Discussion #8:

HLS Optimizations on FPGA

Daniel Tan, Jason Lau, and Jason Cong

Outline

- Tutorial: HLS Optimization on FPGA for Lab 4
  - How to “profile” the kernel
  - How to boost performance
  - How to avoid pitfalls / tool bugs
  - How to reduce high-level synthesis time
  - How to optimize resource usage

How to “Profile” the Kernel

- **Where is the bottleneck for the Lab 4 kernel?**
  - **You already knew that in your Lab 3 :-)**
  - **But where it the bottleneck after some optimizations?**
  - **RTL is cycle accurate**

Loop Name	Latency (cycles) min	Latency (cycles) max	Iteration Latency	Initiation Interval achieved	Initiation Interval target	Trip Count	Pipelined
- main_loop_tile_h_main_loop_tile_w	246701161496	246701161496	61675290374	-	-	4	no
+ main_loop_i	61668316928	61668316928	240891863	-	-	256	no
++ set_bias_L	12544	12544	2	1	1	12544	yes
++ conv_L	240844809	240844809	13	3	1	80281600	yes
++ relu_L	25088	25088	3	2	1	12544	yes
++ maxpool_L	9413	9413	9	3	1	3136	yes

How to “Profile” the Kernel

- How do I know which loop is it?

```
// Convolution
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q)
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
      }
    }
  }
}
```

```
+-------------------------------------+--------------+--------------+
|                                     |      Latency (cycles)       |
|              Loop Name              |      min     |      max     |
+-------------------------------------+--------------+--------------+
|- main_loop_tile_h_main_loop_tile_w  | 246701161496| 246701161496|
| + main_loop_i                       |  61668316928|  61668316928|
|  ++ set_bias_L                      |        12544|        12544|
|  ++ main_loop_i.2                   |    240844809|    240844809|
|  ++ relu_L                          |        25088|        25088|
|  ++ maxpool_L                       |         9413|         9413|
+-------------------------------------+--------------+--------------+
```

How to “Profile” the Kernel

- How do I know which loop is it?

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q)
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
      }
    }
  }
}
```

```
+-------------------------------------+--------------+--------------+-
|                                     |      Latency (cycles)       |
|              Loop Name              |      min     |      max     |
+-------------------------------------+--------------+--------------+-
|- main_loop_tile_h_main_loop_tile_w  | 246701161496| 246701161496|
| + main_loop_i                       |  61668316928|  61668316928|
|  ++ set_bias_L                      |        12544|        12544|
|  ++ conv_L                          |    240844809|    240844809|
|  ++ relu_L                          |        25088|        25088|
|  ++ maxpool_L                       |         9413|         9413|
+-------------------------------------+--------------+--------------+-
```

How to “Profile” the Kernel

- How do I know which loop is it?
- What is _L?

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q)
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
      }
    }
  }
}
```

Loop Name	Latency (cycles) min	Latency (cycles) max
- main_loop_tile_h_main_loop_tile_w	246701161496	246701161496
+ main_loop_i	61668316928	61668316928
++ set_bias_L	12544	12544
++ conv_L ?	240844809	240844809
++ relu_L	25088	25088
++ maxpool_L	9413	9413

How to “Profile” the Kernel

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q)
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
      }
    }
  }
```

automatically →

flatten

```
conv_L:
for (int iter = 0; iter < kNum*kTileH*kTileW*kKernel*kKernel; iter++) {
  int j = iter / (kTileH*kTileW*kKernel*kKernel);
  int h = iter / (kTileW*kKernel*kKernel) % kTileH;
  int w = iter / (kKernel*kKernel) % kTileW;
  int p = iter / kKernel % kKernel;
  int q = iter % kKernel;
  C[h][w] += weight[i][j][p][q] *
             input[j][h + p][w + q];
}
```

Loop Name	Latency (cycles) min	Latency (cycles) max	Iteration Latency	Initiation Interval achieved	Initiation Interval target	Trip Count	Pipelined
- main_loop_tile_h_main_loop_tile_w	246701161496	246701161496	61675290374	-	-	4	no
+ main_loop_i	61668316928	61668316928	240891863	-	-	256	no
++ set_bias_L	12544	12544	2	1	1	12544	yes
++ conv_L	240844809	240844809	13	3	1	80281600	yes
++ relu_L	25088	25088	3	2	1	12544	yes
++ maxpool_L	9413	9413	9	3	1	3136	yes

How to “Profile” the Kernel

- **Where is my loop?**
  - **Unrolled into individual operations, no loops anymore.**

