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 8.5

Firmware Design Document Beam Former

Due date of deliverable:	January 2014
Actual date of deliverable:	January 2014
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: 1.0
WP: 8
Authors: Guenter Knittel
MPIfR Report: MPIfR-UB2-01/2014

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

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, Reinhard Keller, Gundolf Wieching

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
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

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 ... 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 ... 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.

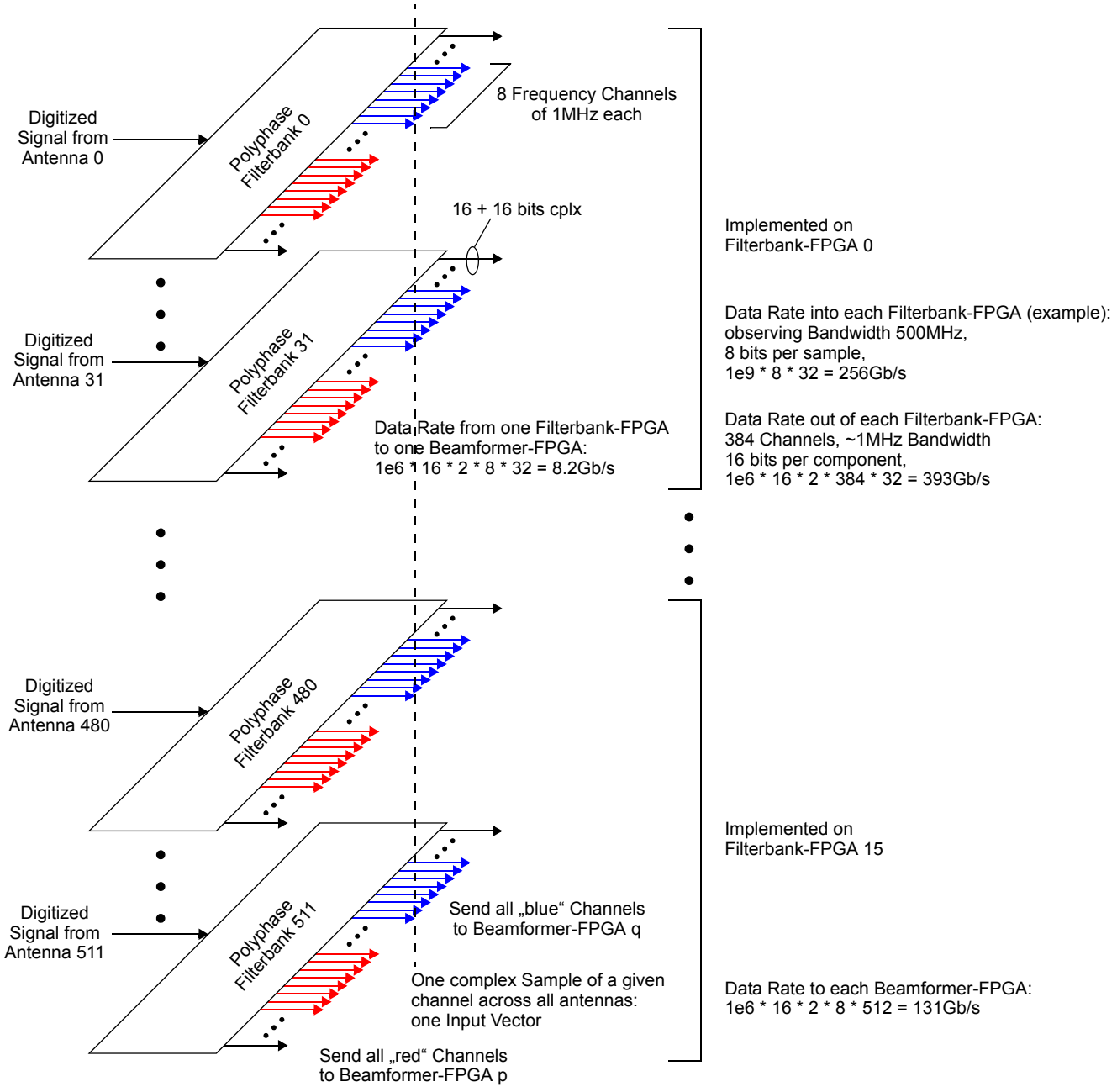

Figure 1: General Filterbank Architecture

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.

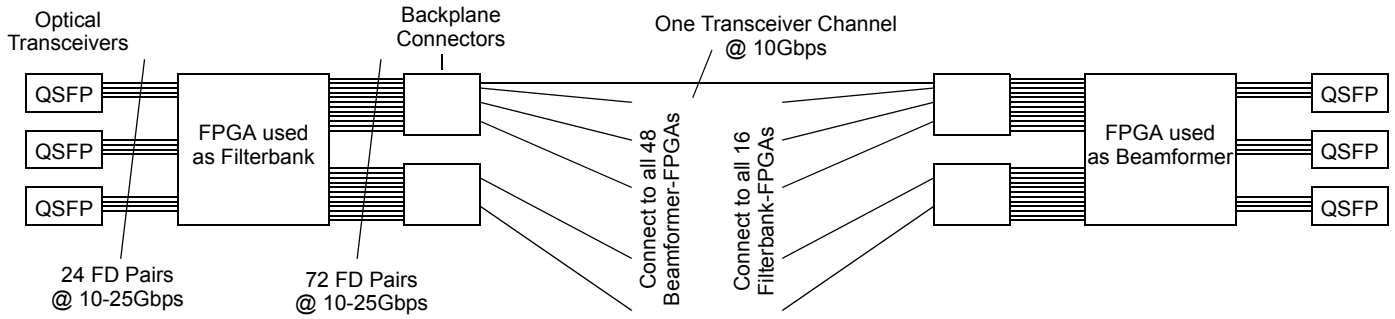

Figure 2: Interconnect Structure

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 3.

