WIDE PORT
FPGA compilation into OmpSs

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Compilation – Wide Port

Few details on the steps to compile

- Compiler: use clang or clang++ (C++)
  - » compiler -fompss-2 -o program program.c
  - » compiler -fompss-2 -fompss-fpga-instrumentation \
  - » **-fompss-fpga-memory-port-width 128** -o program program.c

- Details...
  - » AIT options:
    - --board=$(BOARD_NAME),--clock=100
    - --name=vivado_project_name,--dir=$(VIVADO_WORKSPACE)”

Wrapper Copies and Wide Port

```
    ap_wait();
  }
mcxx_instrument_event(78, __mcxx_taskId, mcxx_instr);
  if (mcxx_flags_0[4]) {
    memcpy(a, mcxx_a + mcxx_offset_0/sizeof(elem_t), (4096));
  }
  if (mcxx_flags_1[4]) {
    memcpy(b, mcxx_b + mcxx_offset_1/sizeof(elem_t), (4096));
  }
  if (mcxx_flags_2[4]) {
    memcpy(c, mcxx_c + mcxx_offset_2/sizeof(elem_t), (4096));
  }
```

- Memcopy is transformed
  - Wide copy
  - Then, split elements

```
if (mcxx_flags_0[4]) {
  for (int __i = 0; __i < ((4096) - 1)/sizeof(ap_uint<128>)+1; ++__i) {
  #pragma HLS pipeline II=1
    ap_uint<128> __tmpBuffer;
    __tmpBuffer = *(mcxx_memport + mcxx_offset_0/sizeof(ap_uint<128>) + __i);
    for (int __j=0; __j <(sizeof(ap_uint<128>)/4); __j++) {
      __mcxx_cast<elem_t> cast_tmp;
      cast_tmp.raw = __tmpBuffer((__j+1)*4*8-1,__j*4*8);
      a[__i*(sizeof(ap_uint<128>)/4)+__j] = cast_tmp.typed;
    }
  }
}
```

Increase granularity to 64x64

```
…
#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;

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;
}   }}
```

Question:

- **How can we increase the granularity in this kernel?**
- **Should we change the II again?**

Increase granularity to 64x64 + None

```
…
#define BSIZE 64

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++) {
         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;
}   }}
```

- Increase BSIZE to 64

Vitis HLS synthesis: 1 64x64 acc w/...

No directives?
CRASHES!!!!

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K      213 used |     280 available -  76.07% utilization
DSP           380 used |     220 available - 172.73% utilization
FF          45474 used |  106400 available -  42.74% utilization
LUT         93792 used |   53200 available -  176.3% utilization
```

64x64 + II=2

```
…
#define BSIZE 64

void matmulBlock_hw(elem_t *a, elem_t *b, elem_t *c) {
   unsigned int i, j, k;
   unsigned int FACTOR=BSIZE/2;



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;
}   }}
```

- Increase BSIZE to 64
- Advice Vitis with II=2

Vitis HLS synthesis: 1 64x64 acc w/...

No directives?
CRASHES!!!!

II=2 added +
no more
directives? OK

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K       213 used |      280 available -   76.07% utilization
DSP            380 used |      220 available - 172.73% utilization
FF           45474 used |   106400 available -   42.74% utilization
LUT          93792 used |    53200 available -   176.3% utilization
```

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K        94 used |      280 available -   33.57% utilization
DSP            219 used |      220 available -   99.55% utilization
FF           28036 used |   106400 available -   26.35% utilization
LUT          52737 used |    53200 available -   99.13% utilization
```

64x64+II=2+Partitioining

```
…
#define BSIZE 64

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;
}   }}
```

- Increase BSIZE to 64
- II=2
- ARRAY PARTITION

Vitis HLS synthesis: 1 64x64 acc w/...

No directives?
CRASH!!!!

II=2 added +
no more
directives? OK

II=2 +
array_partitioning
and pipeline
directives? OK

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K       213 used |      280 available -   76.07% utilization
DSP            380 used |      220 available - 172.73% utilization
FF           45474 used |   106400 available -   42.74% utilization
LUT          93792 used |    53200 available -   176.3% utilization
```

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K        94 used |      280 available -   33.57% utilization
DSP            219 used |      220 available -   99.55% utilization
FF           28036 used |   106400 available -   26.35% utilization
LUT          52737 used |    53200 available -   99.13% utilization
```

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K       148 used |      280 available -   52.86% utilization
DSP            160 used |      220 available -   72.73% utilization
FF           27471 used |   106400 available -   25.82% utilization
LUT          34492 used |    53200 available -   64.83% utilization
```

Vitis HLS synthesis: 1 64x64 acc w/...