```
// ReLU
relu:
for (int h = 0; h < kTileH; ++h) {
#pragma HLS unroll
  for (int w = 0; w < kTileW; ++w) {
#pragma HLS unroll
    if (C[h][w] < 0) C[h][w] = 0;
  }
}
```

Loop Name	Latency (cycles) min	Latency (cycles) max	Iteration Latency
- main_loop_tile_h_main_loop_tile_w	246701161496	246701161496	61675290374
+ main_loop_i	61668316928	61668316928	240891863
++ set_bias_L	12544	12544	2
++ conv_L	240844809	240844809	13
++ maxpool_L	9413	9413	9

No relu

Outline

- Tutorial: HLS Optimization on FPGA for Lab 4
  - How to “profile” the kernel
  - How to boost performance
  - How to reduce high-level synthesis time
  - How to optimize resource usage
  - How to avoid pitfalls / tool bugs

How to Boost the Performance

- `#pragma HLS unroll factor=<n>`
- `#pragma HLS pipeline II=<interval>`
- `#pragma HLS array_partition variable=<name> <type> \`
  `factor=<int>  dim=<int>`

- **Might be useful in this lab but not required for A++:**
- `#pragma HLS array_reshape variable=<name> <type> dim=<int>`
- `#pragma HLS dataflow   // insignificant effect`

How to Boost the Performance

- `#pragma HLS unroll factor=<n>`

only for demo
not a way to achieve
the best performance

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q) {
#pragma HLS unroll   // if unrolled
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
} } } } }
```

Equivalent

=

How is this useful?

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

How to Boost the Performance

- `#pragma HLS unroll factor=<n>`

only for demo
not a way to achieve
the best performance

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        for (int q = 0; q < kKernel; ++q) {
// if without unroll
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
} } } } }
```

Compare

Sequential Control

Scheduling

Possible to have:

1. Rearrange operations order.
2. Multiple operations at the same cycle.
3. Data reuse.

Is this good enough?

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

How to Boost the Performance

- **`#pragma HLS array_partition`**

only for demo
not a way to achieve
the best performance

```
  static input_t   input [kNum][kTileH+kKernel-1][kTileW+kKernel-1];
#pragma HLS array_partition variable=input  dim=3 cyclic factor=5
  static weight_t  weight[kNum][kNum][kKernel][kKernel];
#pragma HLS array_partition variable=weight dim=4 complete
```

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

How to Boost the Performance

- `#pragma HLS array_partition`

only for demo
not a way to achieve
the best performance

```
  static input_t   input [kNum][kTileH+kKernel-1][kTileW+kKernel-1];
#pragma HLS array_partition variable=input  dim=3 cyclic factor=5
  static weight_t  weight[kNum][kNum][kKernel][kKernel];
#pragma HLS array_partition variable=weight dim=4 complete
```

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

Wow!
All in parallel!

How to Boost the Performance

only for demo
not a way to achieve
the best performance

Step 1: Read all `weight` and `input`

Step 2: Compute all multiplication

Step 3: C[h][w] += result[0]

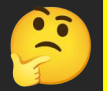
Step 4: C[h][w] += result[1]

Step 5: C[h][w] += result[2]

Step 6: C[h][w] += result[3]

Step 7: C[h][w] += result[4]

Partition does not help



```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

How to Boost the Performance

- `#pragma HLS pipeline II=<interval>`

only for demo
not a way to achieve
the best performance

Step 1: Read all `weight` and `input`
Step 2: Compute all multiplication
Step 3: C[h][w+1] += result[0]
Step 4: C[h][w+1] += result[1]
Step 5: C[h][w+1] += result[2]
Step 6: C[h][w+1] += result[3]
Step 7: C[h][w+1] += result[4]