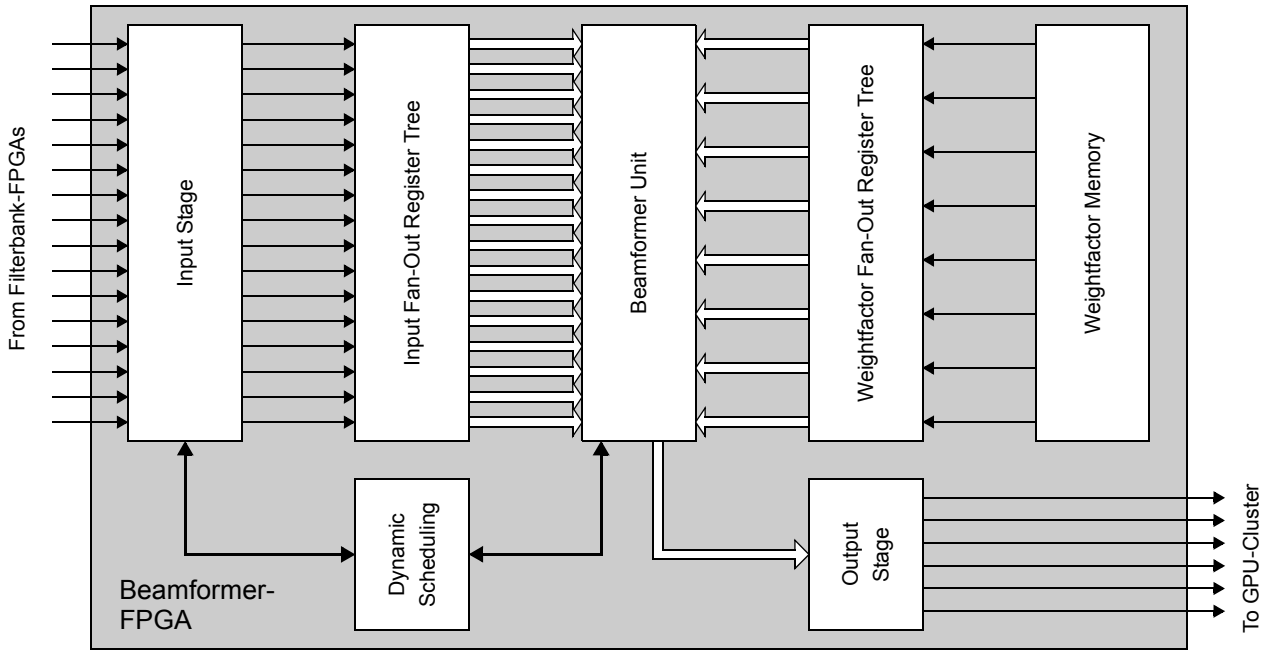

Figure 3: Beamformer-FPGA Block Diagram

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 4. See also Figure 1 for sample arrangement. Sample origin is given by [Antenna Number; Frequency Channel Number].

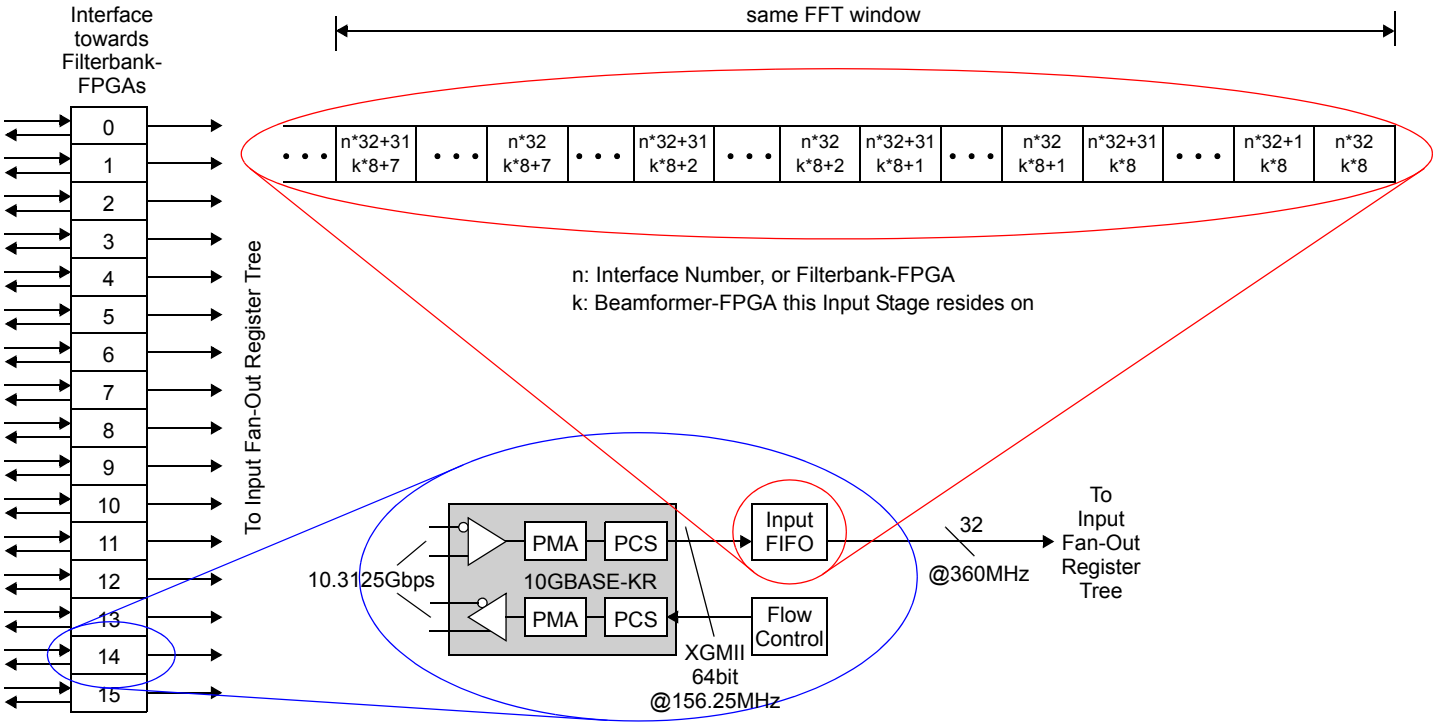

Figure 4: Input Stage Block Diagram

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 27,136 flipflops. This is 1.6% out of 1,708,800 available flipflops. The arrangement is sketched out in Figure 5. Note that the latency introduced by this structure is of no concern.

