An HLS-based Beamformer using Xilinx Alveo Card

Speaker: Yunpeng Men

Max-Plank-Institute for Radio Astronomy

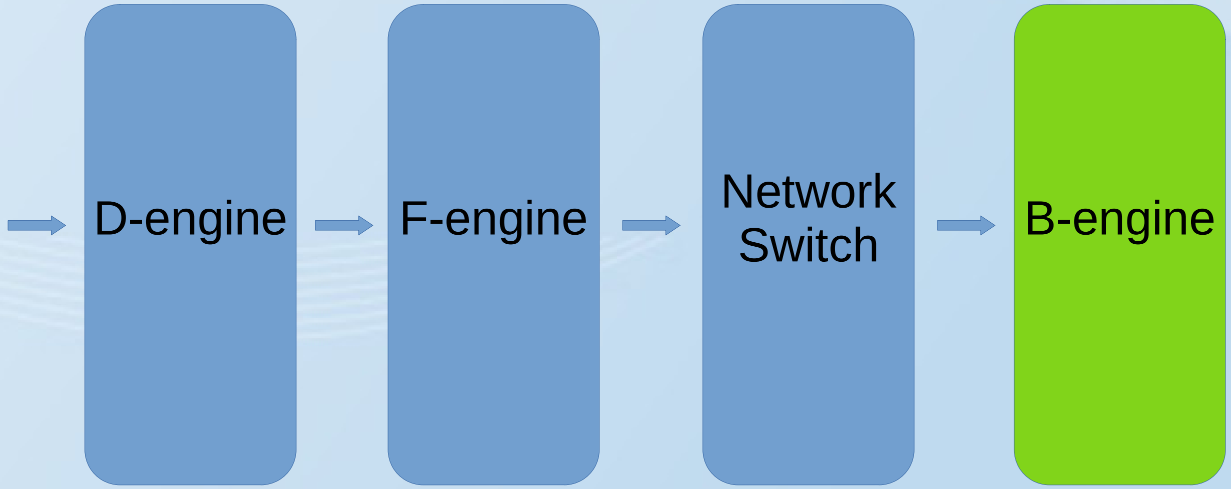
Outline

- MPIfR CryoPAF Project
- High-Level-Synthesis (HLS)
- Beamformer design
- Kernels
  - Spead kernel
  - Reorder kernel
  - Corner-turner kernel
  - Beamformer kernel
- Summary

MPIfR CryoPAF Project

- Dual-pol 128 elements (first generation)
- S-band 2-4 GHz
- Bandwidth 1 GHz or 2 GHz
- Total data rate 4 Tb/s or 8 Tb/s

Effersburg 100m Radio Telesope



Why FPGA?

Xilinx Alveo U55C

- Network bandwidth requirements scale by N
- Computing requirements scale by N x N
- For ~100 elements beamforming, bandwidth is the limitation with current accelerators

Nvidia RTX 3090

- On board 100G NIC
- Less servers needed
- Similar fixed costs, but lower energy costs

High-Level-Synthesis (HLS) C++

GCD(a, b) = GCD(b, a mod b)

a	b	a mob b
105	77	28
77	28	21
28	21	7
21	7	0

```
int gcd(int Ain, int
Bin)
{
    int a = Ain;
    int b = Bin;
    while (b != 0)
    {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return a;
}
```

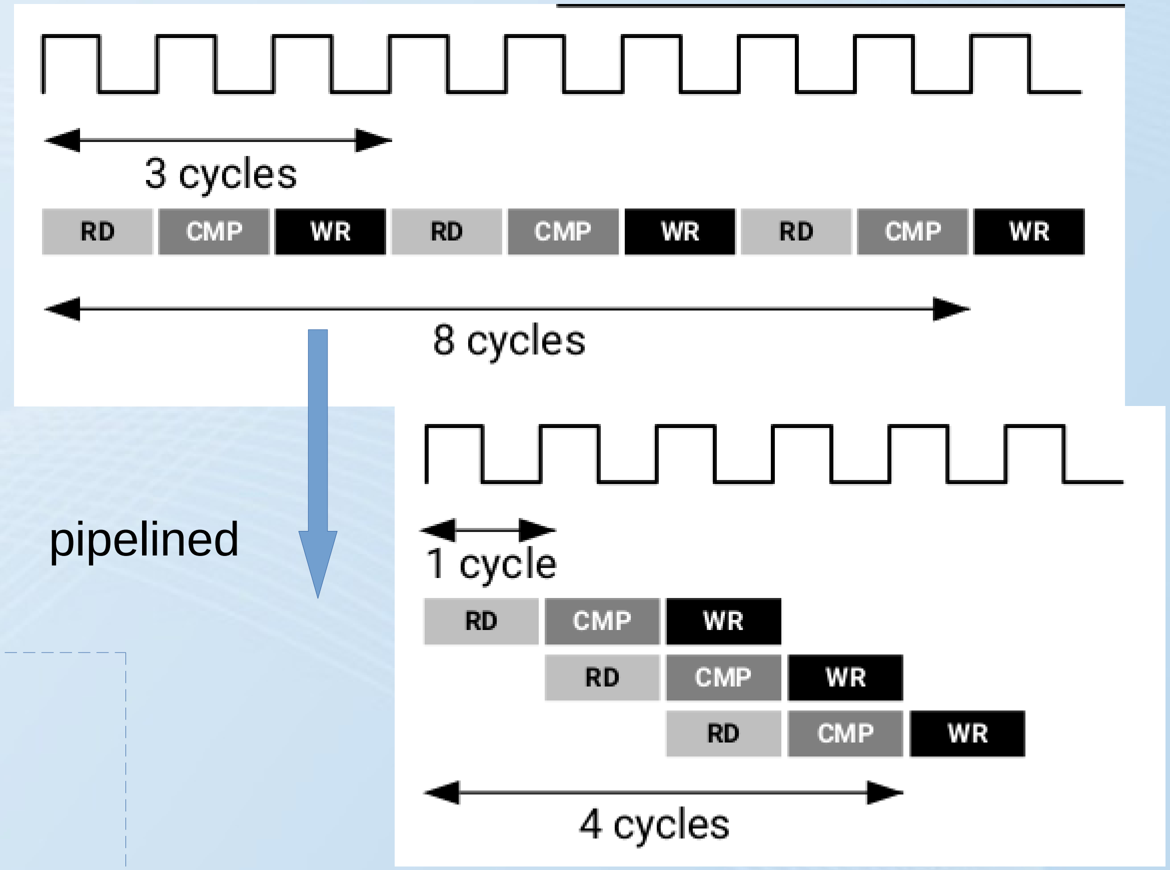
Euclidean algorithm

Verilog

```
module gcd (clk, start, Ain,
Bin, out, done);

input clk, start;
Input [31:0] Ain, Bin;
output reg [31:0] out;
output reg done;

reg [31:0] a, b;

always @ (posedge clk)
    begin
        if (start)
            begin
                a <= Ain; b <=
Bin; done <= 0;
            end
        else if (b == 0)
            begin
                out <= a;done <=
1;
            end
        else if (a > b)
            a <= a – b;
        else
            b <= b – a;
        end
endmodule
```

