Imagine

Ujval Kapasi
Concurrent VLSI Architecture Group

August 28, 2001
Media Processing Applications

- Graphics
- Image & Video Processing
- Audio & Signal Processing

Characteristics
- Abundant Data-parallelism
- Many operations per memory reference (10s-100s)
- Producer-consumer locality
- Low-precision data types

Challenge
- Extract the available parallelism
- Supply data & instruction bandwidth
Stereo Depth Extraction

- 640x480 @ 30 fps
- Requirements
  - 11 GOPS
- Imagine stream processor
  - 12.1 GOPS, 4.6 GOPS/W
Stream Processing

- Little data reuse (pixels never revisited)
- Highly data parallel (output pixels not dependent on other output pixels)
- Compute intensive (60 operations per memory reference)
Locality & Concurrency

Operations within a kernel operate on local data

Kernels can be partitioned across chips to exploit control parallelism

Streams expose data parallelism
Imagine Stream Processor

- Host Processor
- Stream Controller
- Microcontroller
- Stream Register File
- Streaming Memory System
- SDRAM
- Network Interface
- ALU Cluster 0 to ALU Cluster 7
Stereo Depth Extraction

Global Data
- Right Input Image
- Left Input Image

Streams
- Row of Pixels
- Previous Partial Sums
- New Partial Sums
- Blurred Row
- Previous Partial Sums
- New Partial Sums
- Sharpened Row
- Filtered Row Segment
- Previous Partial Sums
- New Partial Sums
- Depth Map Row Segment

Kernels
- Convolution (Gaussian)
- Convolution (Laplacian)

Depth Map
Imagine Bandwidth Hierarchy

Stream Register File

SDRAM
SDRAM
SDRAM
SDRAM

ALU Cluster
ALU Cluster

Depth Unconstrained

0.99GB/s
22GB/s
248GB/s

Depth Actual

0.80GB/s
18GB/s
211GB/s

2GB/s
32GB/s

544GB/s
Application Development Flow

- Compose stream/kernel diagram
  - Identify data parallelism and map to SIMD clusters
  - Stream-oriented algorithm choices
- Write kernel code
- Write stream code
- Performance tuning
Stream Programming System

Stream Application

Stream program (StreamC)

StreamC compiler

Host processor

Kernels (KernelC)

KernelC compiler

Imagine processor
Stream program

- Defines control- and data-flow between kernels

- StreamC (mixed with C++)
  - Declare streams
  - Call kernels

```cpp
def doFooBar():
    stream<int> a(256), b(256), c(256), d(256);
    ...
    foo1(a, b, c);
    bar(c, d);
    ...
```
Stream Compiler

- Allocates the stream register file (and other Imagine resources)
- Determines when to load and store streams
- High-level optimizations

```
foo1(a, b, c);
bar(c, d);
```
Kernels

- Function that operates on streams

- KernelC
  - Loop over stream
  - Read/write streams

- No random data access, limited control flow

```c
KERNEL foo (  
    istream<int> a,  
    istream<int> b,  
    ostream<int> c) { 
    loop_stream(a) { 
        int ai, bi, ci; 
        a >> ai;  
        b >> bi;  
        ci = ai * ai + bi;  
        c << ci;  
    } 
}
```
KernelC compiler

- VLIW scheduler optimized for kernels
- Communication scheduling to handle
  - shared interconnect
  - multiple register files
- Single-phase scheduling
- Modulo software-pipelining

\[ ci = ai \times ai + bi; \]
for (row = 0; row < (NROWS+4); row++) {
    OldSums7x7 <= NewSums7x7
    OldSums3x3 <= NewSums3x3

    // Perform 7x7 Guassian convolution on the current row
    if (row < NROWS)
        convolve7x7(InputRow, OldSums7x7, &ConvRow7x7, &NewSums7x7);
    else if (row < (NROWS+3))
        drain7x7(OldSums7x7, &ConvRow7x7, &NewSums7x7);

    // Perform 3x3 LaPlacian convolution on the current row
    :

    // Store the filtered image
    if (row > 3) {
        // Append ConvRow3x3 to ConvolvedImage
    }
}
Convolve7x7(InputRow, OldSums7x7, 
&ConvRow7x7, &NewSums7x7)
{
...
while( !InputRow.empty() ) {
    half2 in10, ... ;
    InputRow >> in10;
    ...
    communicate_data_to_neighbors();

    p0 = k0  * in10;
    p12 = k21 * in32;
    p34 = k43 * in54;
    p56 = k65 * in76;
    sum = (p0 + p12)+ (p34 + p56);
    ...
}
}