Programming Many-core Systems with Embedded Domain Specific Languages

Satnam Singh
satnams@microsoft.com
METROPOLIS
Heterogeneous => Multilingual?

• If future systems are heterogeneous then do we need to learn different languages and tools for different types of processing elements?
  – Concurrent and a parallel programming in C#, F#?
  – Parallel functional programming in Haskell?
  – VHDL and Verilog for hardware?
  – CUDA or Accelerator for GPUs?
Functional Domain Specific Languages

• A domain specific language captures the **essence** of a class of useful applications.
• Say what is important, leave out what is not.
• Greater freedom to compile to heterogeneous targets.
• **Embedded** Domain Specific Languages.
An Example of a DSL: Lava

• Haskell library
• Higher order functions for circuit behaviour and layout composition
• Library of basic Xilinx FPGA gates
• Library of useful circuits (arithmetic)
• Generates EDIF, VHDL and Verilog
• Interfaces to existing circuit design tools
• Chalmers: formal verification
• Other targets include multi-threaded software and GPUs
FPGAs

Functional Block Diagram

The new Xilinx Virtex architecture features Xilinx SelectRAM memory of distributed, block, and high-speed access to external RAM via the SSTL-3 standard, phase-locked loops (PLL), Virtex Select I/O pins, Xilinx segmented routing, and vector-based interconnect.
Look-up Table

Figure 16: Virtex-II Slice (Top Half)
LUTs as higher order functions

\[
\begin{align*}
\text{inv} &= \text{lut1 \ not} \\
\text{and2} &= \text{lut2 (&&)} \\
\text{mux} &= \text{lut3 (\lambda s d0 d1 . if s then d1 else d0)}
\end{align*}
\]
twoSorter clk = fork2 => fsT comparator => condSwap clk
Serial Composition
Functional Geometry
Behavior and Layout Composition

inv <-< and2

and2 ∧ inv

inv ∨ and2
and2 \rightarrow \text{fd clk} \quad \text{and2} \rightarrow| \text{fd clk}
Four sided tiles
4-bit adder
Registered Adder
Example: A Parallel Sorter

- Batcher’s parallel merger/sorter
- Carefully design and hand-layout one core element: two-sorter
- Use higher order combinantors to recursively combine this core element to produce a larger sorter
- Results in unprecedented layout compactness and excellent performance
- Beautiful description
Batcher’s Bitonic Sorter
Batcher’s Bitonic Sorter
Batcher’s Bitonic Sorter
Batcher’s Bitonic Sorter
Batcher’s Bitonic Sorter
Batcher’s Bitonic Sorter: VHDL

```vhdl
function sorter (i : in num_array) return num_array is
  constant halfi : positive := i'length / 2;
begin
  if i'length = 2 then
    return two_sorter (i);
  else
    midsort(1 to halfi) := sorter (i(1 to halfi));
    midsort(halfi+1 to i'length)
      := reverse (sorter (i(halfi+1 to i'length)));
    return butterfly (midsort);
  end if;
end function sorter;
```
Simulation
VHDL sorter 8x8 Floorplan
VHDL Sorter 8x8 Wiring
two sorter

twoSorter clk = fork2 => fsT comparator => condSwap clk
Two-sorter layout
Two Sorter
riffle

riffle = halve --> zP --> unpair
unriffle = pair -> unzip -> unhalve
two R

two r = halve --> par [r,r] --> unhalve
two two_sorter

2S

4 9
9 4

2S

7 7
3 3

two two_sorter
evens f = chop 2 >>= maP f >>= concat
evens two_sorter
\[ ilv\ R = \text{unriffle} \implies \text{two } r \implies \text{riffle} \]
Butterflies

\[
\begin{align*}
\text{bfly } R \ 1 &= R \\
\text{bfly } R \ n &= \text{ilv} (\text{bfly } R \ (n-1)) \rightarrow \text{evens } R
\end{align*}
\]
bfly R 1

bfly R 1 = R
bfly R 2

bfly R 2 = ilv (bfly R 1)) --> evens R
= ilv R --> evens R
bfly R 3 = ilv (bfly R 2)) -> evens R
= ilv (ilv R -> evens R) -> evens R
bfly 3 two_sorter
function butterfly (i : in num_array)
    return num_array is
begin
    if i'length = 2 then
        return two_sorter (i) ;
    else
        return (evens_two_sorter (ilv_butterfly (i))) ;
    end if ;
end function butterfly ;
Batcher’s Bitionally Sorter

sortB cmp 1 = cmp
sorB cmp n
    = two (sortB cmp (n-1)) ->>
        pair ->> snD reverse ->> unpair ->>
        butterfly cmp n
sorter twoSorter 3
sorter twoSorter 4
sorter twoSorter 5
grep -v RLOC
FPGA Implementation
Lava

\[
\text{bfly } R \ 3 \ = \ ilv \ (\text{bfly } R \ 2)) \rightarrow evens \ R \\
= \ ilv \ (ilv \ R \rightarrow evens \ R) \rightarrow evens \ R
\]
Lava Sorters
data parallel description of FFT-style operations in an E-DSL

FPGA hardware (VHDL)

GPU code (Accelerator)

C#

CCR, Cw or System.Threading

Multi-cores
Lava Embedded in F#

```fsharp
let rec bfly r n =
    match n with
    1 -> r
    | n -> ilv (bfly r (n-1)) --> evens r
```
public delegate List<T> ButterflyElement<T>(List<T> v);

public static List<T> bfly<T>(Func<List<T>, List<T>> r, List<T> l)
{
    if (l.Count == 2)
    {
        return r(l);
    }
    else
    {
        return evens(r, ilv<T,T>(i => bfly(r, i), l)) ;
    }
}
Summary

- Future architectures will be heterogeneous.
- Domain specific languages will be essential tools for developing manycore software.
- Functional languages provide the right abstractions to build powerful embedded domain specific languages.