Let’s do a Step by step Porting

Example: Tiled MxM

```
void matmulBlock_hw(T a[BS][BS],T b[BS][BS],T c[BS][BS]);
…
for (i_b=0; i_b<BSIZE; i_b++)
  for (j_b=0; j_b<BSIZE; j_b++)
    for (k_b=0; k_b<BSIZE; k_b++)
      matmulBlock_hw(AA[i_b][k_b], BB[k_b][j_b], CC[i_b][j_b]);
…
```

It is better to port it to pointer-based code,
as it happens in CUDA … GPU porting...

Note… SMP Porting of Tiled MxM

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(smp) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
            inout([CONST_BSIZE*CONST_BSIZE]c)
void matmulBlock_hw(T *a,T *b,T *c); …
...
    for (unsigned int i = 0; i < msize/BSIZE; i++) {
      for (unsigned int j = 0; j < msize/BSIZE; j++) {
         unsigned int const ci = j*b2size + i*BSIZE*msize;
         for (unsigned int k = 0; k < msize/BSIZE; k++) {
            unsigned int const ai = k*b2size + i*BSIZE*msize;
            unsigned int const bi = j*b2size + k*BSIZE*msize;
            matmulBlock_hw(&a[ai], &b[bi], &c[ci]);
    }}}
#pragma oss taskwait
```

- Target device by default is smp
- No copy deps since it is the same memory space

Taskwait directive!!!!

FPGA Porting of Tiled MxM w/ pointers

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
            inout([CONST_BSIZE*CONST_BSIZE]c) copy_deps num_instances(1)
void matmulBlock_hw(T *a,T *b,T *c); …
...
    for (unsigned int i = 0; i < msize/BSIZE; i++) {
      for (unsigned int j = 0; j < msize/BSIZE; j++) {
         unsigned int const ci = j*b2size + i*BSIZE*msize;
         for (unsigned int k = 0; k < msize/BSIZE; k++) {
            unsigned int const ai = k*b2size + i*BSIZE*msize;
            unsigned int const bi = j*b2size + k*BSIZE*msize;
            matmulBlock_hw(&a[ai], &b[bi], &c[ci]);
    }}}
#pragma oss taskwait
```

- Task device (fpga)
- num_instances
- copy_deps

Taskwait directive!!!!

What is matmulBlock_hw?

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
                inout([CONST_BSIZE*CONST_BSIZE]c) copy_deps num_instances(1)
void matmulBlock_hw(T *a,T *b,T *c);
…
void matmulBlock_hw(elem_t *a, elem_t *b, elem_t *c) {
   unsigned int i, j, k;
   for (i = 0; i < BSIZE; i++) {
      for (j = 0; j < BSIZE; j++) {
         elem_t sum = c[i*BSIZE + j];
         for (k = 0; k < BSIZE; k++)
               sum += a[i*BSIZE + k] * b[k*BSIZE + j];
         c[i*BSIZE + j] = sum;
}}}
```

Advise: Add labels to loops!

What is matmulBlock_hw?

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
            inout([CONST_BSIZE*CONST_BSIZE]c) copy_deps num_instances(1)
void matmulBlock_hw(T *a,T *b,T *c);
…
void matmulBlock_hw(elem_t *a, elem_t *b, elem_t *c) {
   unsigned int i, j, k;
loop_i_matmul:
   for (i = 0; i < BSIZE; i++) {
loop_j_matmul:
      for (j = 0; j < BSIZE; j++) {
         elem_t sum = c[i*BSIZE + j];
loop_k_matmul:
         for (k = 0; k < BSIZE; k++)
               sum += a[i*BSIZE + k] * b[k*BSIZE + j];
         c[i*BSIZE + j] = sum;
}   }}
```

Advise: Add labels to loops!
If you want to analyze the HLS synthesis w/ Vivado HLS

How do we optimize the kernel?

```
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
    inout([CONST_BSIZE*CONST_BSIZE]c) copy_deps num_instances(1)
void matmulBlock_hw(T *a,T *b,T *c);
…
```

```
void matmulBlock_hw(elem_t *a, elem_t *b, elem_t *c) {
   unsigned int i, j, k;
loop_i_matmul:
   for (i = 0; i < BSIZE; i++) {
loop_j_matmul:
      for (j = 0; j < BSIZE; j++) {
         elem_t sum = c[i*BSIZE + j];
loop_k_matmul:
         for (k = 0; k < BSIZE; k++)
               sum += a[i*BSIZE + k] * b[k*BSIZE + j];
         c[i*BSIZE + j] = sum;
      }
   }
```

Performance Analysis: 32x32 acc … low level...

compilation_program.ait.log

Automatic unroll

```
INFO: [HLS 214-376] automatically set the pipeline for Loop< loop_j_matmul> at matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:39:9
INFO: [HLS 214-376] automatically set the pipeline for Loop< unknown> at matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:123:5
INFO: [HLS 214-291] Loop 'loop_k_matmul' is marked as complete unroll implied by the pipeline pragma (matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:42:13)
```