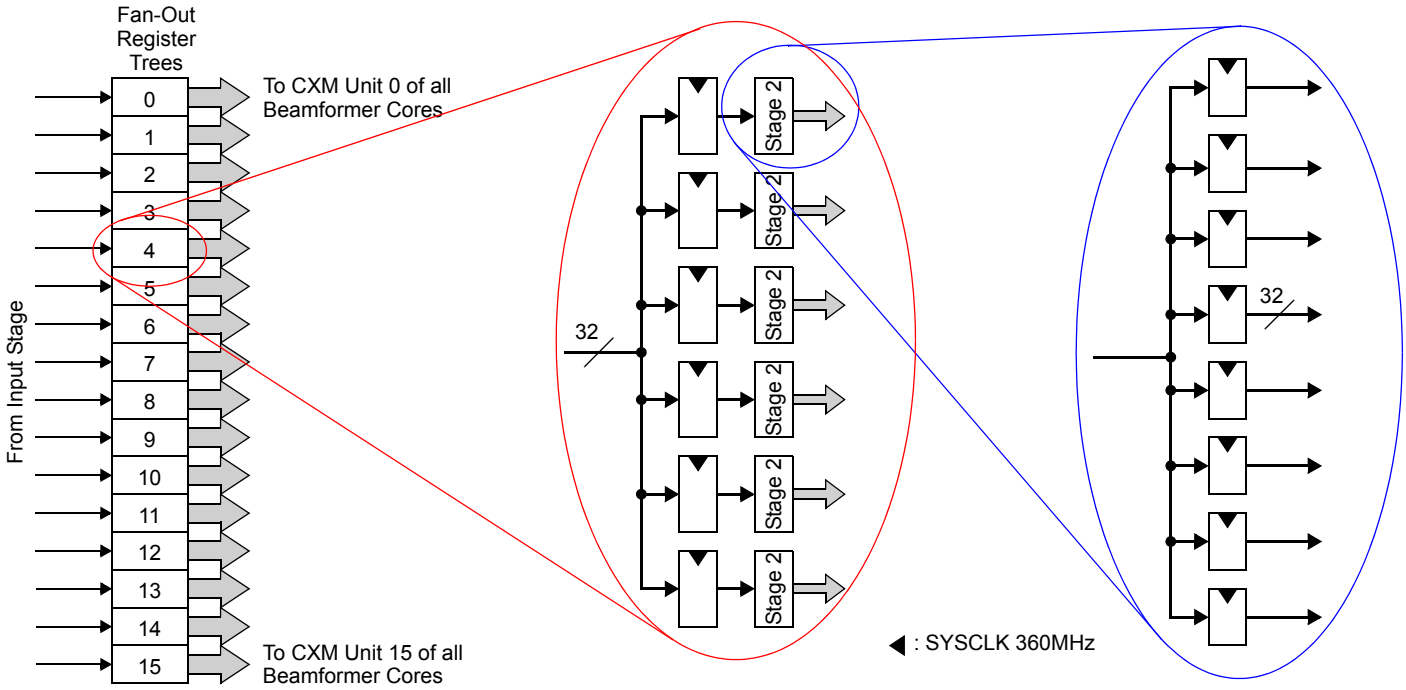

Figure 5: Input Fan-Out Register Tree

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 utilizationof 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 3). 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 6.

