Time sharing pyramids for Hardware Acceleration

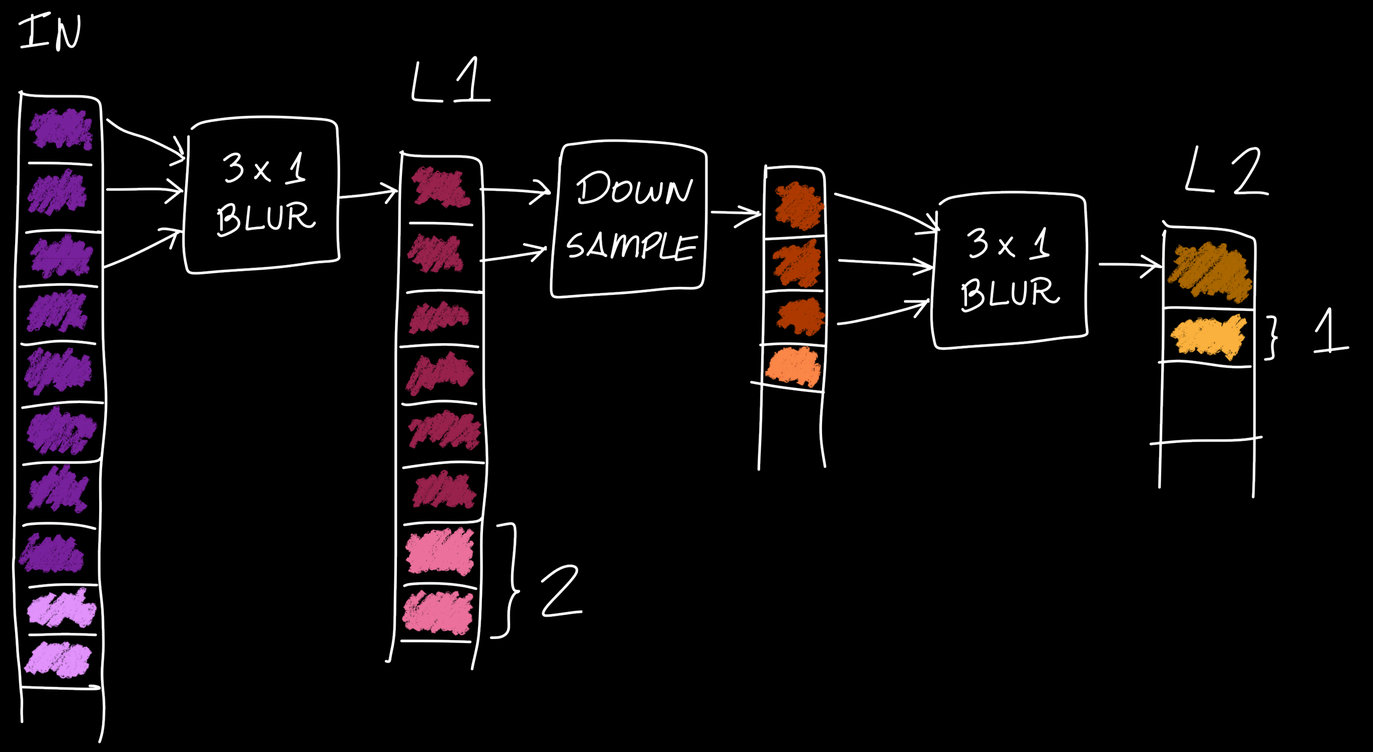
Breno de Mello Dal Bianco

CS 348K

Pyramid imbalance

1D Gaussian Pyramid example

- Imbalance across levels
- At steady state, each L2 pixel requires two L1 pixels to be produced
- One compute unit per level means low utilization
- **Goal: share compute units to increase utilization**



Hardware acceleration challenges

- Handwritten hardware description language (HDL) code akin to handwritten assembly
  - Hard to write, debug, extend…
- There are tools available which generates HDL from C++ code
- **Constrain: use a High Level Synthesis (HLS) tool**

HLS system example

- Input is C++ loop(s):

```
for (int i = 0; i < 10; i++)
  temp[i] = in[i];
for (int i = 0; i < 8; i++)
  out[i] = temp[i] + temp[i+1] + temp[i+2];
```