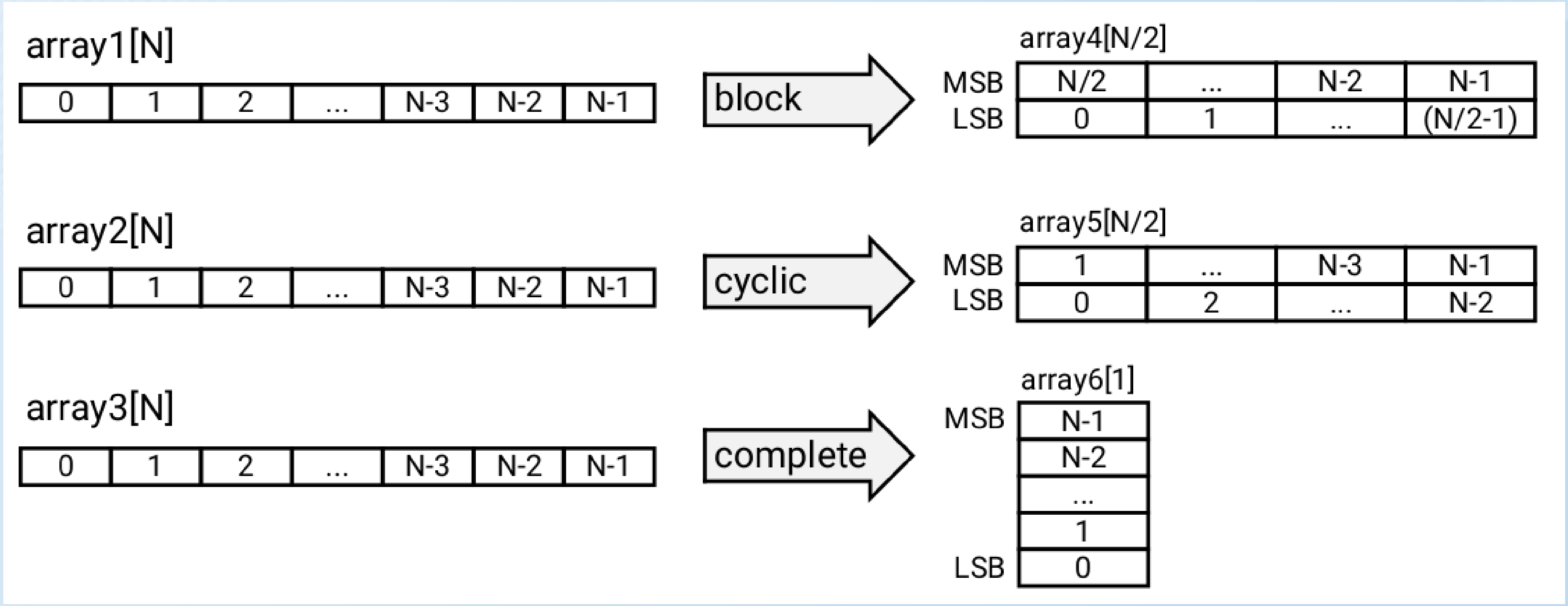
pipeline optimization

```
for (i=2;i>=0;i--) {
#pragma HLS pipeline
            op_Read;
            op_Compute;
            op_Write;
}
```



array reshape

```
int a[N];
#pragma HLS array_reshape
```



AXI4-Lite Interface

```
void example(int *a, int *b)
{
#pragma HLS INTERFACE mode=s_axilite port=return bundle=control
#pragma HLS INTERFACE mode=s_axilite port=a bundle=control
#pragma HLS INTERFACE mode=s_axilite port=b bundle=control

*b += *a + *b;
}
```

Address	Description
0x00	Control signals
0x04	Global Interrupt Enable Register
0x08	IP Interrupt Enable Register (Read/Write)
0x0c	IP Interrupt Status Register (Read/TOW)
0x10	a
0x14	b

AXI4 Interface

```
void example(int *a){
#pragma HLS INTERFACE s_axilite port=return
#pragma HLS INTERFACE mode=m_axi depth=50 port=a

for(int i=0; i < 50; i++){
        #pragma HLS PIPELINE
        a[i] += 1;
}}
```

Easily access the HBM!

AXI4-Stream Interface

```
typedef ap_axiu<32, 0, 0, 0> trans_pkt;
void example(hls::stream< trans_pkt > &A, hls::stream< trans_pkt > &B)
{
#pragma HLS INTERFACE mode=axis port=A
#pragma HLS INTERFACE mode=axis port=B
trans_pkt tmp;
A.read(tmp);
tmp.data += 5;
B.write(tmp);
}
```