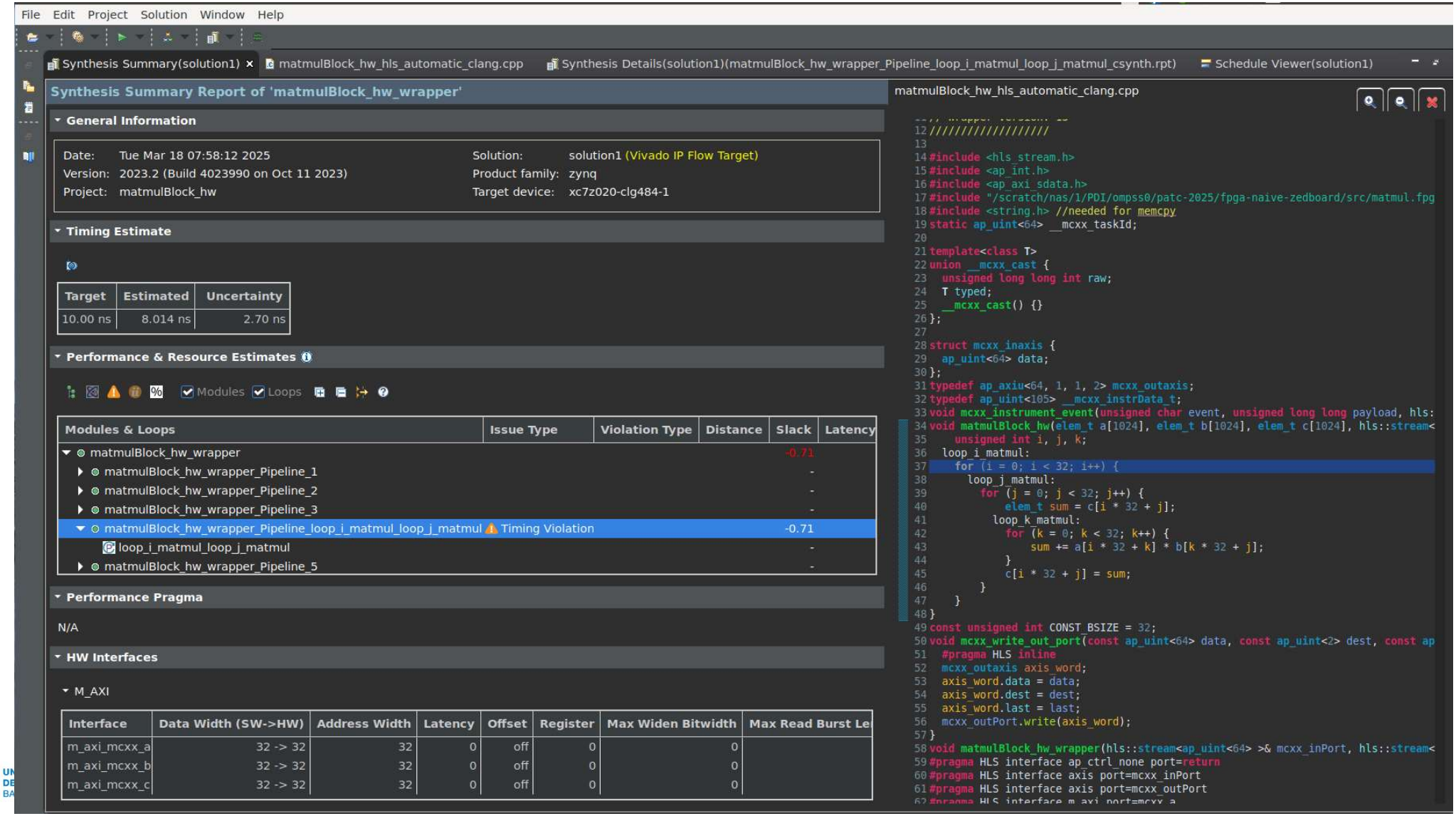
Automatic array partitioning!

```
INFO: [HLS 214-270] Inferring pragma 'array_partition type=cyclic factor=16 dim=1' for array 'matmulBlock_hw_wrapper(hls::stream<ap_uint<64>, 0>&, hls::stream<hls::axis<ap_uint<64>, 1ul, 1ul, 2ul, (unsigne
d char)56, false>, 0>&, float*, float*, float*, hls::stream<ap_uint<105>, 0>&)::a' due to pipeline pragma (matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:66:0)
INFO: [HLS 214-270] Inferring pragma 'array_partition type=cyclic factor=33 dim=1' for array 'matmulBlock_hw_wrapper(hls::stream<ap_uint<64>, 0>&, hls::stream<hls::axis<ap_uint<64>, 1ul, 1ul, 2ul, (unsigne
d char)56, false>, 0>&, float*, float*, float*, hls::stream<ap_uint<105>, 0>&)::b' due to pipeline pragma (matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:67:0)
```