No directives?
CRASH!!!!

II=2 added +
no more
directives? OK

II=2 +
array_partitioning
and pipeline
directives? OK

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K       213 used |      280 available -   76.07% utilization
DSP            380 used |      220 available - 172.73% utilization
FF           45474 used |   106400 available -   42.74% utilization
LUT          93792 used |    53200 available -   176.3% utilization
```

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K        94 used |      280 available -   33.57% utilization
DSP            219 used |      220 available -   99.55% utilization
FF           28036 used |   106400 available -   26.35% utilization
LUT          52737 used |    53200 available -   99.13% utilization
```

```
Finished synthesis of 'matmulBlock_hw'
Resources estimation summary
BRAM_18K       148 used |      280 available -   52.86% utilization
DSP            160 used |      220 available -   72.73% utilization
FF           27471 used |   106400 available -   25.82% utilization
LUT          34492 used |    53200 available -   64.83% utilization
```

Question:

- **What is the best?**

Vitis HLS synthesis: almost-auto vs opt

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.837 ns	2.70 ns

II=2 + no more directives?

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
31297	31297	0.313 ms	0.313 ms	31297	31297	no

Detail

Instance

No collapsing of loops!

Instance	
grp_matmulBlock_hw_Pipeline_loop_j_matmul_fu_745	matmulBlock_hw

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	785	-
FIFO	-	-	-	-	-
Instance	-	219	22407	46576	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	462	-
Register	-	-	2103	-	-
Total	0	219	24510	47823	0
Available	280	220	106400	53200	0
Utilization (%)	0	99	23	89	0

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.909 ns	2.70 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
8518	8518	85.180 us	85.180 us	8518	8518	no

Detail

Instance

Loop

II=2 + array_partitioning and pipeline directives?

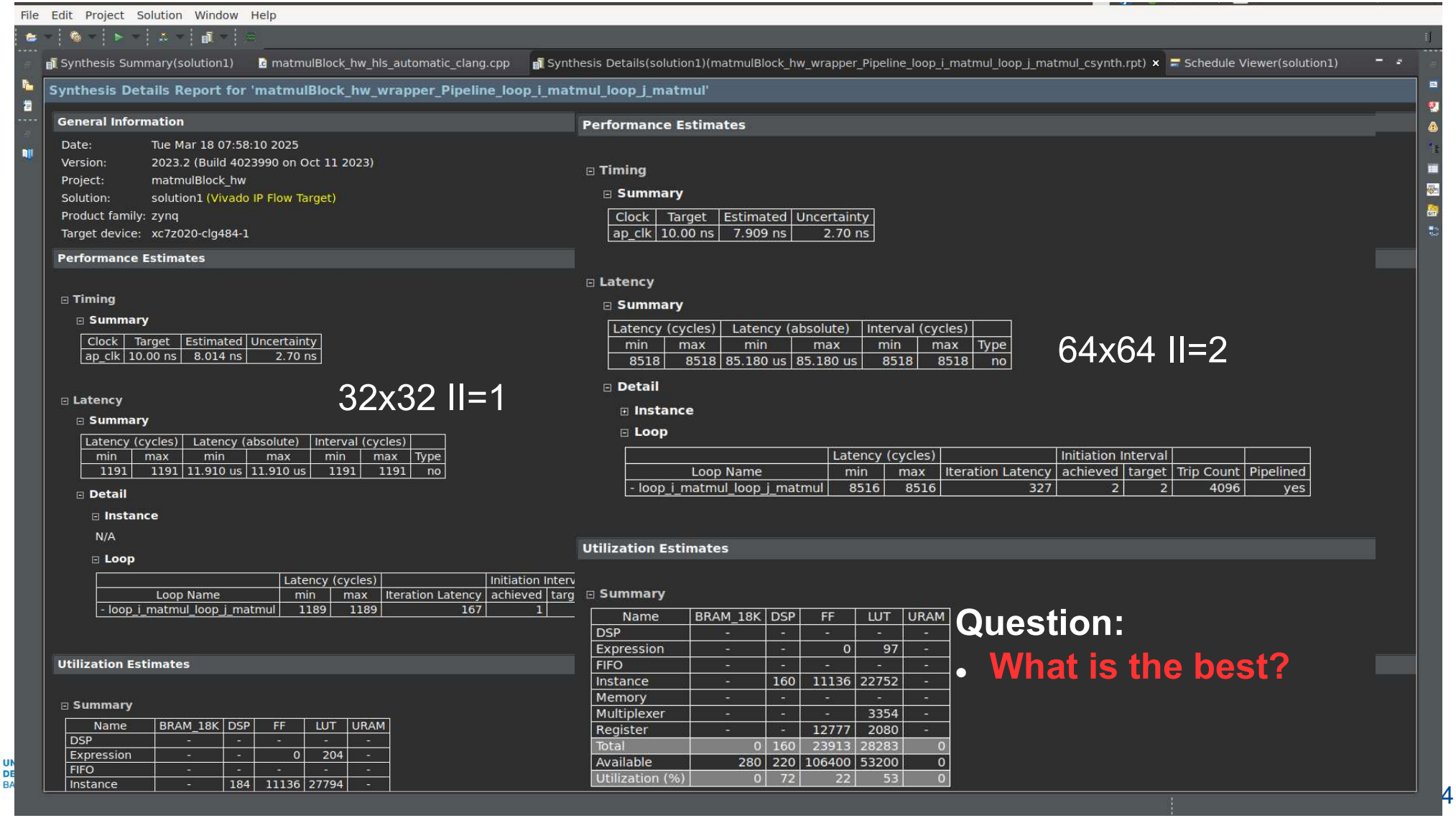
	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- loop_i_matmul_loop_j_matmul	8516	8516	327	2	2	4096	yes