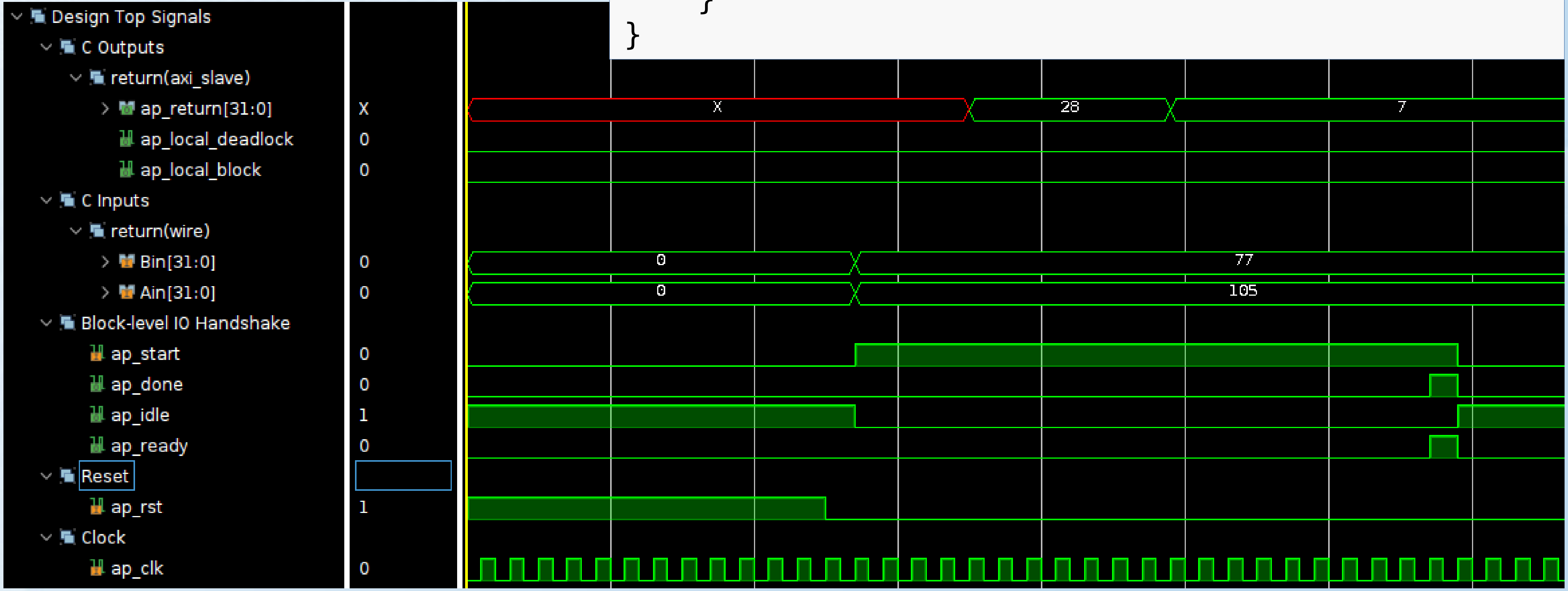
Interface between kernels

Testbench

- C simulation
- RTL cosimulation

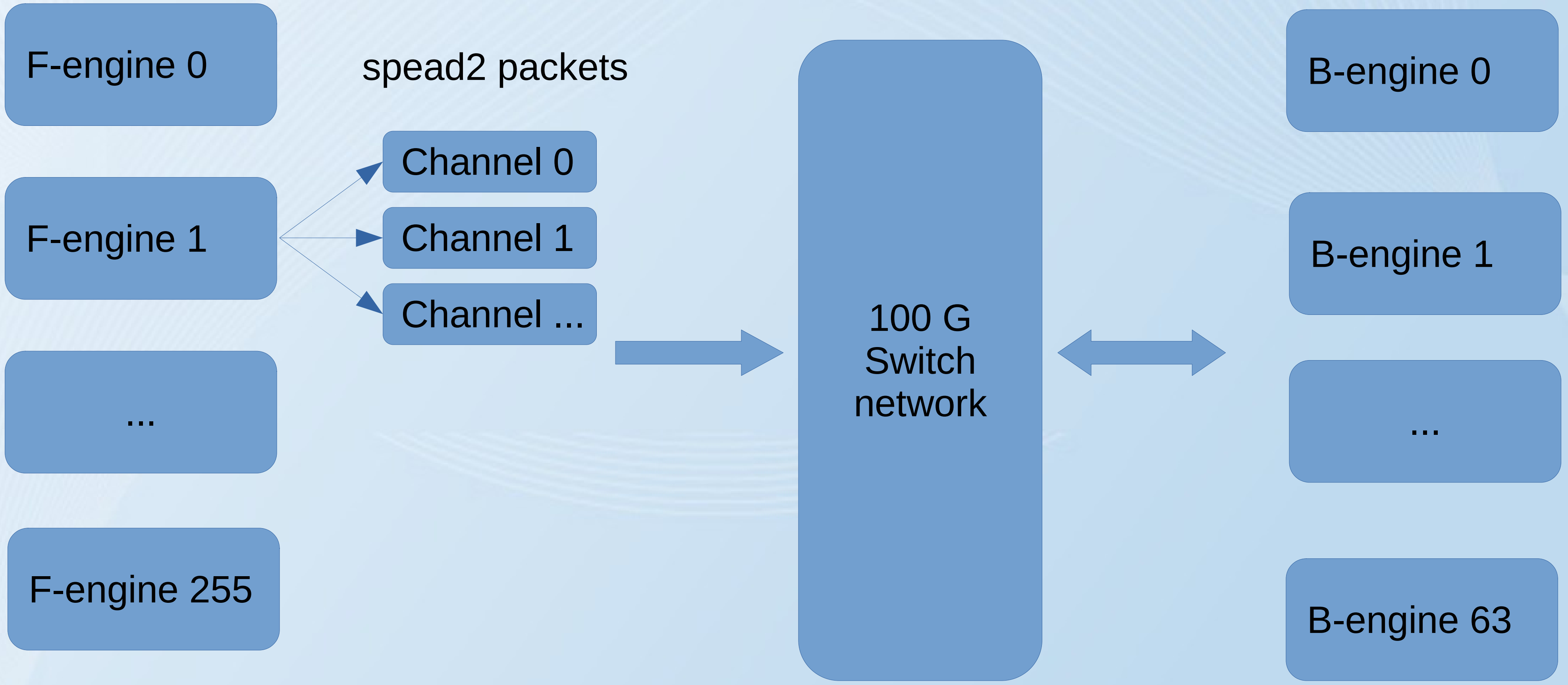
Easily verified!

```
#include "gcd.h"
#include <iostream>
int main()
{
    int a = 105;
    int b = 77;
    int out = gcd(a, b);
    if (out == 7)
    {
        std::cout<<"test passed"<<std::endl;
        return 0;
    }
    else
    {
        std::cout<<"test failed"<<std::endl;
        return -1;
    }
}
```