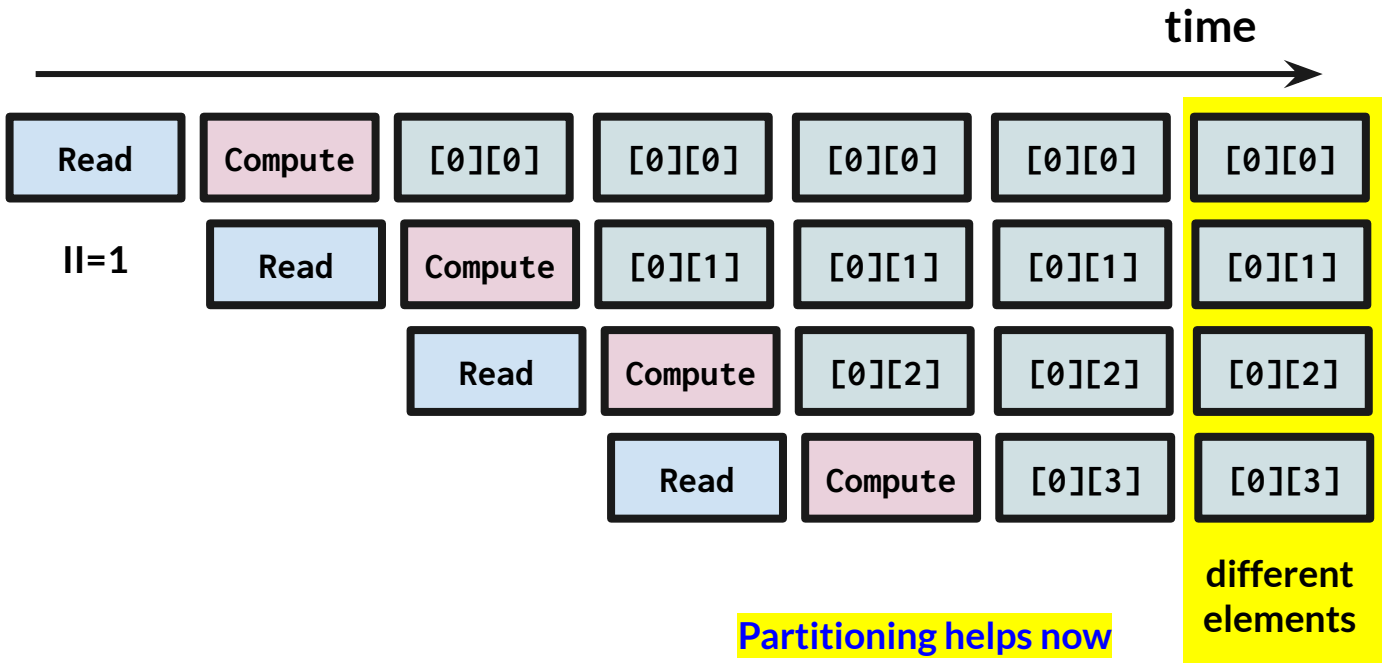
Step 1: Read all `weight` and `input`
Step 2: Compute all multiplication
Step 3: C[h][w] += result[0]
Step 4: C[h][w] += result[1]
Step 5: C[h][w] += result[2]
Step 6: C[h][w] += result[3]
Step 7: C[h][w] += result[4]

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int w = 0; w < kTileW; ++w) {
      for (int p = 0; p < kKernel; ++p) {
        C[h][w] += weight[i][j][p][0] *
                   input[j][h + p][w + 0];
        C[h][w] += weight[i][j][p][1] *
                   input[j][h + p][w + 1];
        C[h][w] += weight[i][j][p][2] *
                   input[j][h + p][w + 2];
        C[h][w] += weight[i][j][p][3] *
                   input[j][h + p][w + 3];
        C[h][w] += weight[i][j][p][4] *
                   input[j][h + p][w + 4];
} } } }
```

pipeline the w loop

How to Boost the Performance

- `#pragma HLS pipeline II=1`



How to Boost the Performance

- `#pragma HLS pipeline II=<interval>`

only for demo
not a way to achieve
the best performance