Utilization Estimates

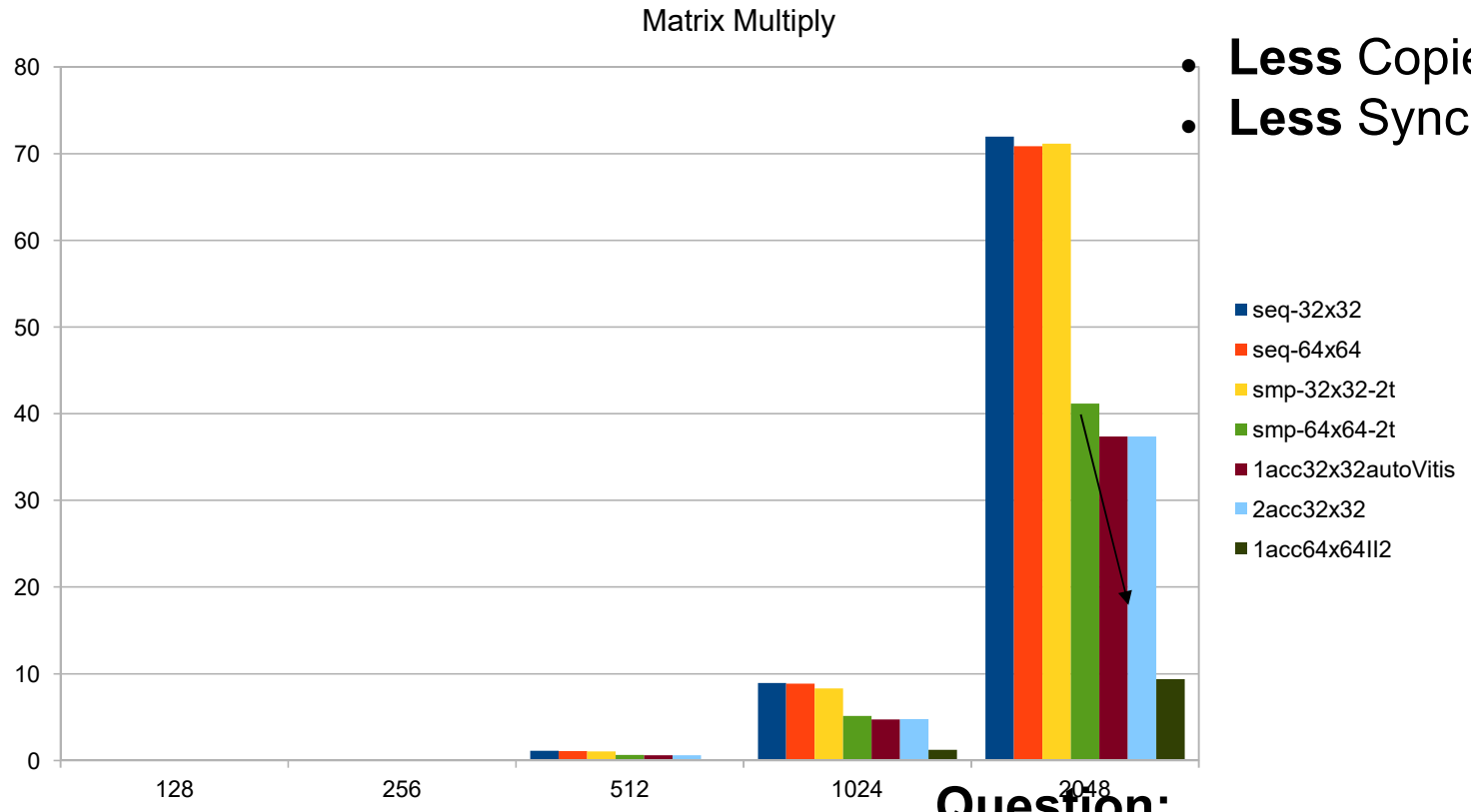
Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	97	-
FIFO	-	-	-	-	-
Instance	-	160	11136	22752	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	3354	-
Register	-	-	12777	2080	-
Total	0	160	23913	28283	0
Available	280	220	106400	53200	0
Utilization (%)	0	72	22	53	0