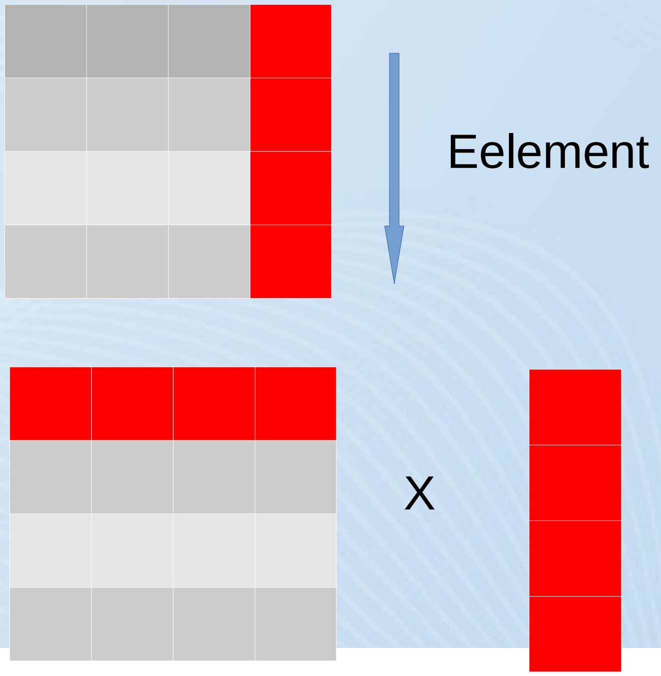
Beamformer design

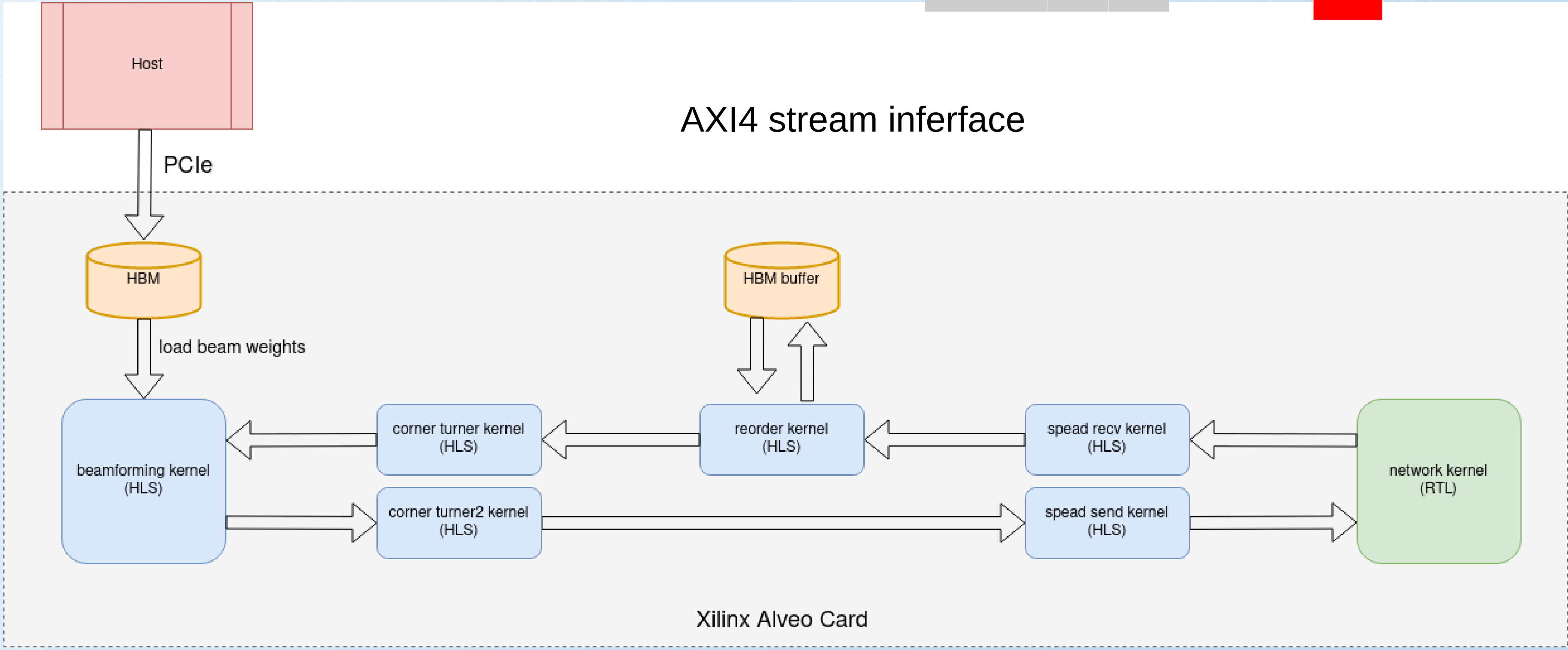
- Each input spead2 packet only contains 1 channel and 1 element, as well as output
- Multi-cast network applied



Issues to be solved:

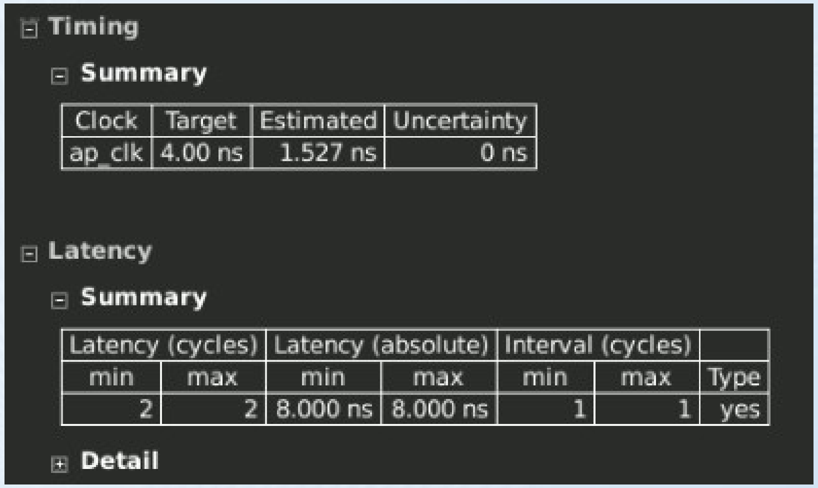
- packet synchronization (reorder kernel)
- packet loss (reorder kernel)
- spead2 unpack and pack (spead kernel)
- “corner turn” problem (corner turner kernel)
- Beamforming (beamformer kernel)