```
// Convolution
conv:
for (int j = 0; j < kNum; ++j) {
  for (int h = 0; h < kTileH; ++h) {
    for (int p = 0; p < kKernel; ++p) {
      for (int w = 0; w < kTileW; ++w) {
#pragma HLS pipeline
        for (int q = 0; q < kKernel; ++q) {
#pragma HLS unroll
          C[h][w] += weight[i][j][p][q] *
                     input[j][h + p][w + q];
} } } } }
```

Pipeline can be applied to any code block (function / loop)

Remember to put inside the code block

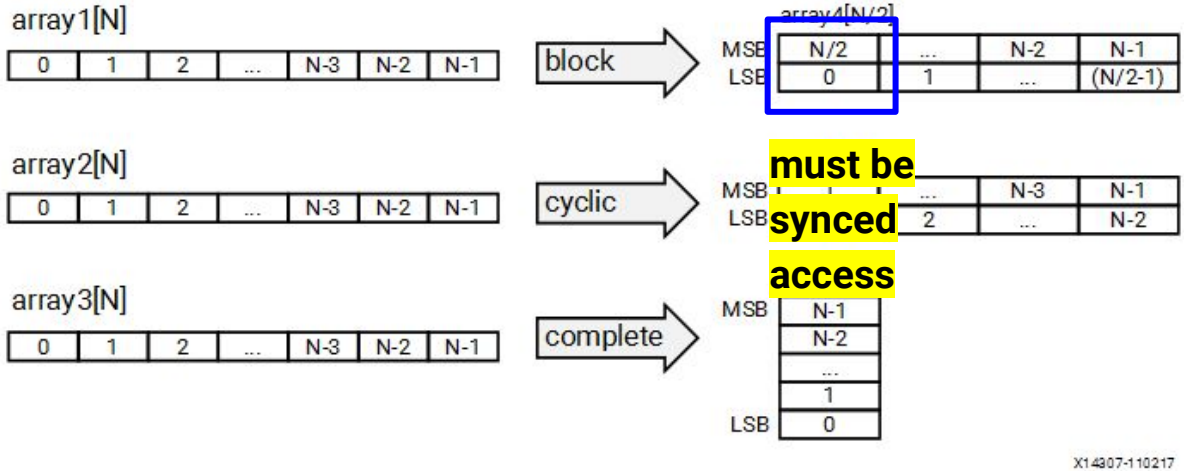
How to Boost the Performance

- `#pragma HLS unroll factor=<n>`
- `#pragma HLS pipeline II=<interval>`
- `#pragma HLS array_partition variable=<name> <type> \`
  `factor=<int>  dim=<int>`

- **Might be useful in this lab but not required for A++:**
- `#pragma HLS array_reshape variable=<name> <type> dim=<int>`
- `#pragma HLS dataflow   // insignificant effect`

How to Boost the Performance

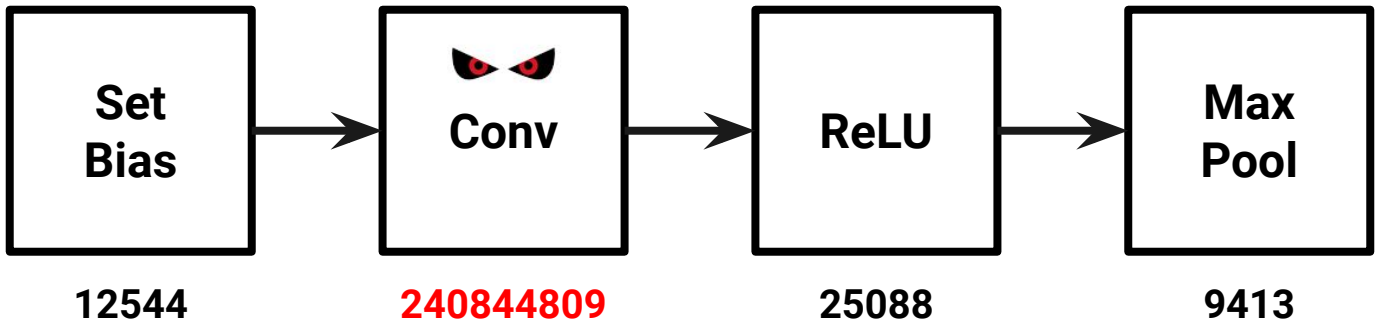
- **#pragma HLS array_reshape variable=<name> <type> dim=<int>**
  - Really similar to #pragma HLS array_partition



https://www.xilinx.com/html_docs/xilinx2017_4/sdaccel_doc/mrl1504034361747.html

How to Boost the Performance

- `#pragma HLS dataflow`
  - Sounds promising?
  - No...

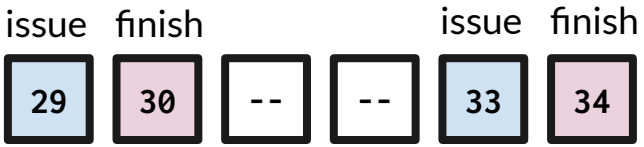


How to “Debug” the Performance

- **Advanced Tips: How to debug pipeline?** (Why the hell is my II=188 instead of 1?)
  - `_x/cnn.hw.*/CnnKernel/vivado_hls.log`

```
INFO: [SCHED 204-61] Pipelining loop 'maxpool_L'.
WARNING: [SCHED 204-69] Unable to schedule 'load' operation ('C_V_load_3', /home/lau/workspace/20210228.starter-kit/lab5/cnn-krnl.cpp:79)
 on array 'C_V' due to limited memory ports. Please consider using a memory core with more ports or partitioning the array 'C_V'.
INFO: [SCHED 204-61] Pipelining result : Target II = 1, Final II = 3, Depth = 9.