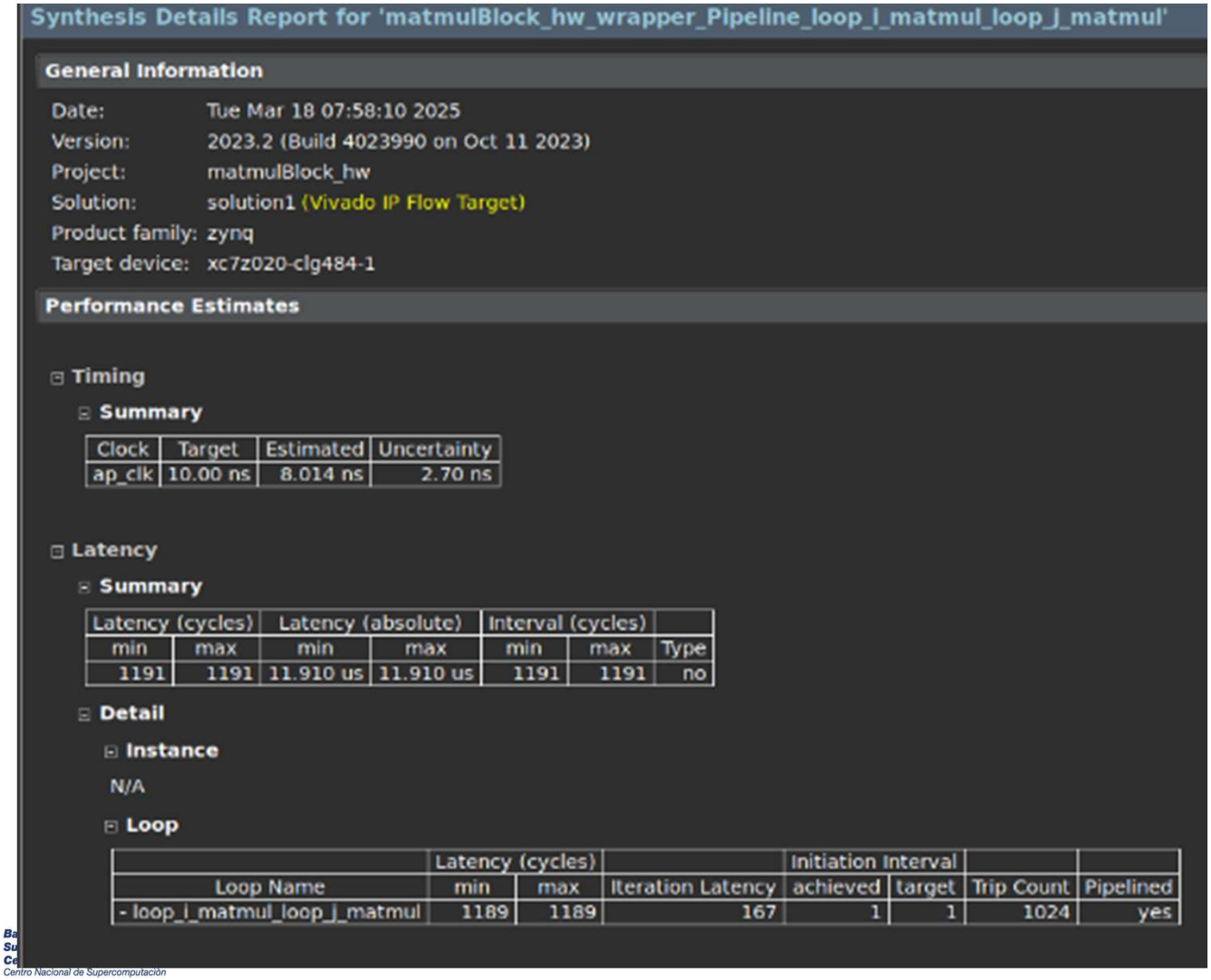
Automatic pipeline!

```
INFO: [HLS 200-42] -- Implementing module 'matmulBlock_hw_wrapper_Pipeline_loop_i_matmul_loop_j_matmul'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'loop_i_matmul_loop_j_matmul'.
INFO: [HLS 200-1470] Pipelining result : Target II = NA, Final II = 1, Depth = 167, loop 'loop_i_matmul_loop_j_matmul'
WARNING: [HLS 200-871] Estimated clock period (8.014 ns) exceeds the target (target clock period: 10.000 ns, clock uncertainty: 2.700 ns, effective delay budget: 7.300 ns).
Resolution: For help on HLS 200-871 see docs.xilinx.com/access/sources/dita/topic?Doc_Version=2023.2%20English&url=ug1448-hls-guidance&resourceid=200-871.html
WARNING: [HLS 200-1016] The critical path in module 'matmulBlock_hw_wrapper_Pipeline_loop_i_matmul_loop_j_matmul' consists of the following:
        'store' operation 0 bit ('j_write_ln35', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) of constant 0 on local variable 'j', ma
tmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119 [120]  (1.588 ns)
        'load' operation 6 bit ('j_load', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:39->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) on local variable 'j', matmulBlock_hw/matmulBl
ock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119 [128]  (0.000 ns)
        'icmp' operation 1 bit ('icmp_ln39', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:39->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) [132]  (1.825 ns)
        'select' operation 6 bit ('select_ln35', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) [133]  (1.188 ns)
        'add' operation 6 bit ('add_ln39', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:39->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) [2448]  (1.825 ns)
        'store' operation 0 bit ('j_write_ln35', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119) of variable 'add_ln39', matmulBlock_hw/
matmulBlock_hw_hls_automatic_clang.cpp:39->matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:119 on local variable 'j', matmulBlock_hw/matmulBlock_hw_hls_automatic_clang.cpp:35->matmulBlock_hw/matmulBl
ock_hw_hls_automatic_clang.cpp:119 [2451]  (1.588 ns)
```

Performance Analysis: 32x32 acc ...GUI Vitis HLS



Performance Analysis: 32x32 acc ...GUI (II/III)



Synthesis Details Report for 'matmulBlock_hw_wrapper_Pipeline_loop_i_matmul_loop_j_matmul'

General Information

Date: Tue Mar 18 07:58:10 2025
Version: 2023.2 (Build 4023990 on Oct 11 2023)
Project: matmulBlock_hw
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg484-1

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.014 ns	2.70 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1191	1191	11.910 us	11.910 us	1191	1191	no