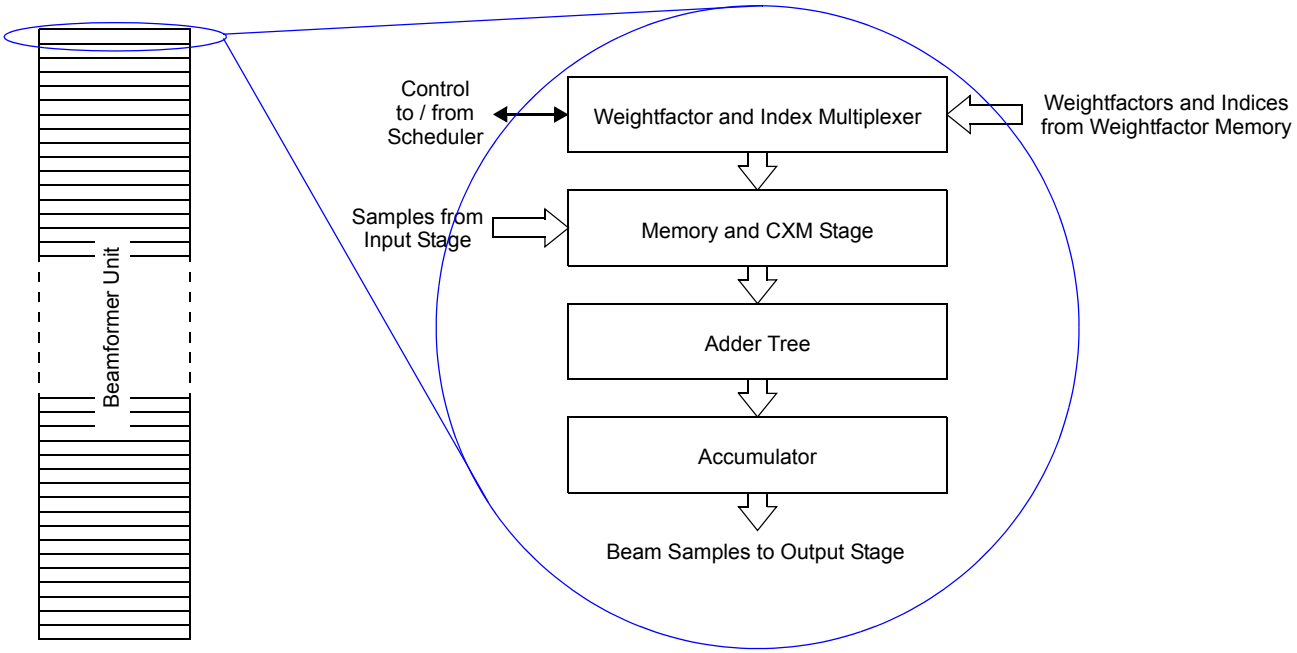

Figure 6: Beamformer Core Block Diagram

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.

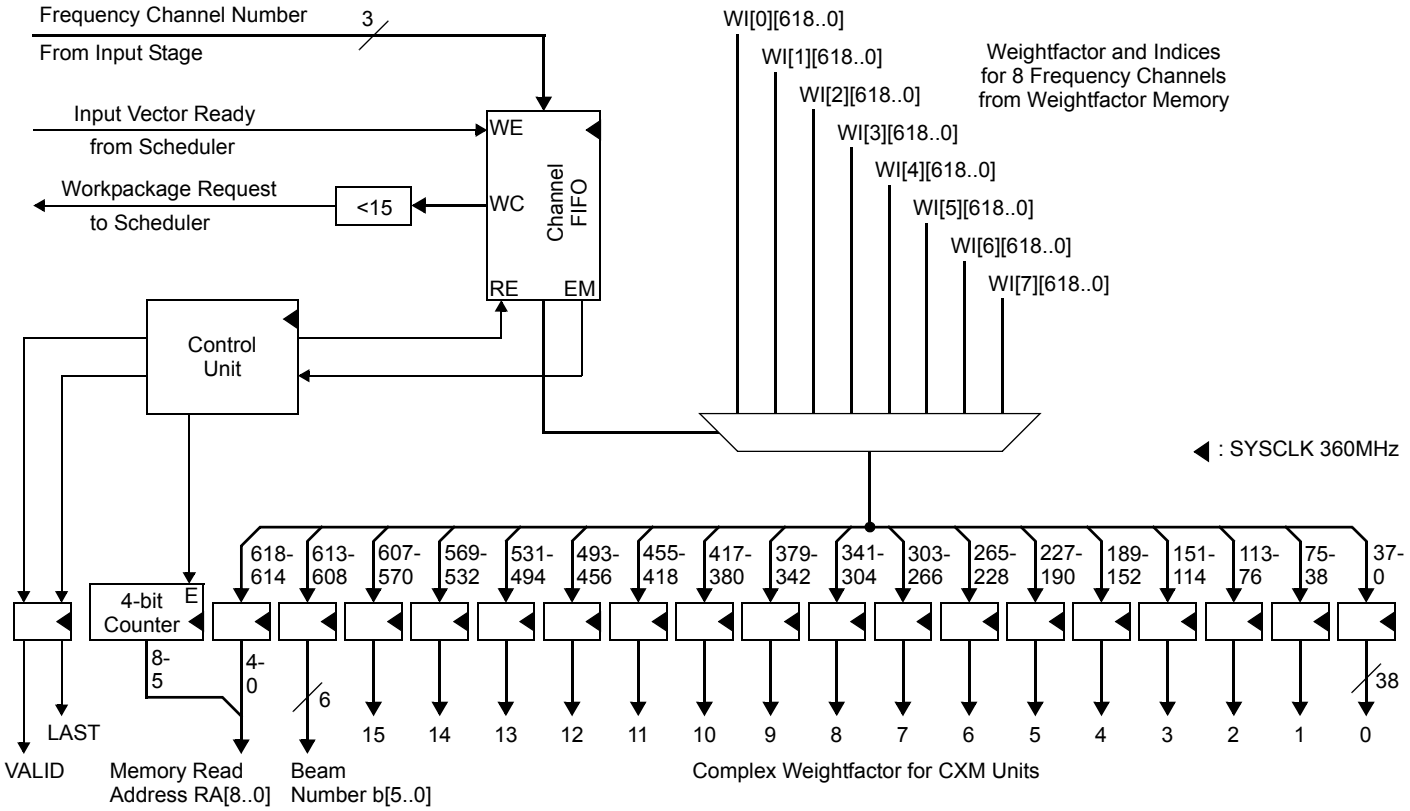

Figure 7: Weightfactor and Index Multiplexer

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 8.