+---------------------------------------+-------------+-------------+-------------+-----------+-----------+----------+-----------+
|                                       |      Latency (cycles)     |  Iteration  |  Initiation Interval  |   Trip   |           |
|              Loop Name                |     min     |     max     |   Latency   |  achieved |   target  |   Count  | Pipelined |
+---------------------------------------+-------------+-------------+-------------+-----------+-----------+----------+-----------+
|- main_loop_tile_h_main_loop_tile_w    | 246701161496| 246701161496| 61675290374|          -|          -|         4|    no     |
| + main_loop_i                         |  61668316928|  61668316928|   240891863|          -|          -|       256|    no     |
|  ++ set_bias_L                        |        12544|        12544|           2|          1|          1|     12544|    yes    |
|  ++ conv_L                            |    240844809|    240844809|          13|          3|          1|  80281600|    yes    |
|  ++ relu_L                            |        25088|        25088|           3|          2|          1|     12544|    yes    |
|  ++ maxpool_L                         |         9413|         9413|           9|          3|          1|      3136|    yes    |
+---------------------------------------+-------------+-------------+-------------+-----------+-----------+----------+-----------+
```

“Debug” Performance



- **Advanced Tips: How to debug scheduling?** (Why the hell is `maxpool` II=3 instead of 2?)
  - **`_x/cnn.hw.*/CnnKernel/CnnKernel/solution/.autopilot/db/CnnKernel_YourCode.verbose.sched.rpt`**

```
// Max pooling
maxpool:
for (int h = 0; h < kTileH/2; ++h) {
  for (int w = 0; w < kTileW/2; ++w) {
    output[i][h][w] = max(
        max(C[h * 2][w * 2    ], C[h * 2 + 1][w * 2    ]),
        max(C[h * 2][w * 2 + 1], C[h * 2 + 1][w * 2 + 1]));
  }
}
```

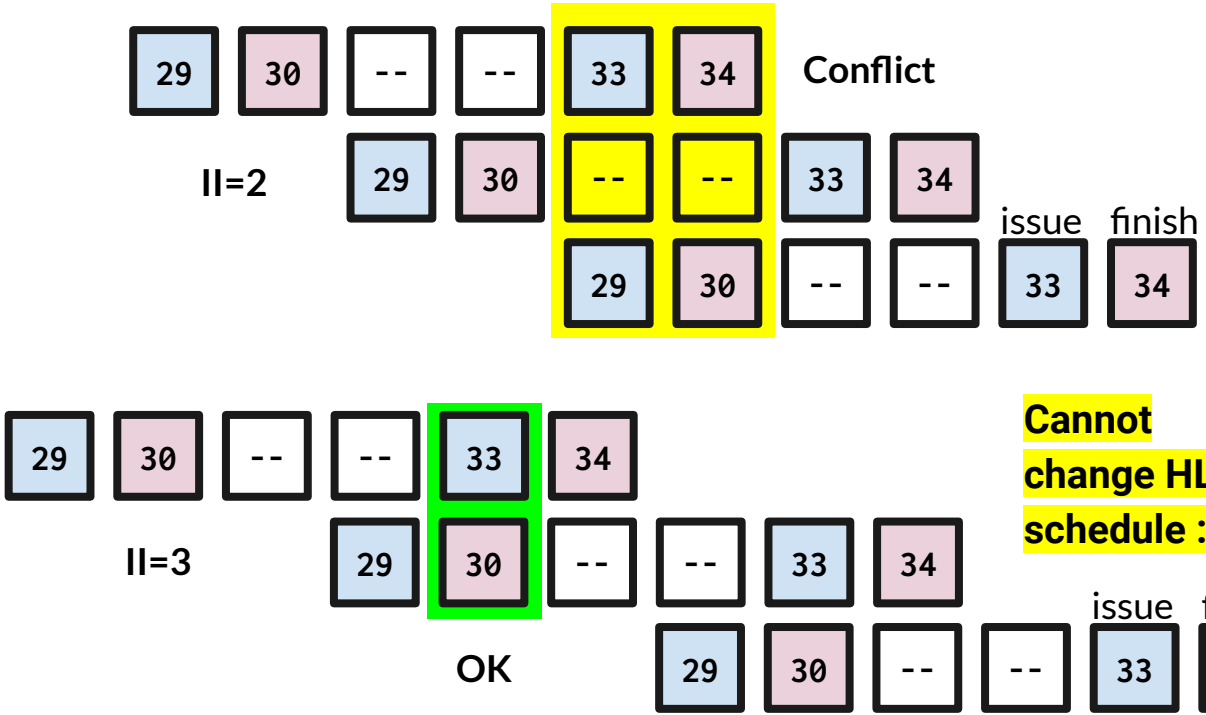
```
ST_29  Operation 308 [2/2] (1.15ns)   --->   "%C_V_load_3 = load i17* %C_V_addr_5, align 4" [/home/lau/workspace/20
210228.starter-kit/lab5/cnn-krnl.cpp:79]   --->   Operation 308 'load' 'C_V_load_3' <Predicate = (!icmp_ln75)> <Dela
y = 1.15> <Core = "RAM">   --->   Core 37 'RAM' <Latency = 1> <II = 1> <Delay = 1.15> <Storage> <Opcode : 'load' 'st
ore'> <Ports = 2> <Width = 8> <Depth
ST_29 : Operation 309 [2/2] (1.15ns)   --->   "%C_V_load_4 = load i17* %C_V_addr_6, align 4" [/home/lau/workspace/20
210228.starter-kit/lab5/cnn-krnl.cpp
y = 1.15> <Core = "RAM">   --->   C
ore'> <Ports = 2> <Width = 8> <Depth