Spead kernel

Specific purpose not general purpose



Timing

Summary

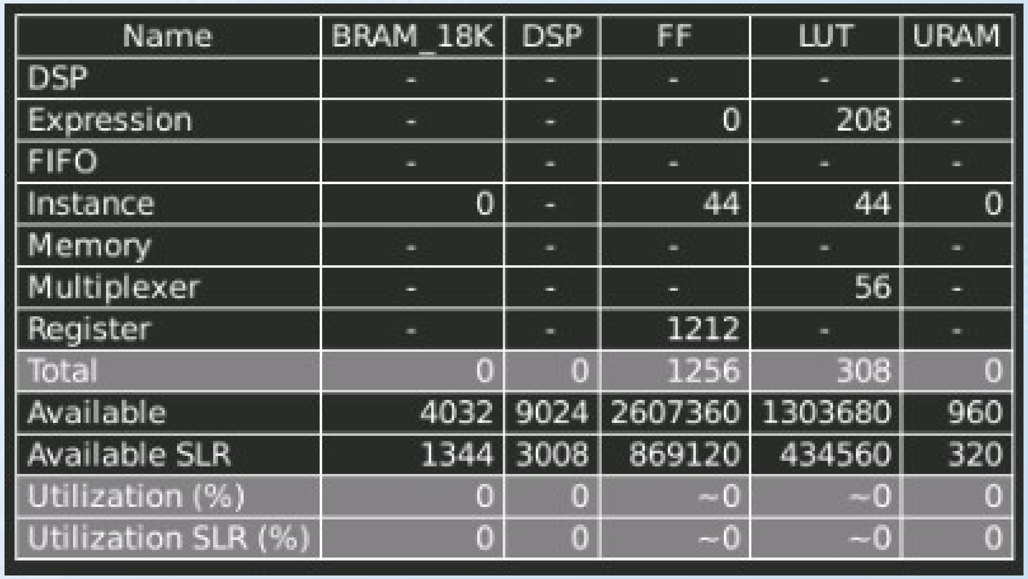
Clock	Target	Estimated	Uncertainty
ap_clk	4.00 ns	1.527 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
2	2	8.000 ns	8.000 ns	1	1	yes

Detail



Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	208	-
FIFO	-	-	-	-	-
Instance	0	-	44	44	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	56	-
Register	-	-	1212	-	-
Total	0	0	1256	308	0
Available	4032	9024	2607360	1303680	960
Available SLR	1344	3008	869120	434560	320
Utilization (%)	0	0	~0	~0	0
Utilization SLR (%)	0	0	~0	~0	0

spead packet format (https://github.com/ska-sa/spead2)

0x53	0x04	0x02	0x06	0x00		0x0b	
0x8001				heap id			
0x8002				heap size			
0x8003				heap offset			
0x8004				payload length			
...				timestamp			
...				clipping cnt			
				order vector			
...				...			
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

