Concurrency expression in high-level languages, Best practice and amenability to h/w compilation.

(povocative statements for BoF Panel Discussion !)

David Greaves
University of Cambridge
Computer Laboratory

David.Greaves@cl.cam.ac.uk

Parallel Programming Disciplines

- Hardware is parallel (massively).
- Software must go parallel owing to end of clock frequency growth.
- Hardware is software is hardware – we need (an) effective expression language(s) amenable to codesign.

So: three classes of parallelism:

- 1. Embarrassingly parallel – no control or data interaction between strands.
- 2. Stream processing – pipelined parallelism – great if there are no control hazards.
- 3. General, fine-grained parallel programming!

David.Greaves@cl.cam.ac.uk

Eager versus Lazy Dichotomy

- Separating control and data flows often mooted:
  - It is the key enabler for 'Spatial Computing'
  - `A New Dataflow Compiler IR for Accelerating Control-Intensive Code in Spatial Hardware' A M Zaidi and DJ Greaves @ IPDPS'14.
- But why are people happier with OCAML than Haskell ?

General purpose language must keep them quite close together – e.g. call-by-value in ML/Java/C etc..

David.Greaves@cl.cam.ac.uk

Von Neumann Imperative Parallelism

- Shared-memory imperative programming is stupid – how have we got there?

- Using strongly typed C/C++/C# we can compile pointers and abstract data structures quite safely.

- But aliasing problem restricts available parallelism (w.r.t. critical ALU path) by:
  - Factor of 100 by conservative static analysis
  - Factor of < 10 in reality (Jonathan Mak's PhD).

David.Greaves@cl.cam.ac.uk

Eliminate shared memory ?

- *Eliminate it entirely – Erlang, Occam, Pi calculus and so on...*
  - *DRAM can still be used (thank god) but all regions are fully disambiguated and local to a task.*

- *Restrict to Imutable shared memory – preferably combined with an interlock to avoid RbW on initialisation.*

- *Do reference counting on your pointers –*
  - *Rust pointer mangament*
  - *or linear type systems with an explicit duplicate operator for sharing*

David.Greaves@cl.cam.ac.uk

Kiwi C# High-Level Synthesis

- Compile C# with some language restrictions:

- ➲ Program can freely instantiate classes but not at run time!
- ➲ Array sizes must all be statically determinable (ie at compile time).
- ➲ Program can use recursion but max depth must be statically determined.
- ➲ Stack and heap must have same shape at each run-time iteration of non-unwound loops.
- ➲ Program can freely create new threads but creation sites statically determined too.

David.Greaves@cl.cam.ac.uk

Kiwi HLS Concurrency

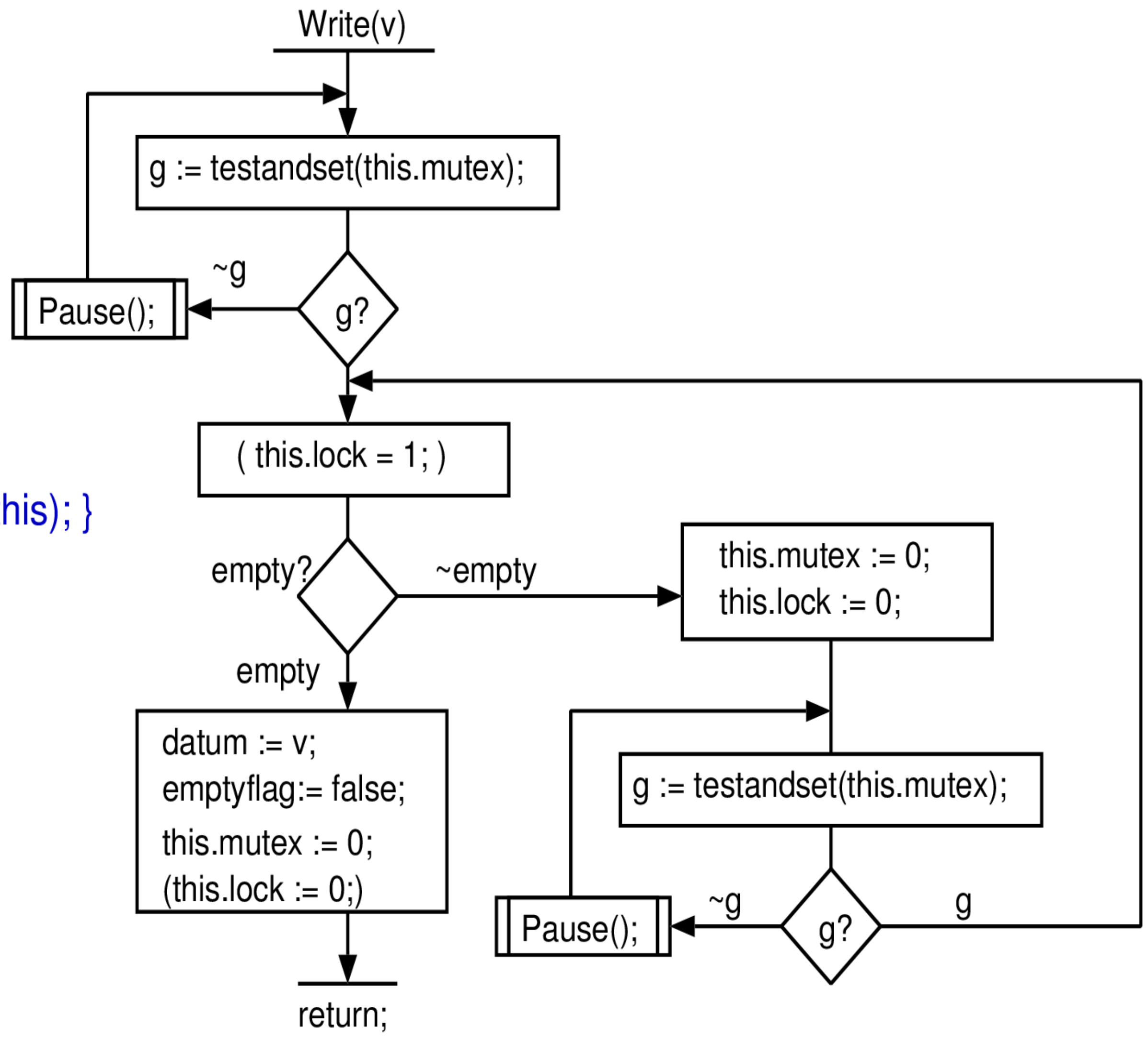
- Use the .net library concurrency primitives
- Below a certain level, replace implementations with our own hardware alternatives
- Each thread has classical HLS static schedule. Arbiters added to all threadshared resouces.
- This is ultimately a shared-variable model with exclusion locks.
- But what about async dispatch? Later.