ST_30  Operation 310 [1/2] (1.15ns)   --->   "%C_V_load_3 = load i17* %C_V_addr_5, align 4" [/home/lau/workspace/20
210228.starter-kit/lab5/cnn-krnl.cpp:79]   --->   Operation 310 'load' 'C_V_load_3' <Predicate = (!icmp_ln75)> <Dela
y = 1.15> <Core = "RAM">   --->   Core 37 'RAM' <Latency = 1> <II = 1> <Delay = 1.15> <Storage> <Opcode : 'load' 'st
ore'> <Ports = 2> <Width = 8> <Depth = 802816> <RAM>
ST_30 : Operation 311 [1/2] (1.15ns)   --->   "%C_V_load_4 = load i17* %C_V_addr_6, align 4" [/home/lau/workspace/20
210228.starter-kit/lab5/cnn-krnl.cpp:79]   --->   Operation 311 'load' 'C_V_load_4' <Predicate = (!icmp_ln75)> <Dela
y = 1.15> <Core = "RAM">   --->   Core 37 'RAM' <Latency = 1> <II = 1> <Delay = 1.15> <Storage> <Opcode : 'load' 'st
ore'> <Ports = 2> <Width = 8>

ST_33  Operation 336 [2/2] (1.
starter-kit/lab
ration 336 'load' 'C_V_load_6'
cy = 1> <II = 1> <Delay = 1.15>
ST_33 : Operation 339 [2/2] (1.15ns)   --->   "%C_V_load_7 = load i17* %C_V_addr_8, align 4" [/home/lau/workspace/202
10228.starter-kit/lab5/lib/cnn-krnl.h:32->/home/lau/workspace/20210228.starter-kit/lab5/cnn-krnl.cpp:79]   --->   Ope
ration 339 'load' 'C_V_load_7' <Predicate = (!icmp_ln75)> <Delay = 1.15> <Core = "RAM">   --->   Core 37 'RAM' <Laten
cy = 1> <II = 1> <Delay = 1.15>

