Think sequential,
Go parallel
Overview

- Problem statement
- The Vector Fabrics solution
- Automatic program analysis
- Program partitioning
- Constructing the parallel program
How to create a parallel program?

- Use parallel programming language or library
  Exposed parallelism: CSP, Kahn, Open-CL, MPI
- Extend sequential language with threads
  Edward Lee, 2006: The Problem with Threads
- Extend sequential language with annotations
  Open-MP
Problem statement (2)

- Partitioning choice is driven by technology constraints
  
  Hardware cost, synchronization cost, bus bandwidth

- Different partitioning required over product lifetime, range
  
  Reuse high-end TV module 3 years later in mid-range TV set

Can we quickly find a program partitioning that matches our target technology?
The Vector Fabrics solution

1. Create sequential program in C/C++
   - Fully deterministic, matches capabilities of human brain
   - Largely independent of target technology

2. Interactively explore partitioning scenarios
   - Using graphical interface of vfAnalyst

3. Implement selected partitioning
   - Automatically generate object code for CPUs, and Verilog for hardware
   - Alternatively, run refactoring script on original source code
Automatic program analysis (1)

• Build a model of the program
  Compile to analysis VM, run with representative input data
  Analysis VM observes program behavior, captures many details

• Display the model graphically

Complete function and loop hierarchy, box width represents work load
Complete load/store analysis: memory deps, streaming access patterns
Fully cross-linked with original source code
Automatic program analysis (2)

Streaming access patterns

- Well-behaved memory access patterns
- Represent opportunities for parallelism
- Complicated patterns, not only simple FIFO
- Pattern detector captures all the essentials
  - All synchronization options, from fine-grained to course-grained
  - Required communication buffer size
  - Source identifier of the used memory space
  - All store and load instructions involved, with source file cross-probing
  - Channel bandwidth
Can we partition loop L2?

- Yes, cut streaming dependency from funB to funA
- Distribute loop L2 over two parallel threads
Program partitioning (2)

main
L1

funA

L2a

L2b
funB

task 0

task 1

Think sequential, Go parallel
vfAnalyst helps you explore your partitioning options

1. Right-click a loop you want to explore, select “optimize”
2. vfAnalyst displays a list of feasible partitioning options
   includes speedup, synchronization overhead, cost, implementation effort
3. Click your preferred partitioning option
4. vfAnalysts redraws the screen, showing partitioned design

Repeat until your speedup target is met
Constructing the parallel program (1)

Original program:

```c
funB(int i)
{
    frame[i] = ...;
}

funA(int i_prev)
{
    x += frame[i_prev];
}
```

Partitioned program with channel:

```c
funB(int i)
{
    fifo_wr(port1, ...);
}

funA(int i_prev)
{
    x += fifo_rd(port2);
}
```

Partitioned program with shared memory:

```c
funB(int i)
{
    frame[i] = ...;
    V(frame_semaphore);
}

funA(int i_prev)
{
    P(frame_semaphore);
    x += frame[i_prev];
}
```
Constructing the parallel program (2)

Original program:

```
main ()
{
    for (i=0; i<n; n++) acc += foo % bar;
    for (i=0; i<n; n++) {funA(i-1); funB(i);}
}
```

Partitioned program:

```
main ()
{
    for (i=0; i<n; n++) acc += foo % bar;
    fifo_wr(port1,frame[-1]);
    PARALLEL {
        for (int i=0; i<n; n++) funA(i-1); /* task 0 */
        for (int i=0; i<n; n++) funB(i); /* task 1 */
    }
    not_used = fifo_rd(port2);
}
```
Conclusions

We need parallel programs to match technology constraints

But: the human brain is wired to think sequentially

So we perform interactive partitioning of a sequential program

The Vector Fabrics solution

1. Create sequential program in unconstrained C or C++
2. Interactively explore partitioning options to match target technology
3. Retrieve partitioned application (either fully synthesized or as refactoring script)
Any software engineer can create parallel multi-core systems

- Little effort, no application insight required
- No system expertise required
- Predictable results

www.vectorfabrics.com