David.Greaves@cl.cam.ac.uk

Kiwi Example: One-place buffer Write Method.

```
public class Channel<T>
  {
     T datum;
     volatile bool emptyflag = true;

     public void Write(T v)
     {  lock (this)
        {
           while (!emptyflag) {  Monitor.Wait(this); }
           datum = v;
           emptyflag = false;
           Monitor.PulseAll(this);
        }
     }
     ...
 }
```



David.Greaves@cl.cam.ac.uk

Field Programmable Logic and Applications (FPL) 2nd September 2014, Munich.

Bluespec Verilog (BSV)

```
// A simple rule
rule  rule1 (emptyflag && req);
    emptyflag <= false;
    ready <= true;
endrule
```

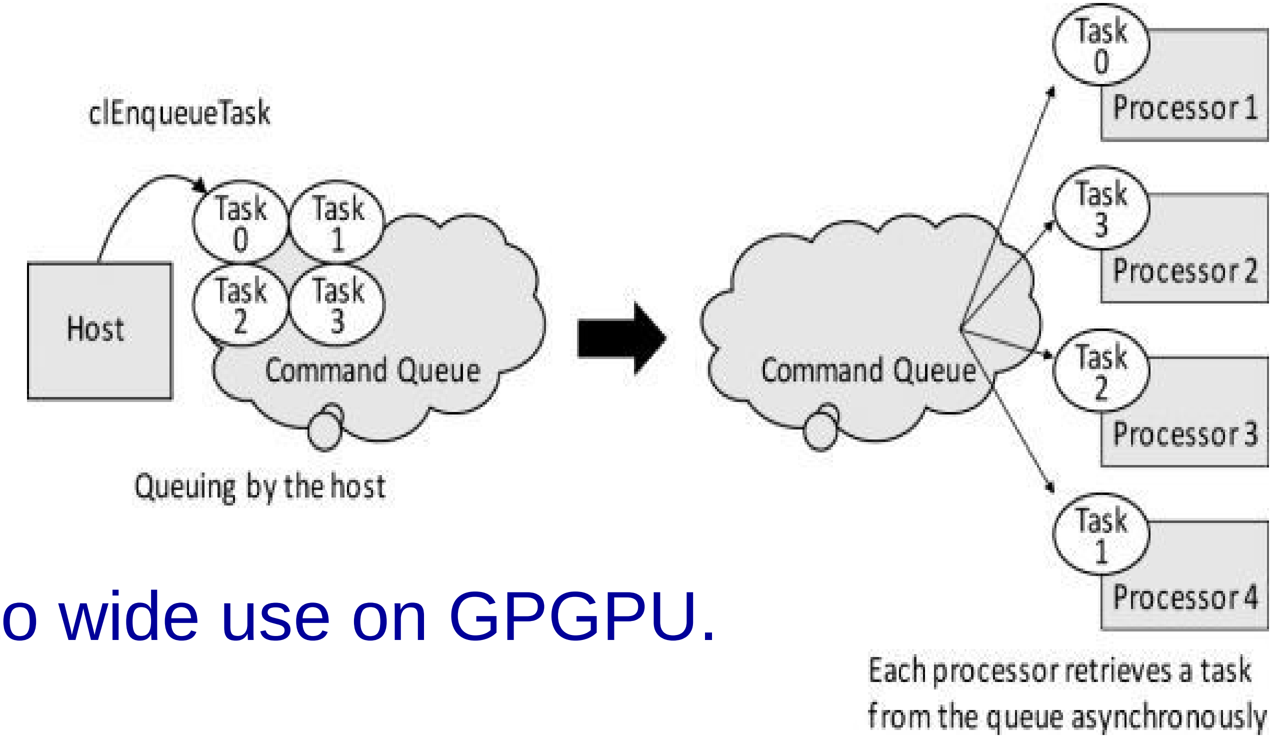
Design is expressed as guarded atomic actions. So locking primitives are innate.