ST_34  Operation 340 [1/2] (1.
starter-kit/lab
ration 340 'load' 'C_V_load_6'
cy = 1> <II = 1> <Delay = 1.15>
ST_34 : Operation 341 [1/2] (1.15ns)   --->   "%C_V_load_7 = load i17* %C_V_addr_8, align 4" [/home/lau/workspace/202
10228.starter-kit/lab5/lib/cnn-krnl.h:32->/home/lau/workspace/20210228.starter-kit/lab5/cnn-krnl.cpp:79]   --->   Ope
ration 341 'load' 'C_V_load_7' <Predicate = (!icmp_ln75)> <Delay = 1.15> <Core = "RAM">   --->   Core 37 'RAM' <Laten
cy = 1> <II = 1> <Delay = 1.15> <Storage> <Opcode : 'load' 'store'> <Ports = 2> <Width = 8> <Depth = 802816> <RAM>
```

Search for the line in the log

first step of an operation

second step of an operation

How to “Debug” the Performance



Common Pitfalls for HLS

- Double check the pragma syntax
  - Invalid syntax will simply be ignored! Takes a lot of time to debug.
  - Common error: `#pragma array_partitioning variable=weight_l complete`
- It's been running for 15 mins. Should I wait?
  - No. Unless you know what you are doing.
  - You can achieve the A range, even the A+ range within 15 mins.
  - The very last step to achieve A++ may take hours.

Common Pitfalls for HLS

- **Why is my kernel even slower after the first optimization?**
  - **It is expected. Continue working if the estimated time is < 800 sec.**
  - **Vitis will do some basic optimizations if you do not.**
  - **You have to start anyways. :-)**
- **Why don't we partition everything and unroll everything?**
  - **You will end up hundreds of years of HLS time and a huge design lol.**
  - **Perform unrolling / partitioning only if necessary.**
- **More coming. Sure, we are going to have a lot!**

HLS Optimization on FPGA for Lab 4

- Are the examples sufficient for me to achieve A?
  - No. You can probably get E
- What should I use to achieve A?
  - You have a lot of loops & high degree of parallelism!
  - `#pragma HLS unroll factor=<n>`
  - `#pragma HLS pipeline II=<interval>`
  - `#pragma HLS array_partition variable=<name> <type> factor=<int>  dim=<int>`
  - & some simple loop transformations
    - One student achieved 285 GOPS with only 9 lines of changes!

HLS Optimization on FPGA for Lab 4

- **What do I need to do for A+ / A++?**
  - **Reduce high-level synthesis (compilation) time**
  - **Optimize resource usage**

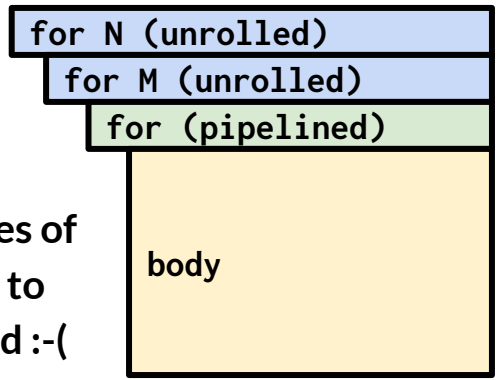
How to Reduce HLS Time?

- Partition the arrays only when necessary.
  - One array partitioned to 100 arrays should work fine.