Detail

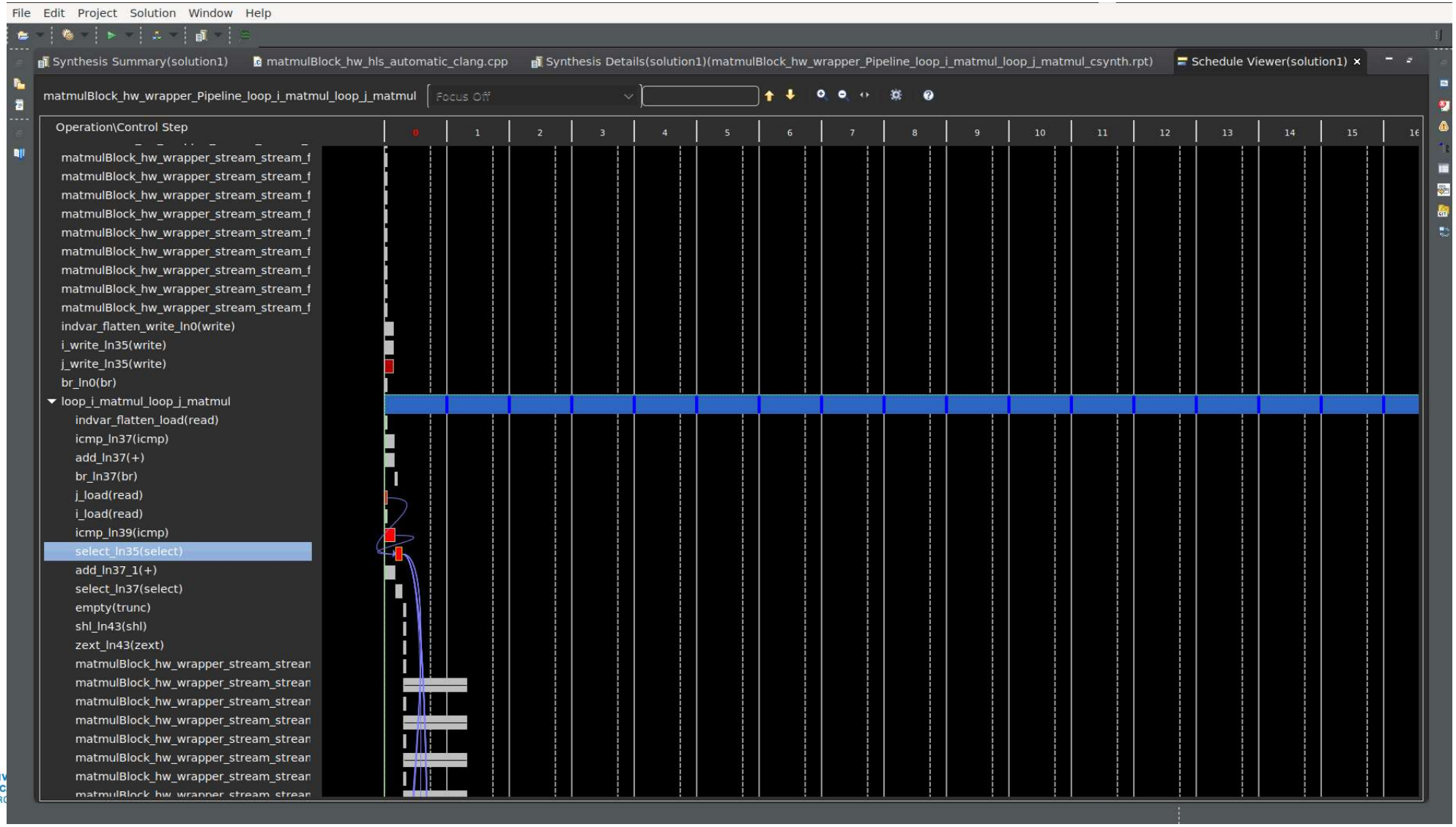
Instance

N/A

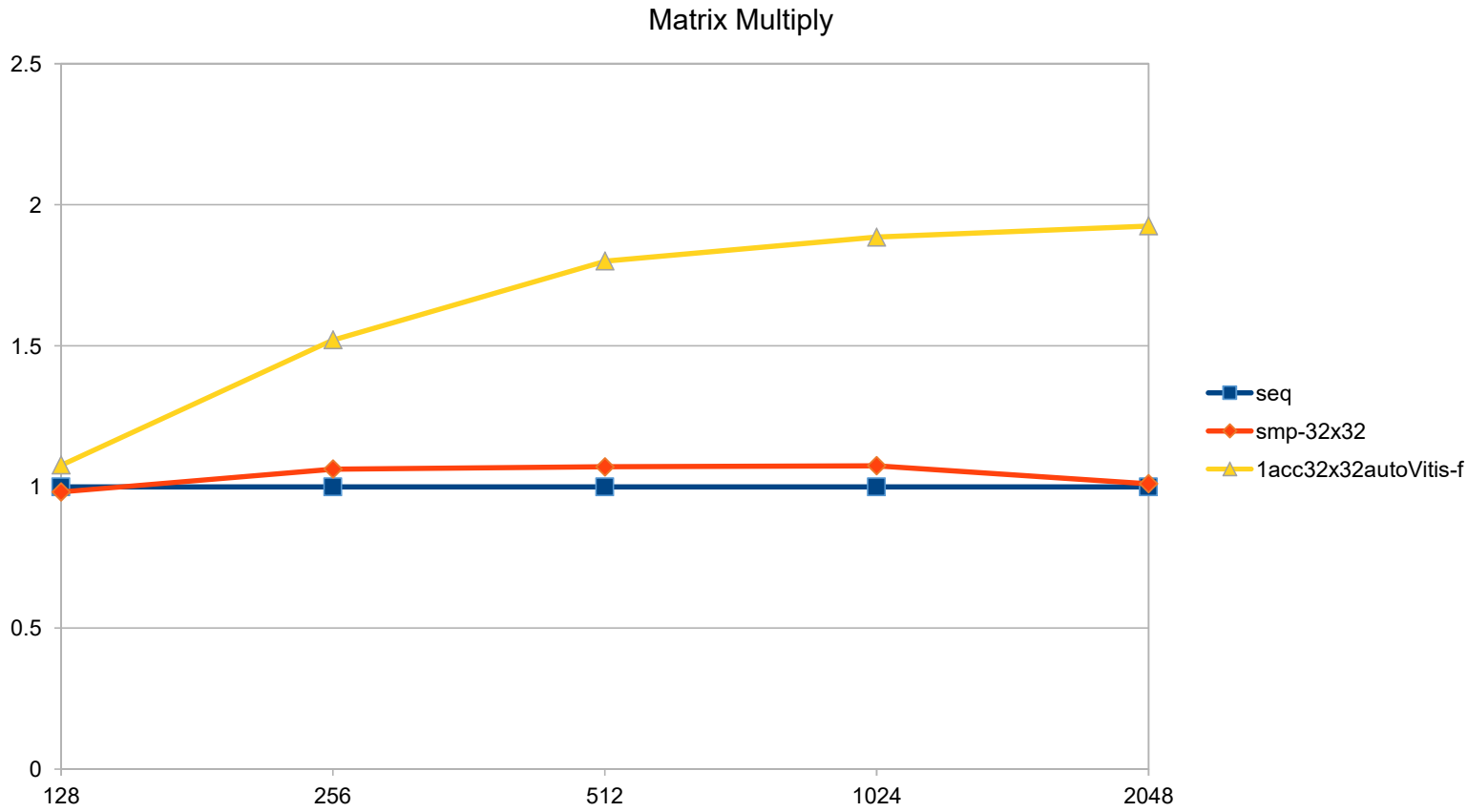
Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- loop_i_matmul_loop_j_matmul	1189	1189	167	1	1	1024	yes

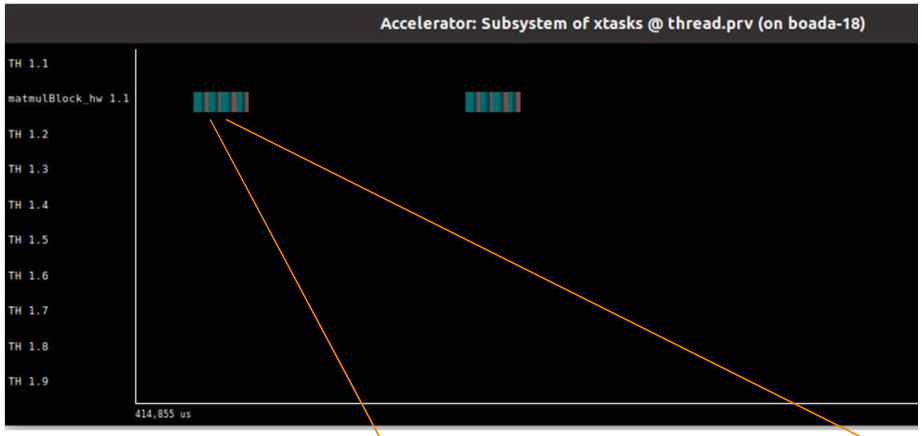