Comparison 32x32 II=1 vs 64x64 II=2



What is the performance of the Acc version?

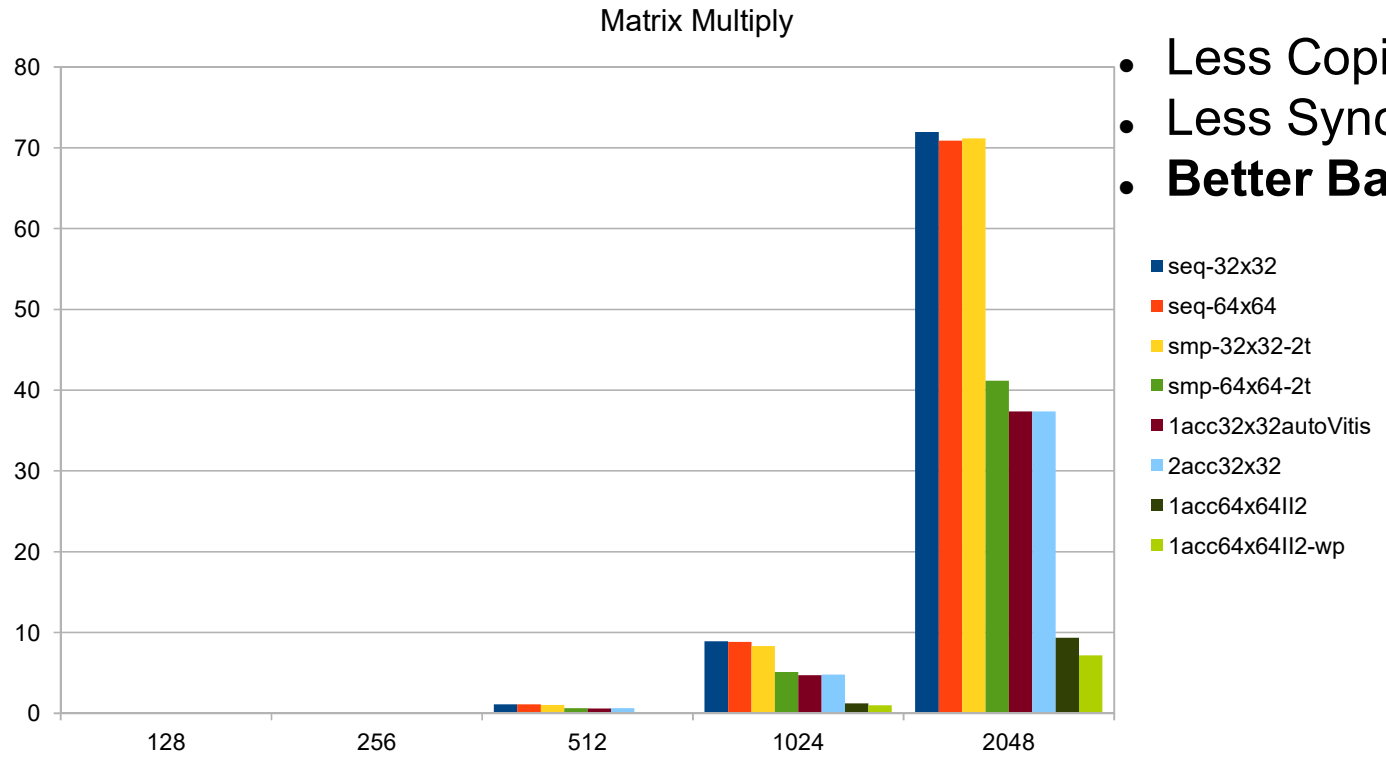


- **Less** Copies
- **Less** Synchronizations

Question:

- **Can we improve it again? WP?**

What is the performance of the Acc version?



- Less Copies
- Less Synchronizations
- **Better Bandwidth!**

Question:

- **Can we improve it again?** Blocking, you will work on that as a project!