Remove header

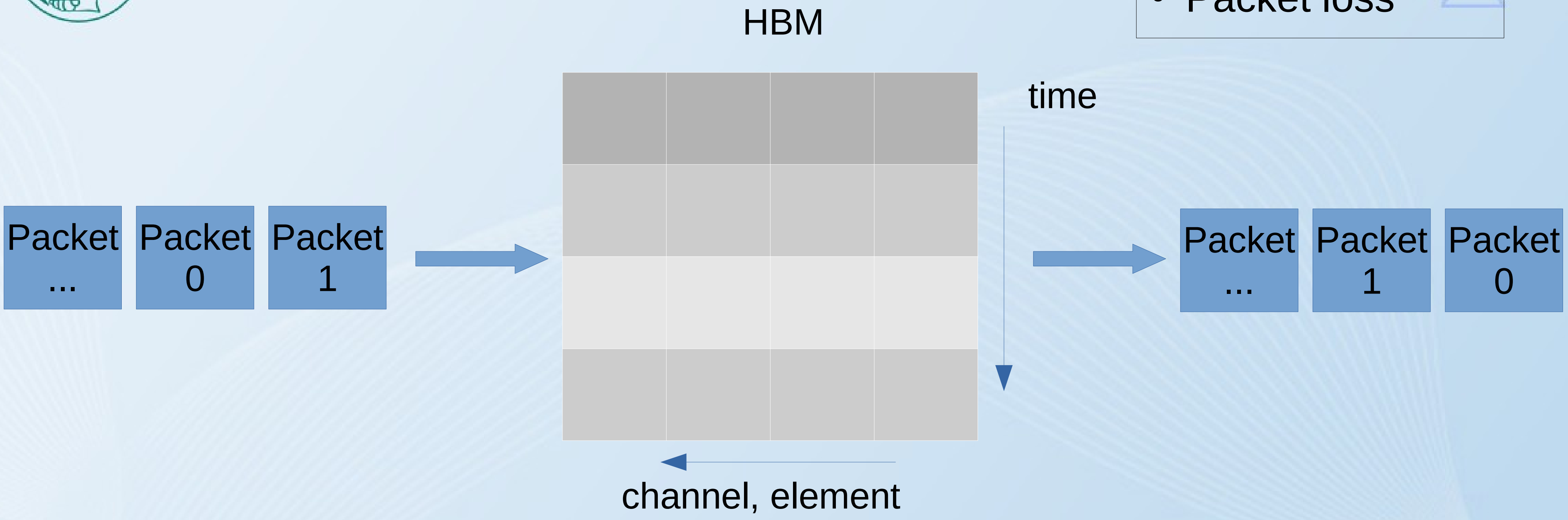
Side channel

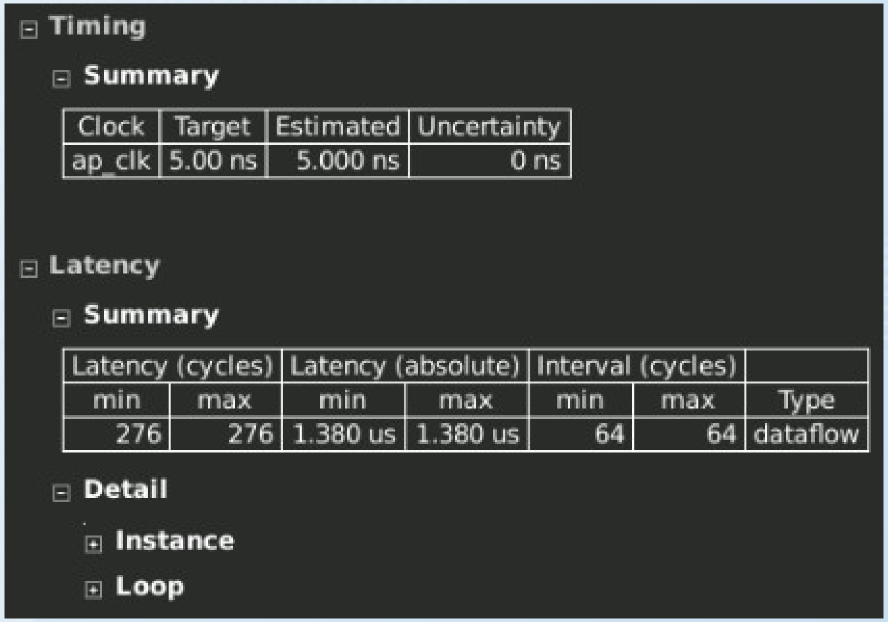
12	8	4	0
13	9	5	1
14	10	6	2
15	11	7	3

Timestamp,channel_id,element_id

Reorder kernel

- Packet buffering
- Packet sync
- Packet loss




Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	5.000 ns	0 ns

Latency

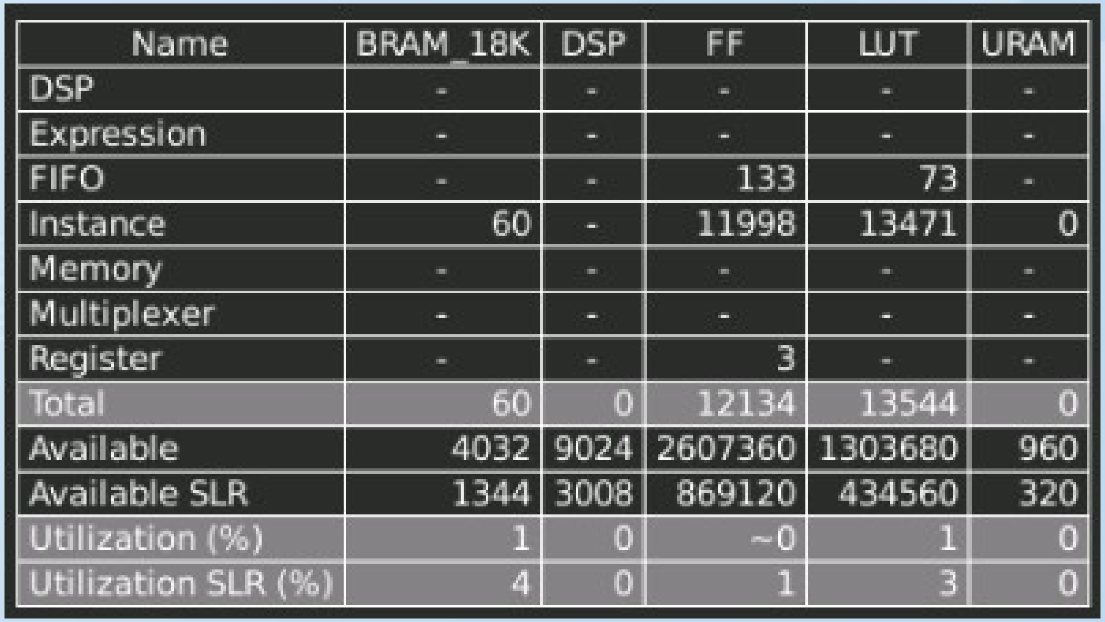
Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
276	276	1.380 us	1.380 us	64	64	dataflow