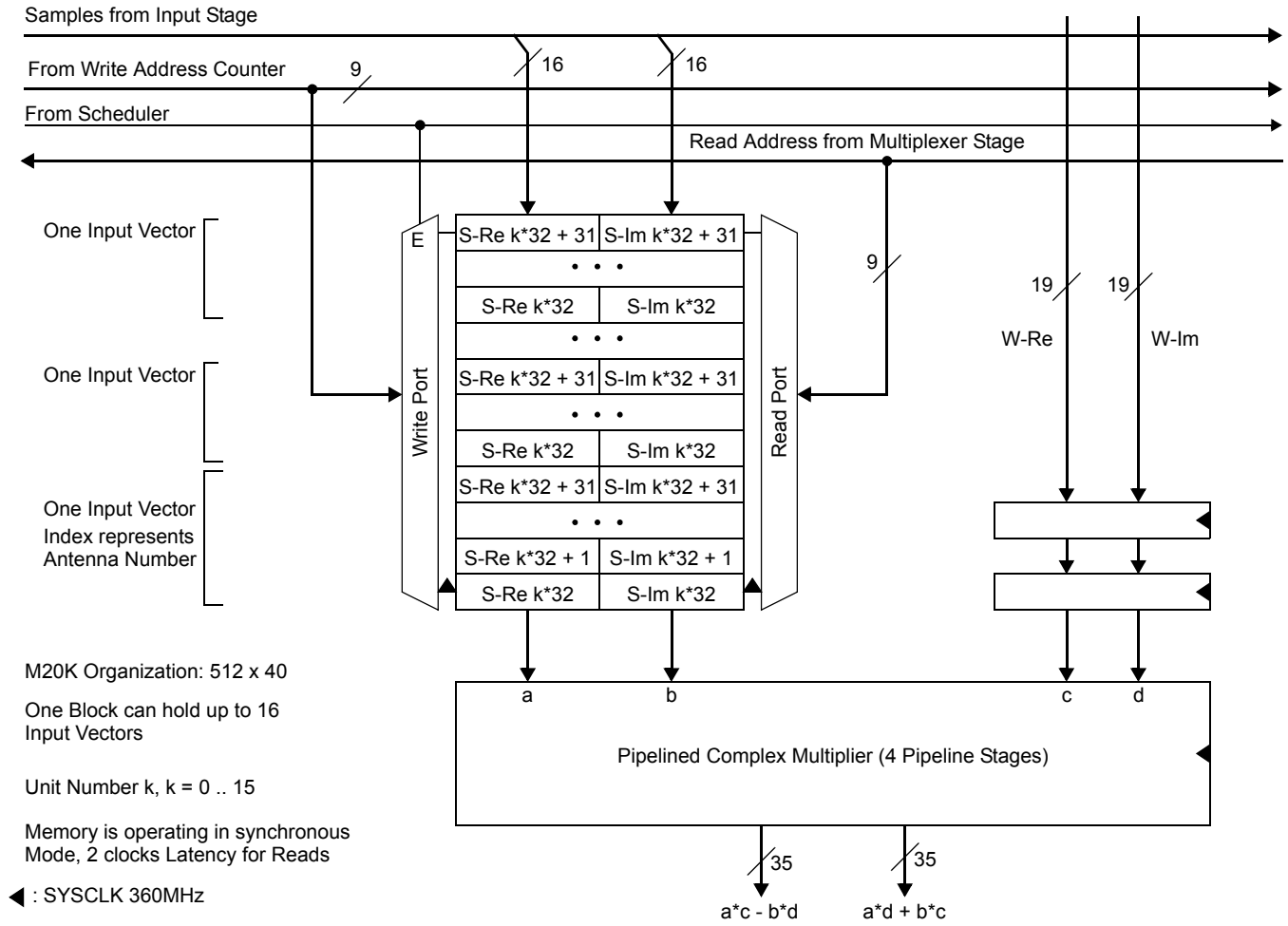

Figure 8: One Sample Memory and associated CXM Unit

The entire stage is shown in Figure 9. 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.

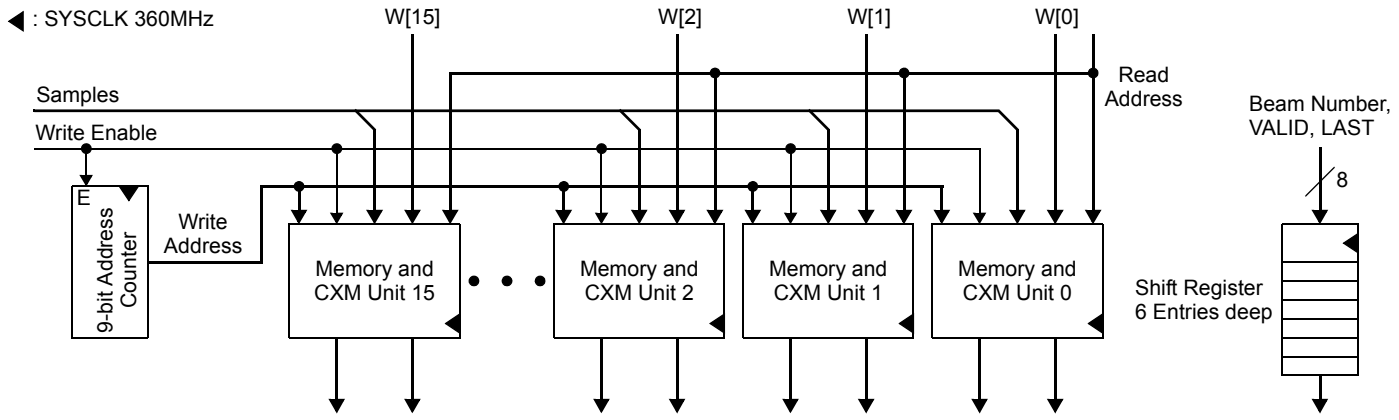

Figure 9: Memory and CXM Stage

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.