Performance Analysis: 32x32 acc ...GUI (III/III)

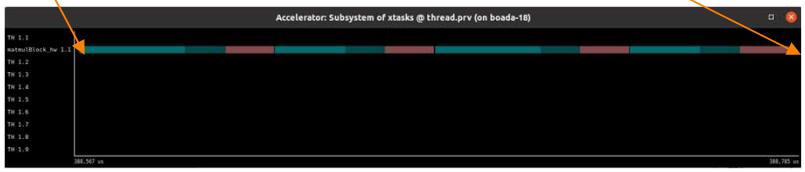


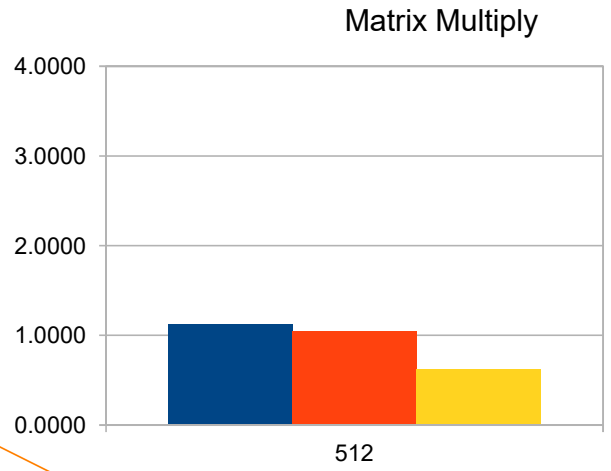
What is the performance of the Acc version?



Paraver Analysis: 32x32 – Vitis HLS opt.