Detail

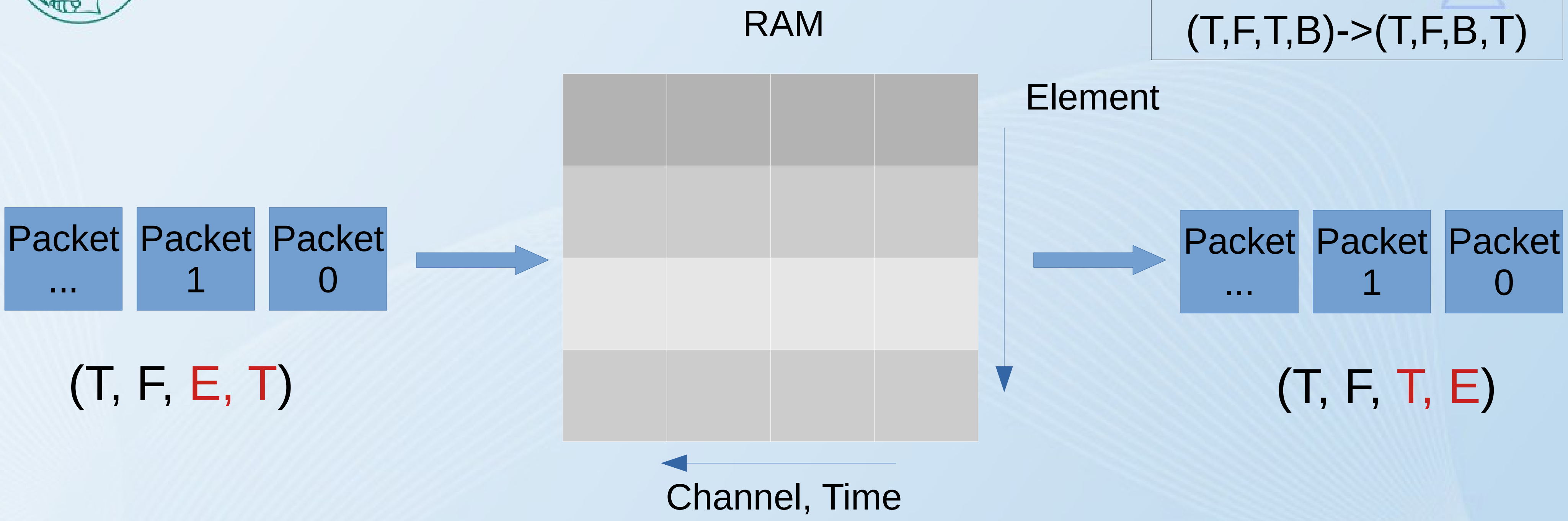
Instance

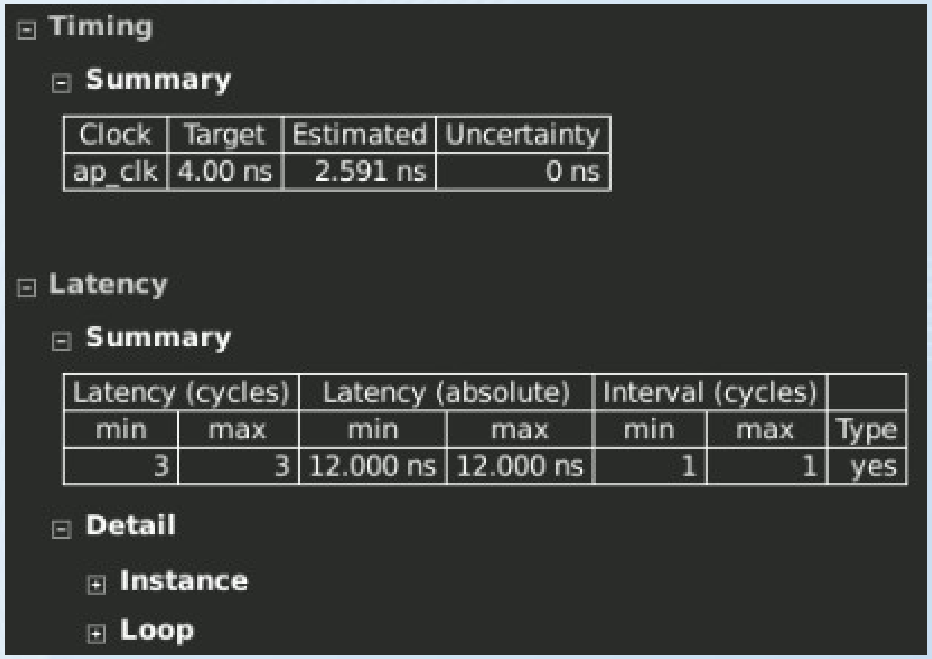
Loop


Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	133	73	-
Instance	60	-	11998	13471	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	3	-	-
Total	60	0	12134	13544	0
Available	4032	9024	2607360	1303680	960
Available SLR	1344	3008	869120	434560	320
Utilization (%)	1	0	~0	1	0
Utilization SLR (%)	4	0	1	3	0

Corner-turner kernel

Data transpose
(T,F,E,T) -> (T,F,T,E)
or
(T,F,T,B)->(T,F,B,T)





Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00 ns	2.591 ns	0 ns

Latency

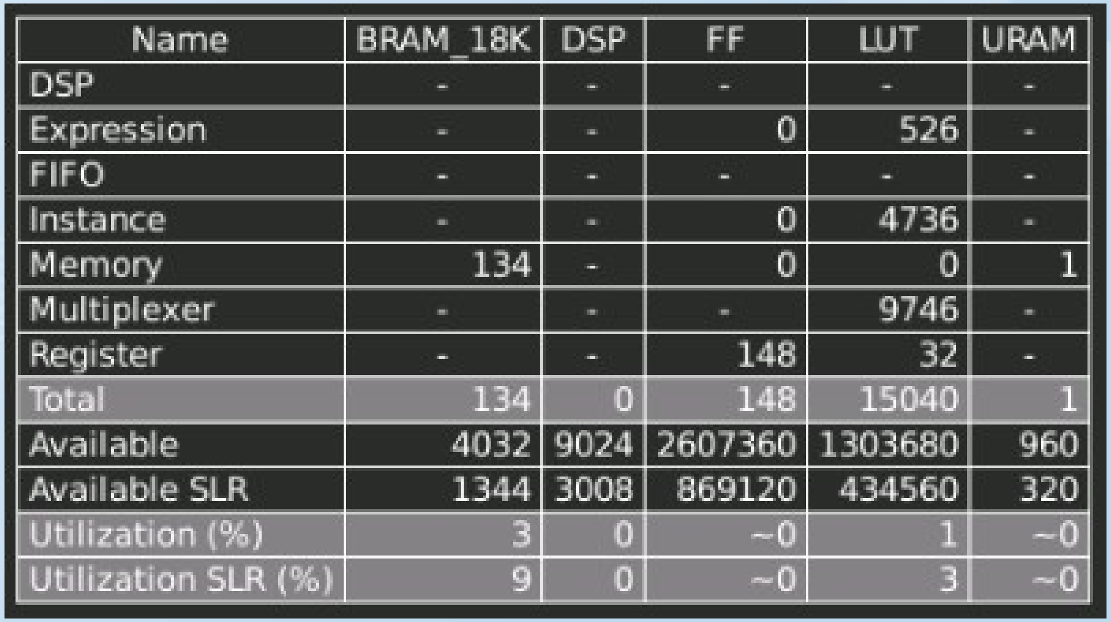
Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
3	3	12.000 ns	12.000 ns	1	1	yes