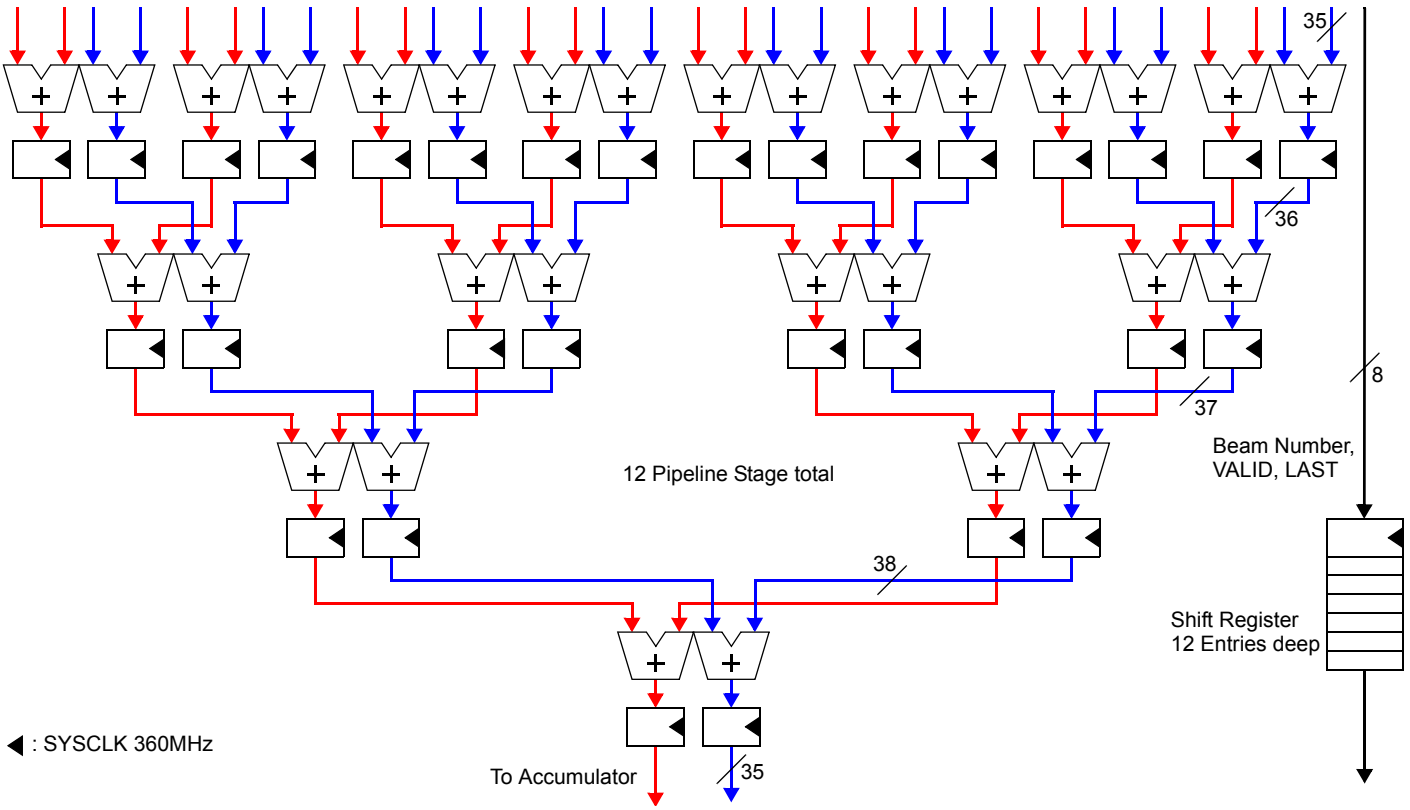

Figure 10: Adder Tree

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 11.

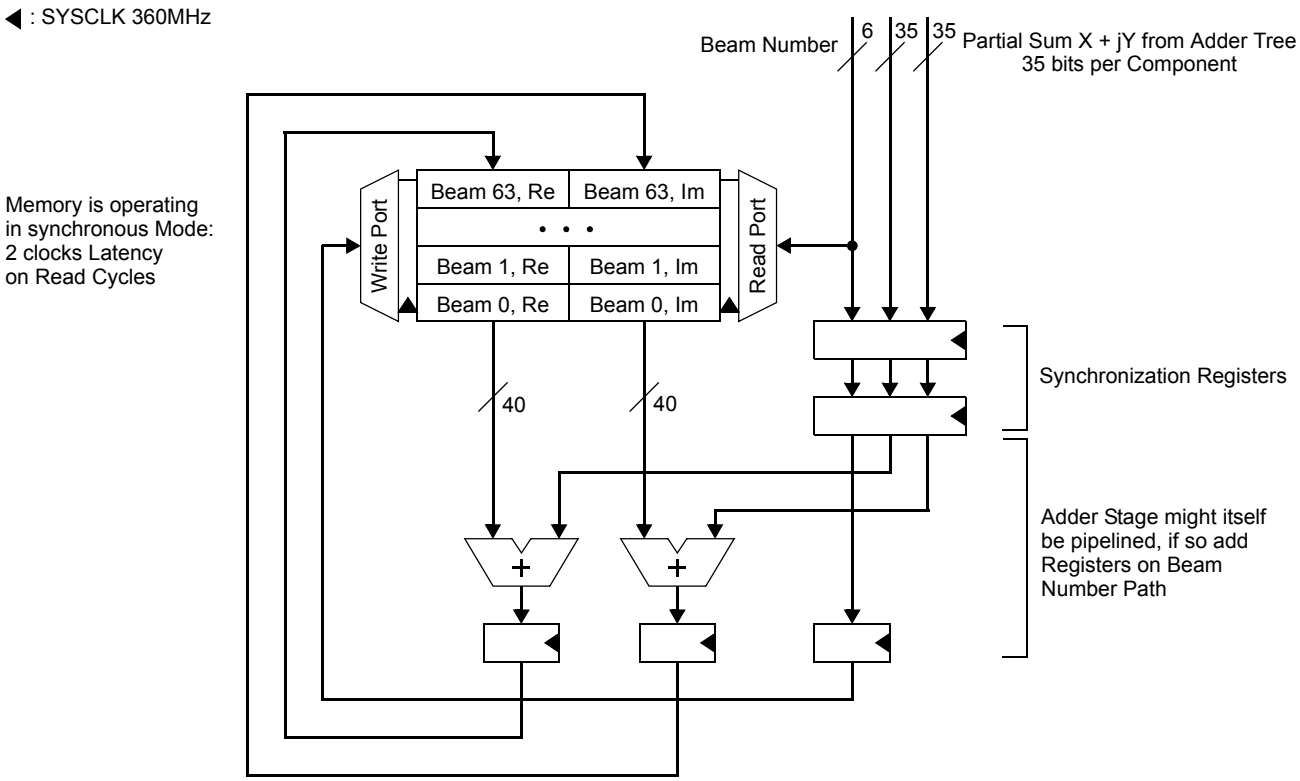

Figure 11: Accumulator Stage (Basic Form)

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 12. 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.

