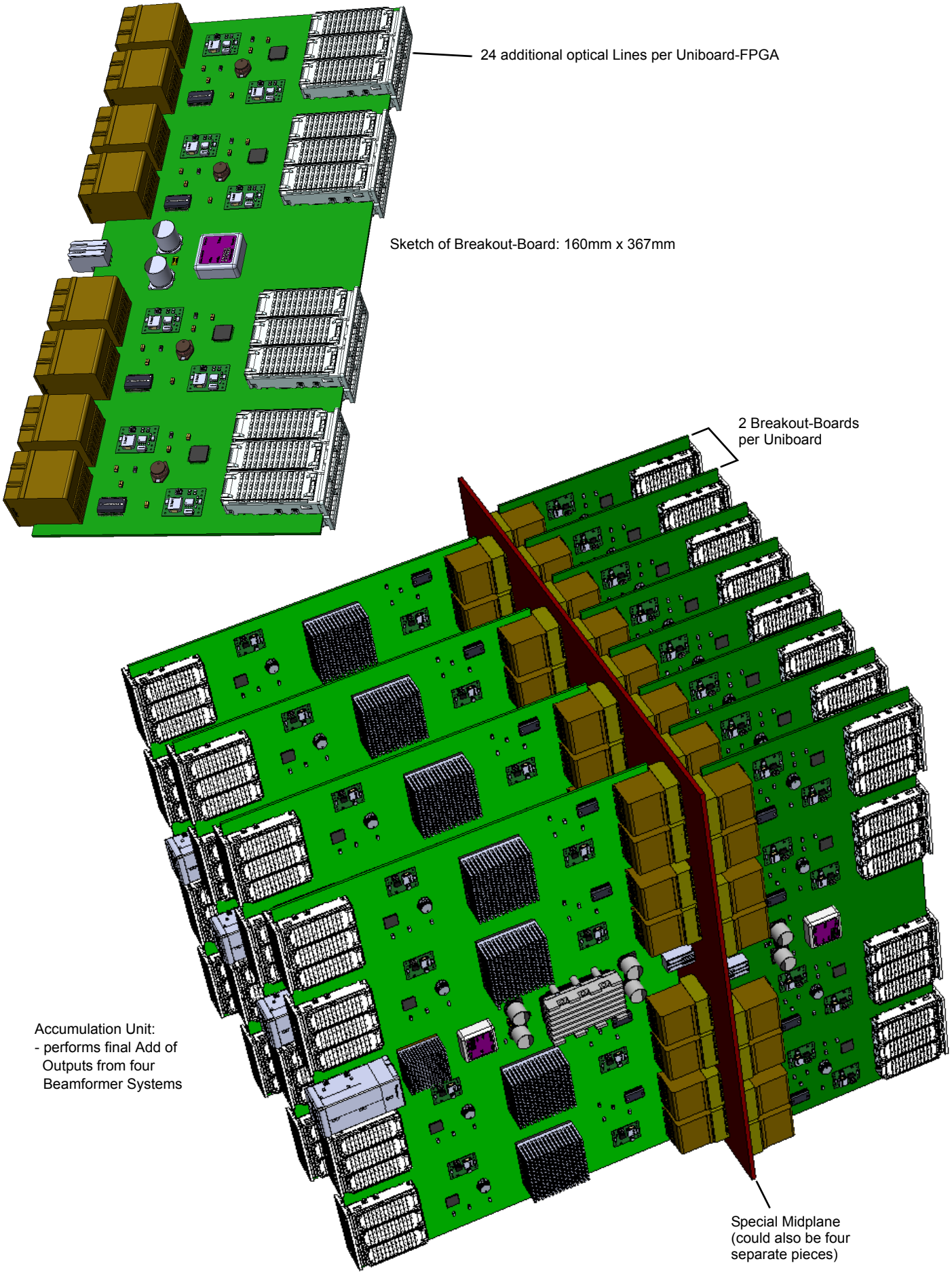

Figure 24: Sketch of Breakout Board and Accumulation Unit

5.6 Scalability Summary

On the microarchitectural level we have explored several points in the design space, as shown in Figure 25. The far-out points require the more expensive Stratix-10 FPGAs. It stands to reason that this configuration can also implement other points in the spanned design space. We have also shown how multiple beamformer systems can be operated in parallel to be suitable for certain scenarios in the SKA environment.