New version

Increase to two accelerators

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
            inout([CONST_BSIZE*CONST_BSIZE]c) copy_deps num_instances(1)
void matmulBlock_hw(T *a,T *b,T *c); …
...
    for (unsigned int i = 0; i < msize/BSIZE; i++) {
      for (unsigned int j = 0; j < msize/BSIZE; j++) {
         unsigned int const ci = j*b2size + i*BSIZE*msize;
         for (unsigned int k = 0; k < msize/BSIZE; k++) {
            unsigned int const ai = k*b2size + i*BSIZE*msize;
            unsigned int const bi = j*b2size + k*BSIZE*msize;
            matmulBlock_hw(&a[ai], &b[bi], &c[ci]);
    }}}
#pragma oss taskwait
```

Question:
How can we obtain two accelerators?

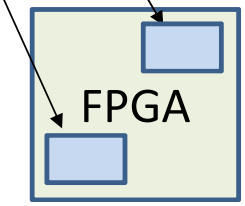
Increase to two accelerators

```
…
#define BSIZE 32
const unsigned int CONST_BSIZE = BSIZE;
#pragma oss task device(fpga) in([CONST_BSIZE*CONST_BSIZE]a, [CONST_BSIZE*CONST_BSIZE]b) \
                 inout([CONST_BSIZE*C                                    stances(1)
void matmulBlock_hw(T *a,T *b,T *c); …
...
    for (unsigned int i = 0; i < msize/BSIZE; i++) {
      for (unsigned int j = 0; j < msize/BSIZE; j++) {
         unsigned int const ci = j*b2size + i*BSIZE*msize;
         for (unsigned int k = 0; k < msize/BSIZE; k++) {
            unsigned int const ai = k*b2size + i*BSIZE*msize;
            unsigned int const bi = j*b2size + k*BSIZE*msize;
            matmulBlock_hw(&a[ai], &b[bi], &c[ci]);
    }}}
#pragma oss taskwait
```

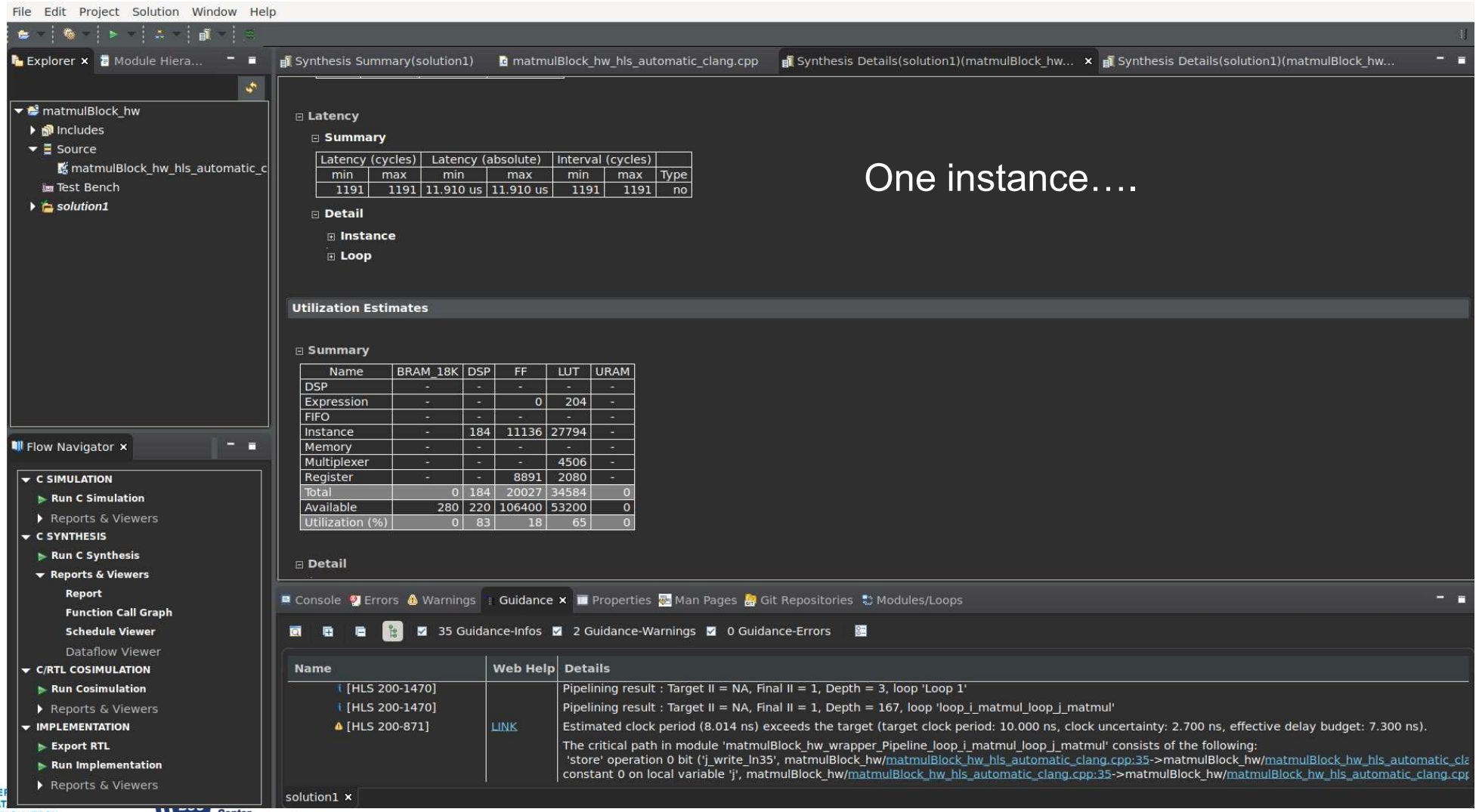
num_instances(2)

With AIT tool

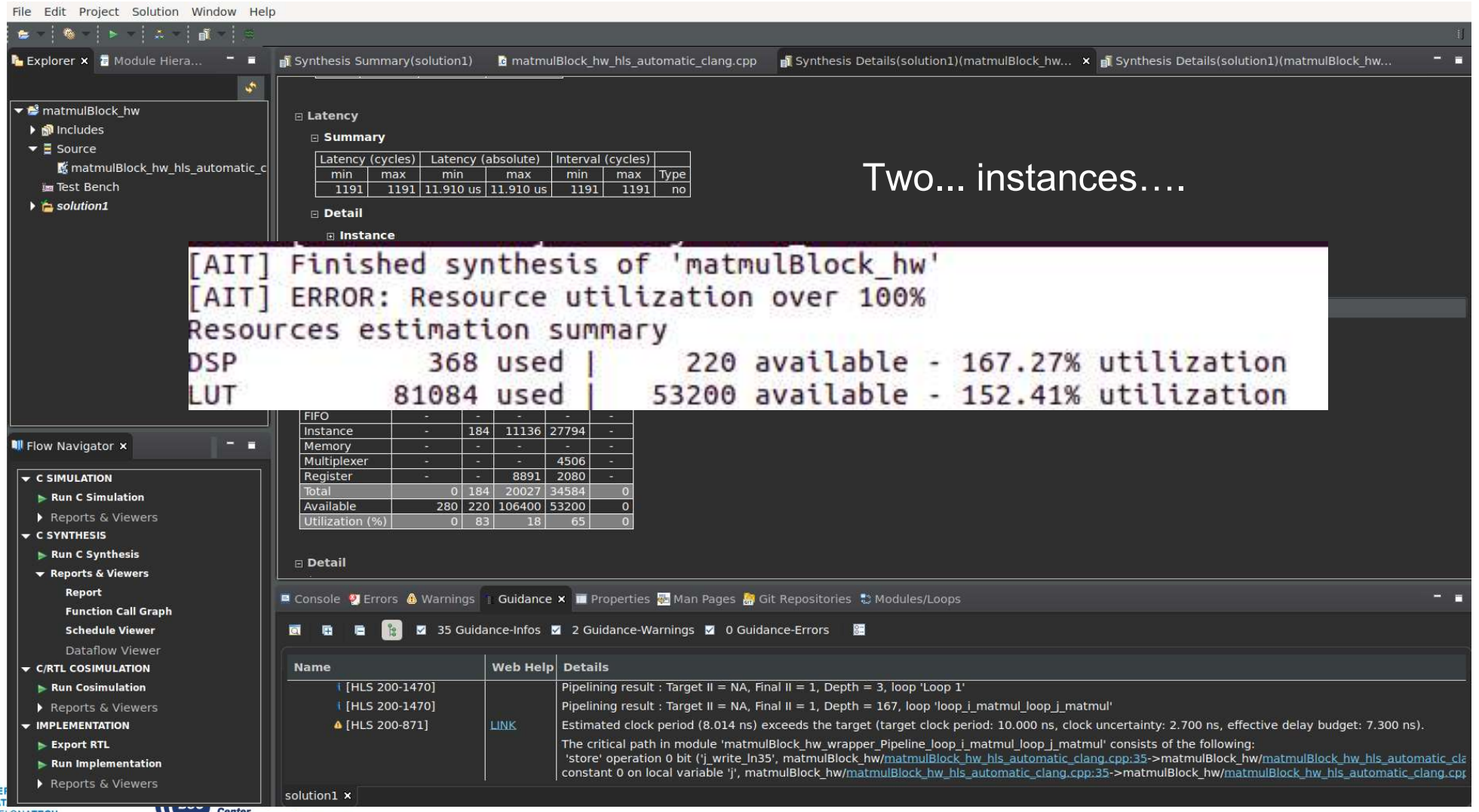
- Few extensions (num_instances)
- Triggers the bitstream generation automatically (Stub function generated)



Overall – Vitis... synthesis - auto



Overall – Vitis... synthesis - auto



Increase to two accelerators … II=2

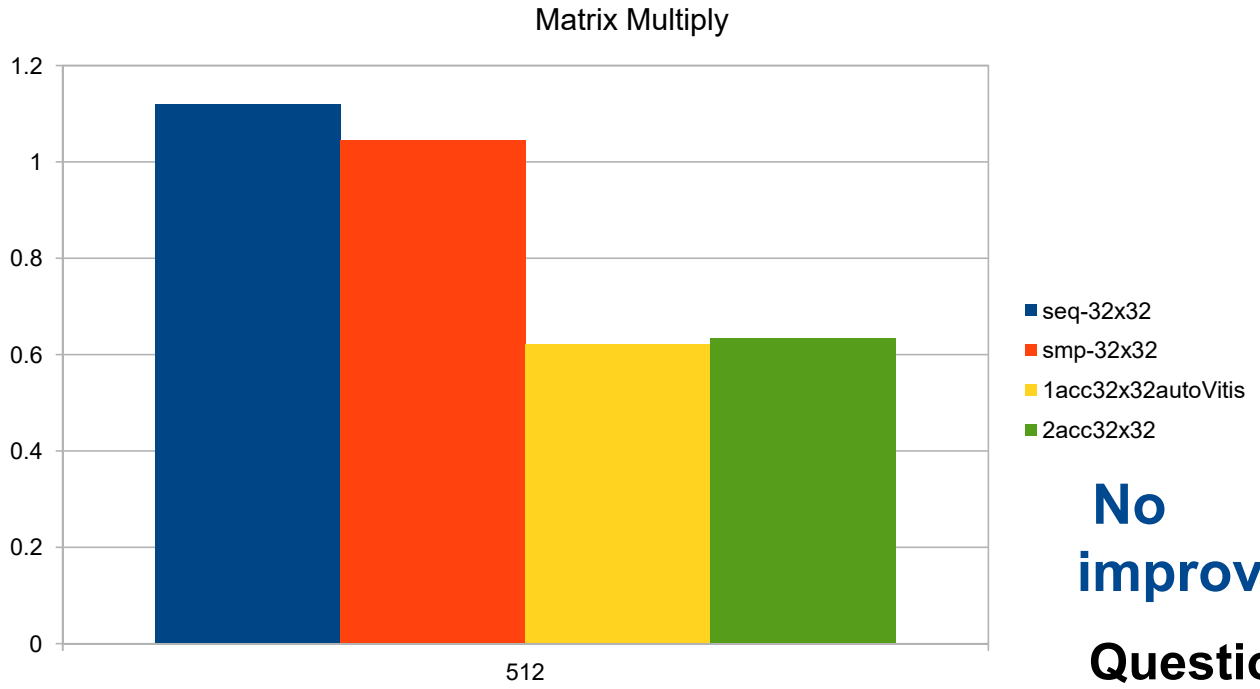
```
…
#define BSIZE 32