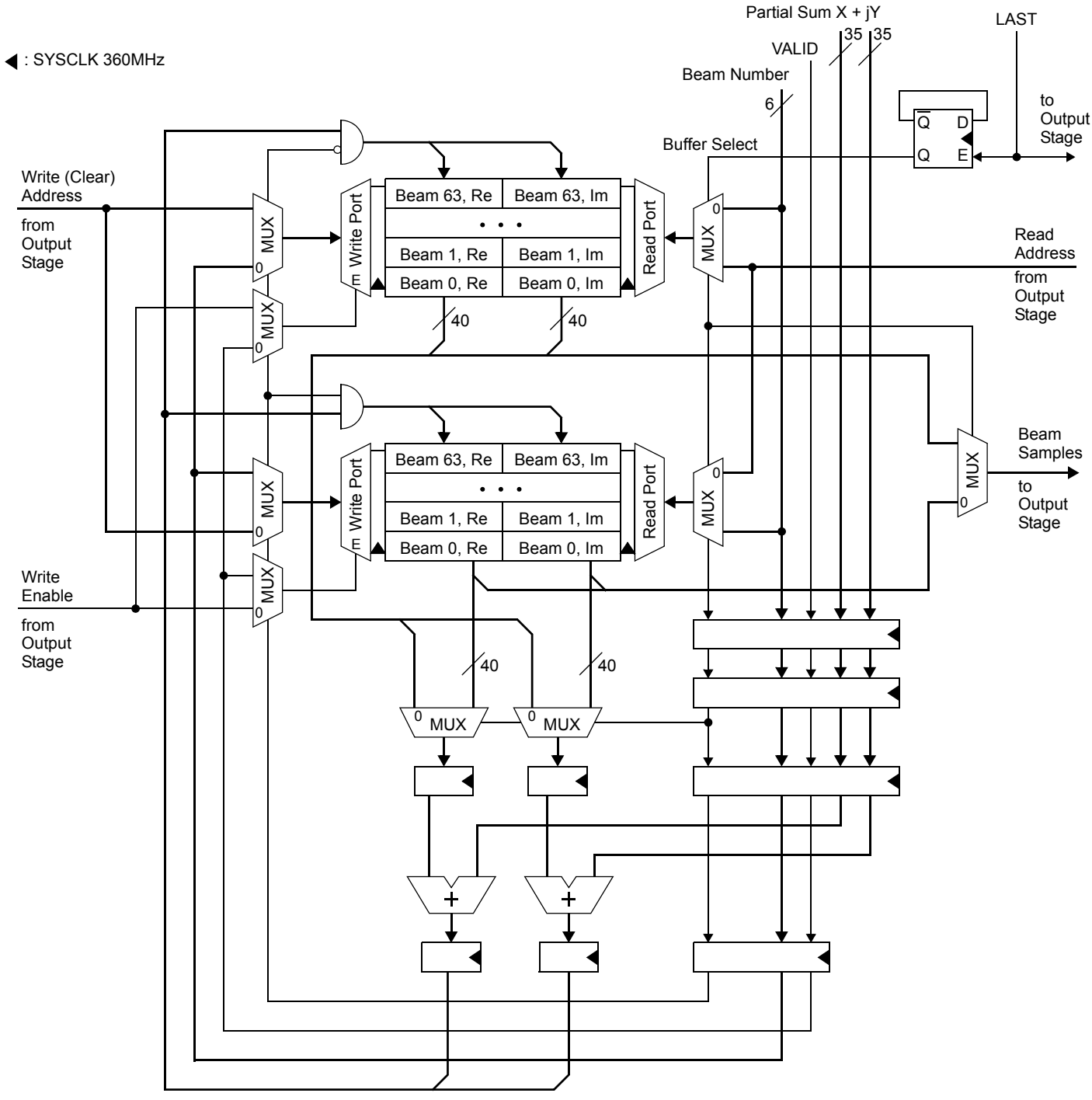

Figure 12: Double-Buffered Accumulator Stage

In Figure 12, 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 13 for a block diagram of one weightfactor memory bank.