  - You might need to wait for a while for ~500x.
  - It does not work for ~1000x.
- Copy from the large BRAM buffer to small BRAM/registers buffers
  - Partition the local buffers
- Find data reuse opportunities
  - Reuse registers (a lot of connections), not BRAM (has only two ports)

How to Reduce HLS Time?

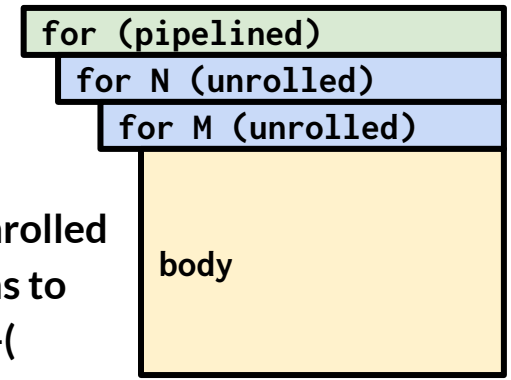
- **Unroll first or pipeline first?**
- **Neither. Functions are your friends.**
  - **You can pipeline a function and call it in a pipelined loop!**
  - **Avoid duplicating designs.**

NxM pipelines to be scheduled individually



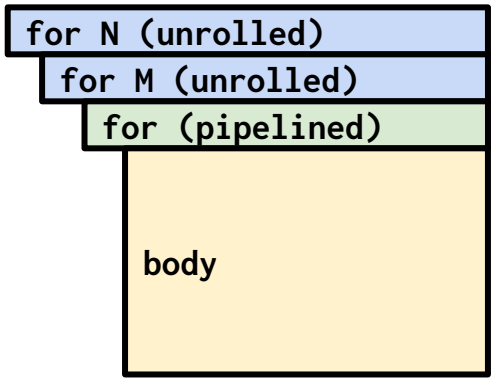
a lot of copies of the pipeline to be generated :-(

NxM operations to be scheduled

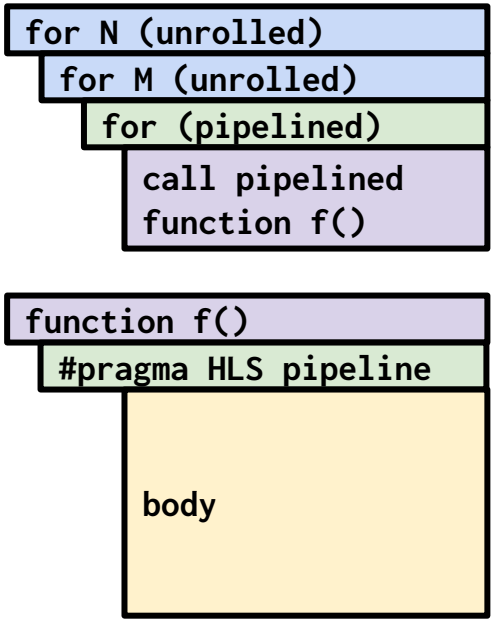


a lot of unrolled operations to pipeline :-(

How to Reduce HLS Time?



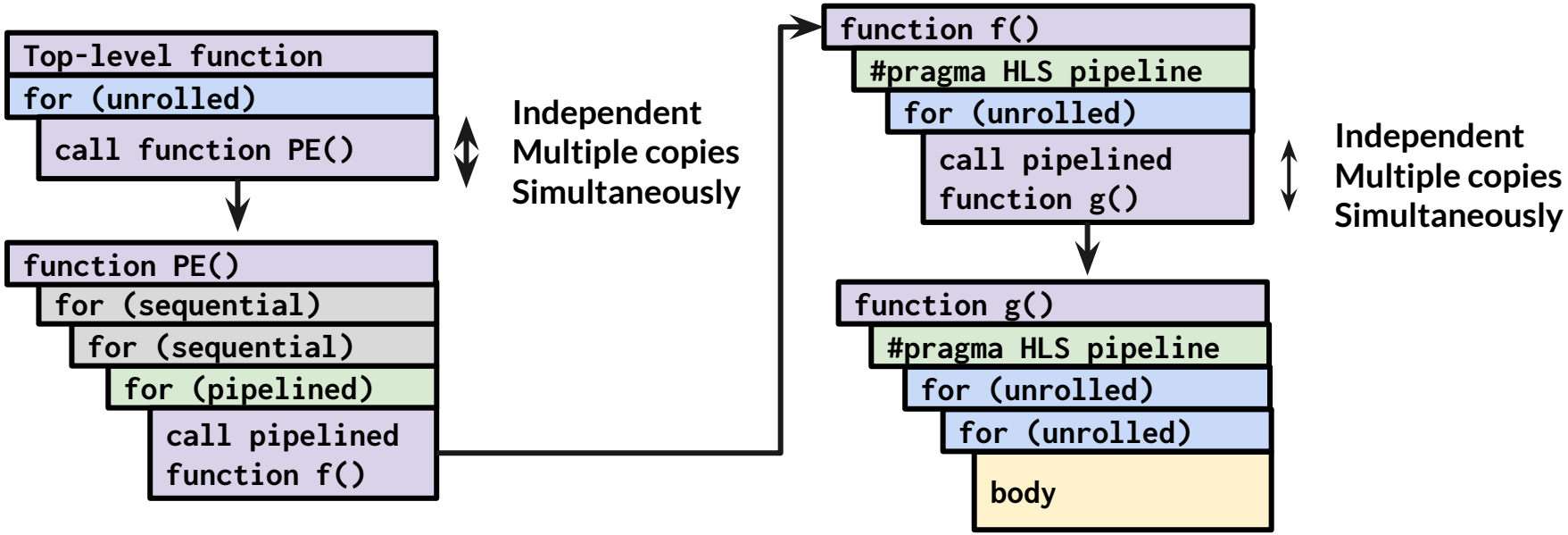
You should instead



Easier to perform. Only one instruction.

How to Reduce HLS Time?

A real-world example (from a real HLS design)



How to Reduce Resource Usage?

- Make the H and W tilling smaller

```
#define kTileH   (28)
#define kTileW   (56)
```

  - Might also reduce HLS time.
  - Reduce the global memory access complexity
  - Trade off between:
    - Less resource, faster HLS
    - or, less control / bandwidth overhead

How to Reduce Resource Usage?

- “Profiling”
  - Put into functions
  - `_x/cnn.hw.*/CnnKernel/CnnKernel/solution/syn/report/YourFunc.rpt`
- Reuse registers.
  - Using a register in concurrent computations, we increase the parallelism to data partitioning ratio!
  - Data partitioning is expensive.

```
================================================================
== Vivado HLS Report for 'ConvW'
================================================================
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      0|        0|     1314|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |        -|      -|        0|     3825|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|        9|    -|
|Register             |        -|      -|     3162|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |        0|      0|     3162|     5148|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        0|      0|    ~0   |        1|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        0|      0|    ~0   |    ~0   |    0|
+---------------------+---------+-------+---------+---------+-----+
```

Q&A