void matmulBlock_hw(elem_t *a, elem_t *b, elem_t *c) {
   unsigned int i, j, k;
   unsigned int FACTOR=BSIZE/2;
#pragma HLS ARRAY_PARTITION variable=a dim=1 cyclic factor=FACTOR
#pragma HLS ARRAY_PARTITION variable=b dim=1 block factor=FACTOR
loop_i_matmul:
   for (i = 0; i < BSIZE; i++) {
loop_j_matmul:
      for (j = 0; j < BSIZE; j++) {
         #pragma HLS PIPELINE II=2
         elem_t sum = c[i*BSIZE + j];
loop_k_matmul:
         for (k = 0; k < BSIZE; k++)
               sum += a[i*BSIZE + k] * b[k*BSIZE + j];
         c[i*BSIZE + j] = sum;
}   }}
```

Let's increase II =2 … to help to fit the acc…

What is the performance?
- Copies in parallel?
- Accelerator slower?

What is the performance of the Acc version?

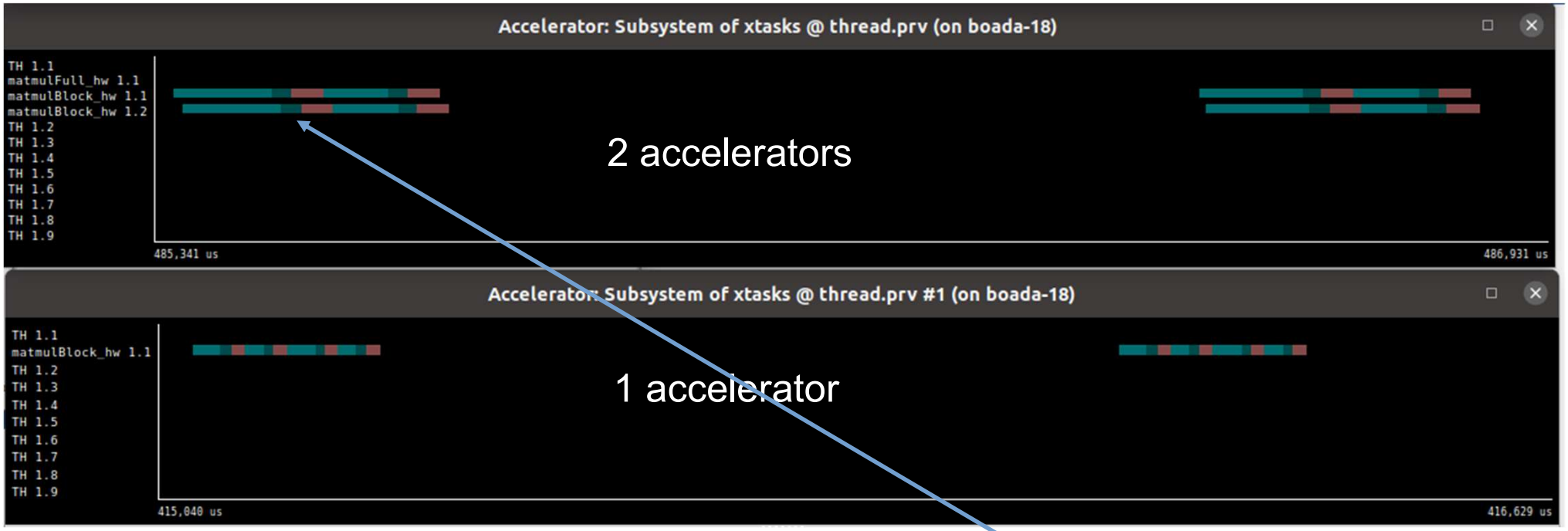


No improvement...

Question:
Why?

Paraver Analysis: 2acc 32x32 II=2

Paraver view with same duration and size...



- Slower Accelerator
II=2 vs II=1
- Memory conflicts
Accessing memory