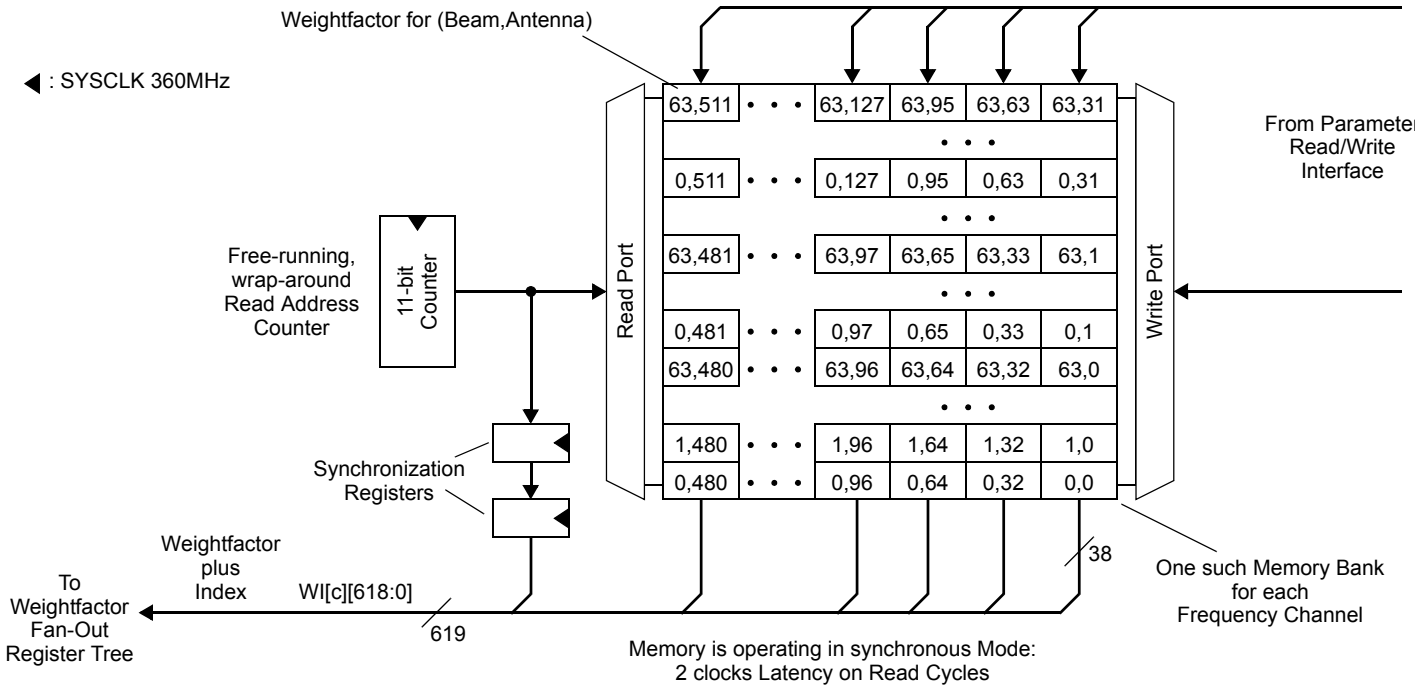

Figure 13: One Weightfactor Memory Bank

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 5. 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 14. 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.

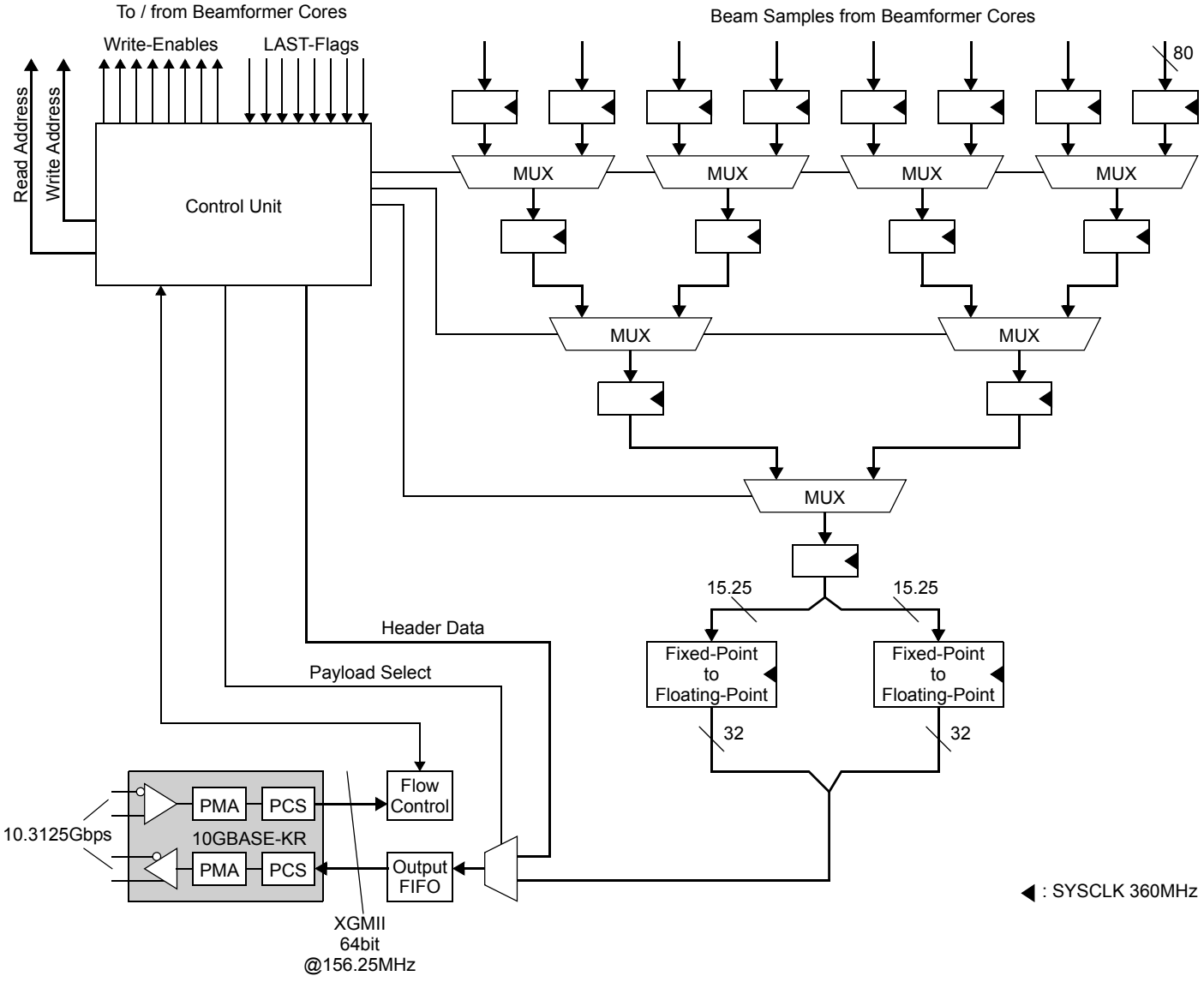

Figure 14: Output Unit

5 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.