- Output is full HDL design which implements loop(s)
  - Handles internal memories, dependencies, pipelining…
- Inputs stream in, outputs stream out

HLS limitations

Not all loops are created equal

- More natural to write

```
for (int i = 0; i < 10; i++)
  temp[i] = in[i];
for (int i = 0; i < 8; i++)
  out[i] = temp[i] + temp[i+1] + temp[i+2];
```

- More efficient memory access (requires smaller `temp` buffer)

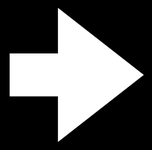
```
for (int i = 0; i < 8; i++) {
  temp[i] = in[i];
  if (i > 2) out[i] = temp[i] + temp[i+1] + temp[i+2];
}
```

Clockwork compiler

- Can convert natural loop to efficient loop
- Outputs c++ file ready for Vivado HLS
- **Goal: extend clockwork compiler to produce pyramid design with time sharing**

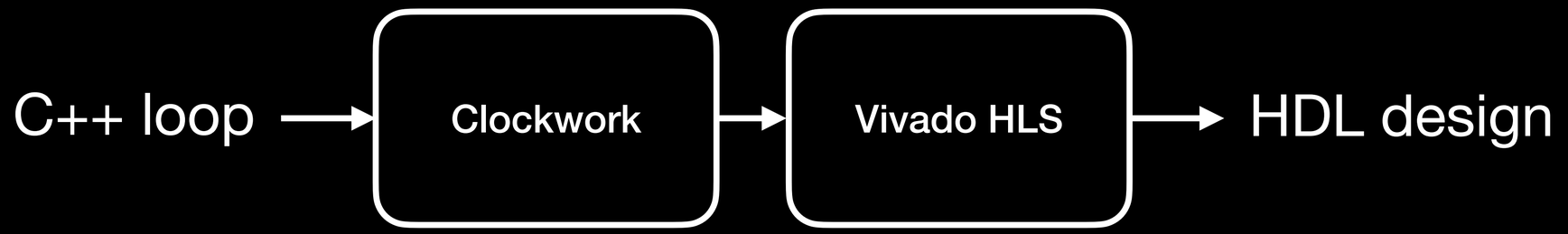
Clockwork compiler

```
for (int c3 = 0; c3 <= 133; c3 += 1)
  get_input(0, c3);
for (int c3 = 0; c3 <= 131; c3 += 1)
  compute_level_1(0, c3);
for (int c3 = 0; c3 <= 65; c3 += 1)
  downsample1(0, c3);
for (int c3 = 0; c3 <= 63; c3 += 1)
  compute_level_2(0, c3);
for (int c3 = 0; c3 <= 31; c3 += 1)
  downsample2(0, c3);
for (int c3 = 0; c3 <= 29; c3 += 1)
  compute_level_3(0, c3);
```



```
for (int c0 = 0; c0 <= 133; c0 += 1) {
  get_input(0, c0);
  if (c0 >= 2) {
    compute_level_1(0, c0 - 2);
    if (c0 % 2 == 0) {
      downsample1(0, (c0 / 2) - 1);
      if (c0 >= 6) {
        compute_level_2(0, (c0 / 2) - 3);
        if ((c0 + 2) % 4 == 0) {
          downsample2(0, (c0 - 6) / 4);
          if (c0 >= 14)
            compute_level_3(0, (c0 - 14) / 4);
        }
      }
    }
  }
}
```

System overview



Naive vs Unrolled solution

Pipeline diagram comparison

- Baseline loop

```
for (int i = 0; i < 8; i++) {
  A();
  if (i % 2 == 0) B();
}
```

- Naive (baseline + resource constrain)
- Unrolled loop

```
for (int i = 0; i < 4; i++) {
  A();
  A();
  B();
}
```

Cycle	1	2	3	4	5	6	7	8	9	10
Op 1	A	A	A	A	A	A	A	A	A	A
Op 2	–	B	–	B	–	B	–	B	–	B

Cycle	1	2	3	4	5	6	7	8	9	10
Op	A	–	A	B	A	–	A	B	A	–

Cycle	1	2	3	4	5	6	7	8	9	10
Op	A	A	B	A	A	B	A	A	B	A

Synthesis report comparison

450 pixel wide 1D Gaussian pyramid with 3 levels