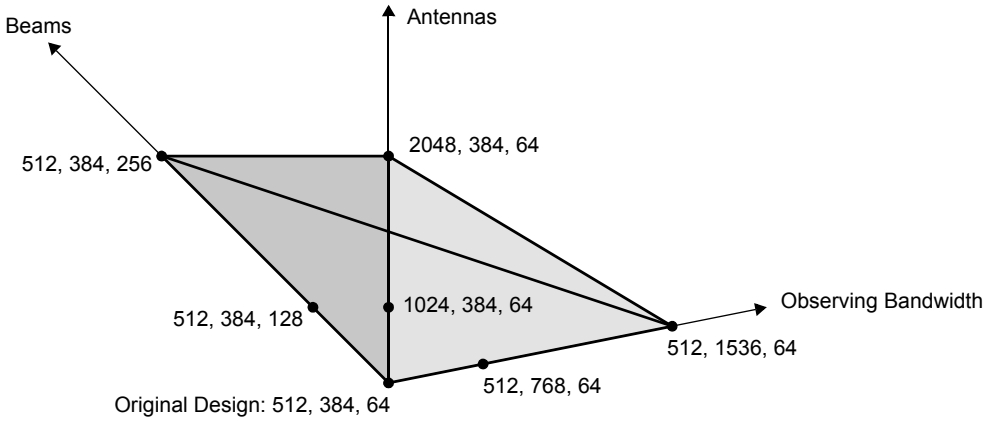

Figure 25: Design Space

6 ALTERNATIVE ARCHITECTURES

One class of beamformers, called FFT-based beamformers, is not considered here since their use is mostly in radar mapping. Apart from the presented architecture, which is commonly called "subband beamformer", the most prominent other architecture is called "ring beamformer". The different architectures are shown in Figure 26, which was adopted from [13].

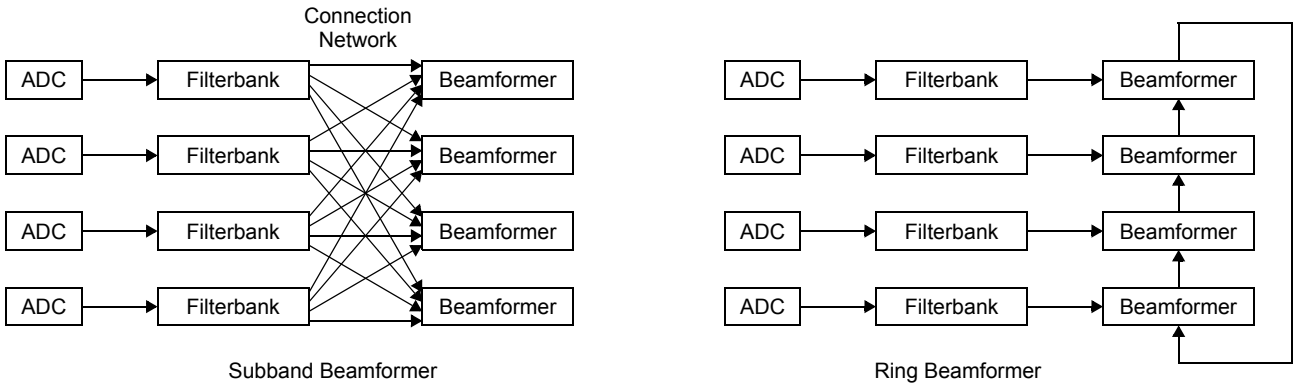

Figure 26: Subband and Ring Beamformers

The obvious advantage of the ring architecture is its simplicity (the cross-connect is avoided) and the fact that the number of antennas scales easily. However, the bandwidth of the ring connect (which also defines the maximum output rate) can quickly become a limiting factor when the number of beams increases. If B beams are to be generated, then for each input sample B output samples need to be put on the ring. Thus, this architecture is most suitable for single-beam beamformers. In case of the Uniboard^2, the ring is a relatively low-bandwidth channel, so this architecture is probably not optimal.