Detail

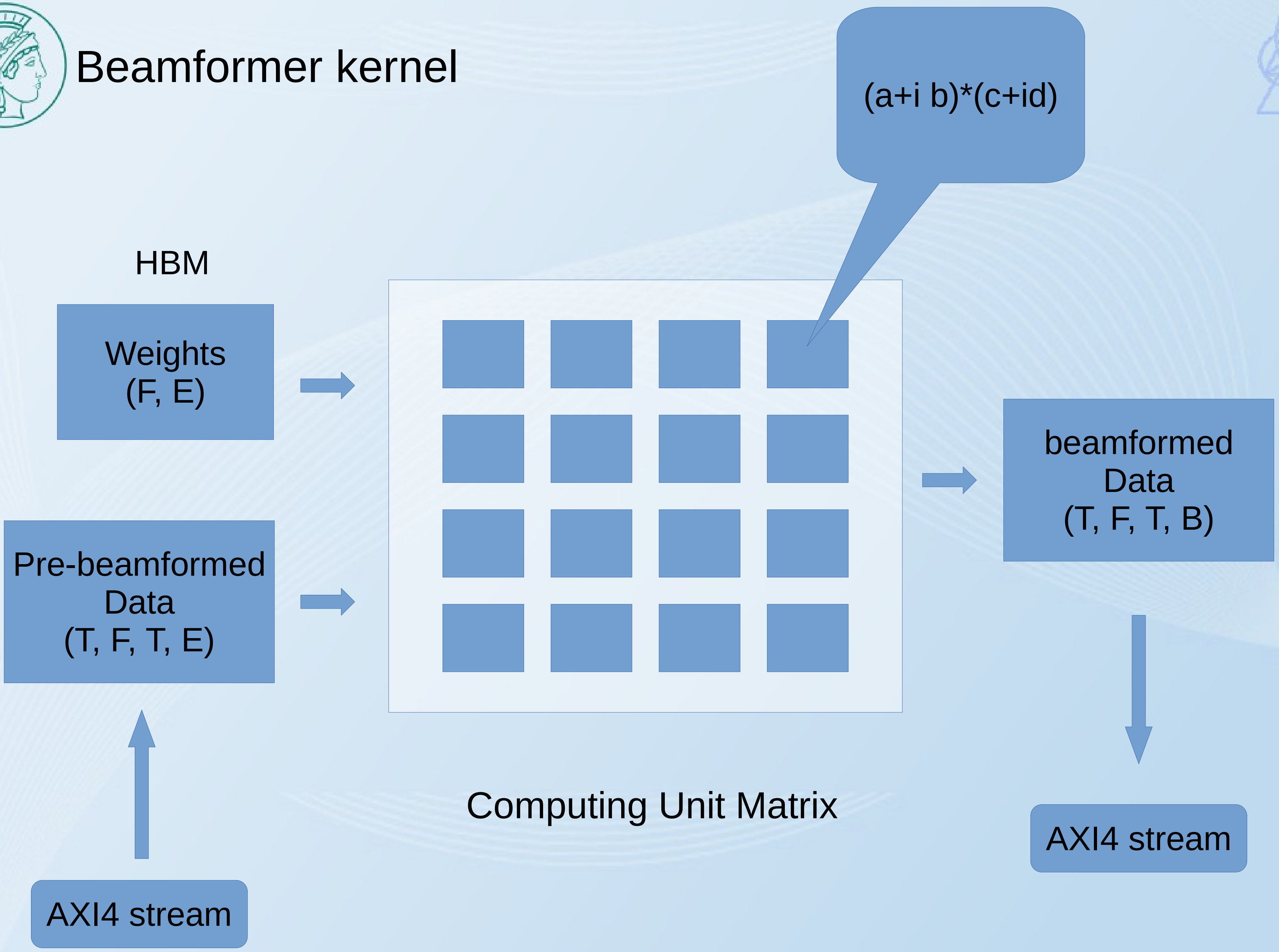
Instance

Loop



Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	526	-
FIFO	-	-	-	-	-
Instance	-	-	0	4736	-
Memory	134	-	0	0	1
Multiplexer	-	-	-	9746	-
Register	-	-	148	32	-
Total	134	0	148	15040	1
Available	4032	9024	2607360	1303680	960
Available SLR	1344	3008	869120	434560	320
Utilization (%)	3	0	~0	1	~0
Utilization SLR (%)	9	0	~0	3	~0

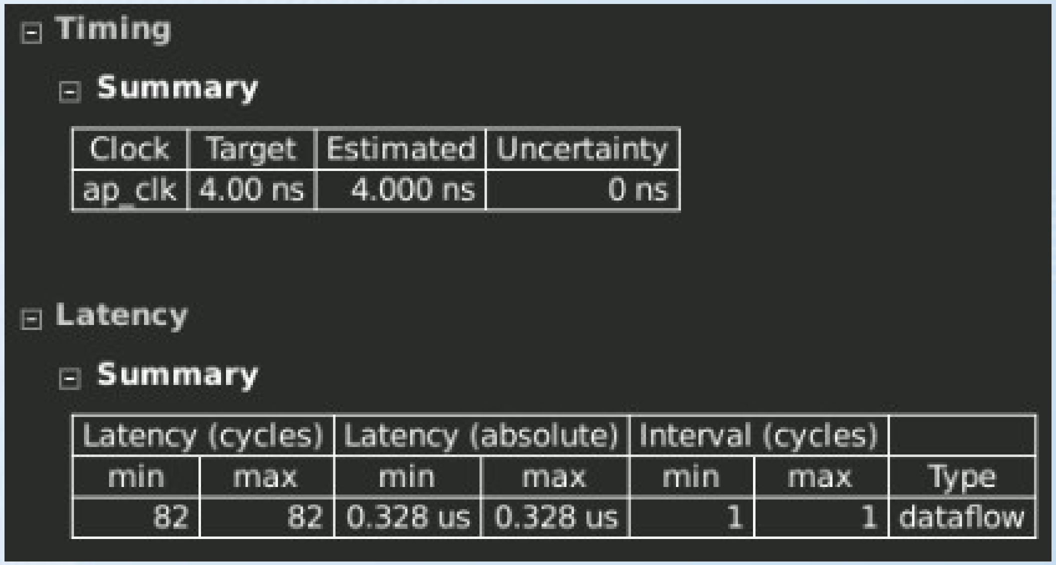
Beamformer kernel



Beamformer kernel

```
#include <ap_int.h>
#include <hls_stream.h>
#include <ap_axi_sdata.h>

#define DWIDTH 512

typedef ap_axiu<DWIDTH, 128, 0, 0> packet;

#define W 8

#define NCHANNEL 32
#define NELEMENT 32
#define NBEAM 32
```

- Easily re-scale
- Low latency
- Little overhead actually
- High development efficiency



Timing

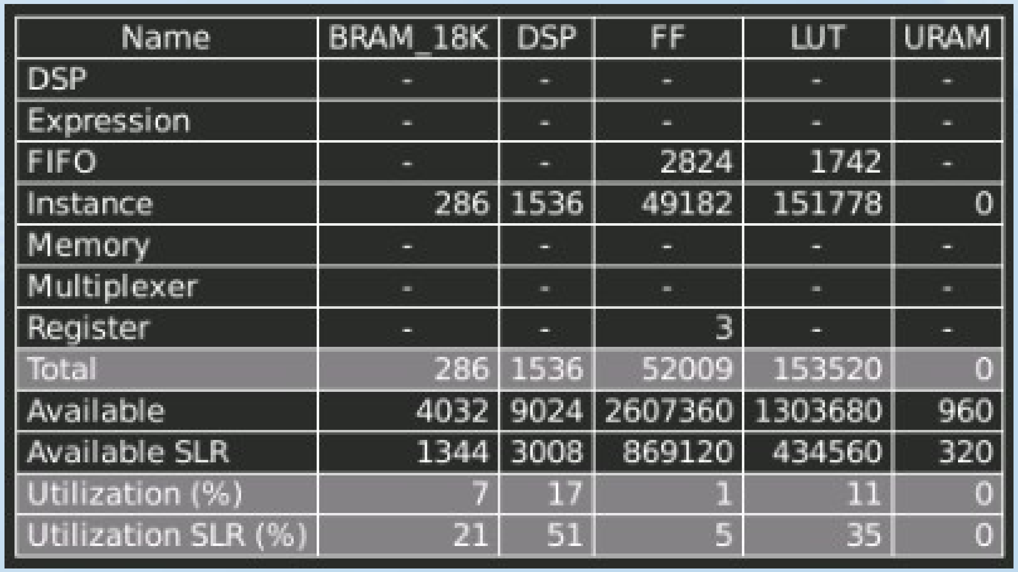
Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00 ns	4.000 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
82	82	0.328 us	0.328 us	1	1	dataflow



Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	2824	1742	-
Instance	286	1536	49182	151778	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	3	-	-
Total	286	1536	52009	153520	0
Available	4032	9024	2607360	1303680	960
Available SLR	1344	3008	869120	434560	320
Utilization (%)	7	17	1	11	0
Utilization SLR (%)	21	51	5	35	0

Summary and future plan

- We have already implemented a 32-element prototype FPGA beamformer using HLS.
- The unit tests were done.
- Still working on the integration test.
- We will scale the design to 128-element and see how many cards are needed finally. (Timing closure is the main limit)

Thanks for your attention!