- Parallelism comes from rules firing in parallel.
- Performance comes from packing multiple, potentially interacting rules into one execution clock cycle.
- All rules gen'd by a rich static elaboration language.

Stuttering is the default semantic – unless `must fire' pragma is applied: THIS LEADS TO RaW HAZARDS ON RAMs AND REGs SO MUST USE FIFOs WHEREVER POSSIBLE OR ELSE PAY ATTENTION TO WARNING MESSAGES.

David.Greaves@cl.cam.ac.uk

Open CL



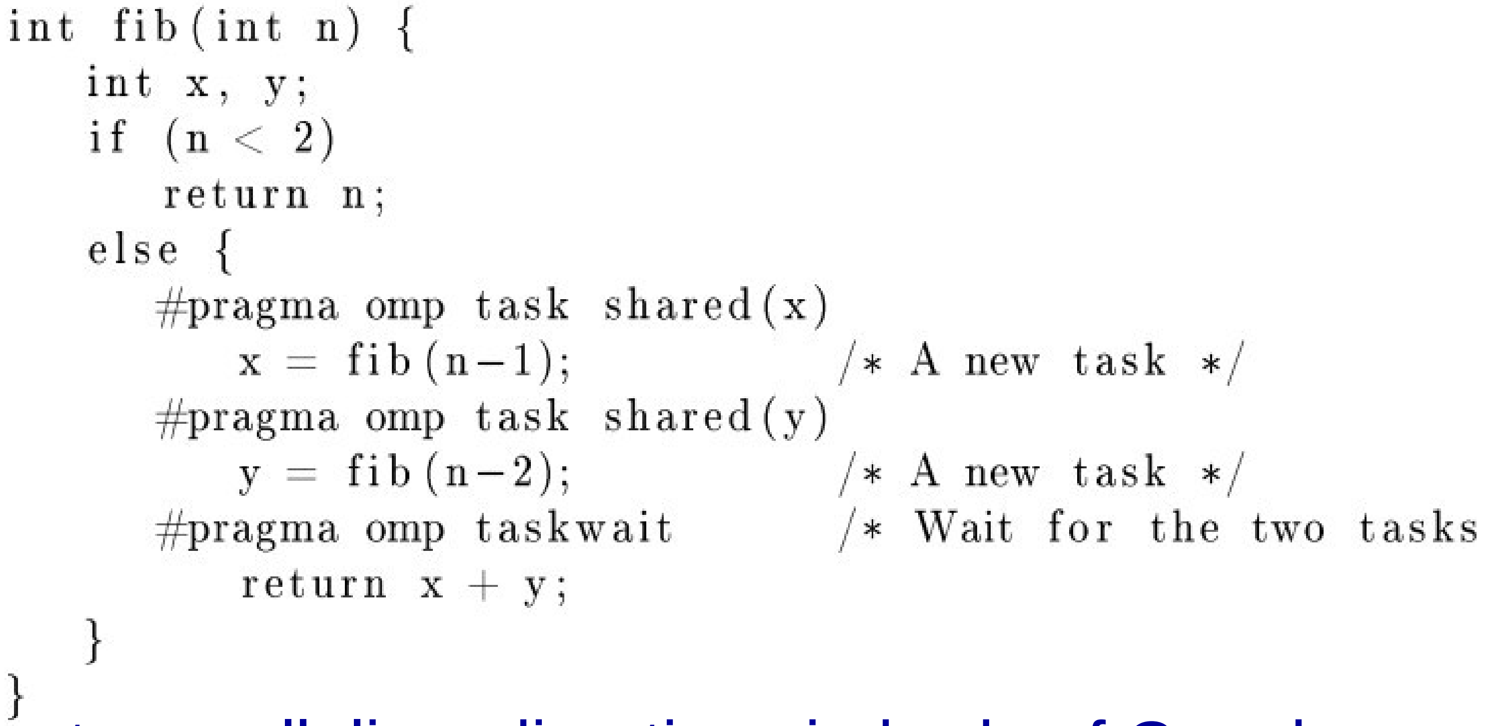
Interest arises owing to wide use on GPGPU.

Programmer manually:
- splits inner loop kernels off for separate compilation.
- allocates storage over a 4-level hierarchy with pragmas
- makes calls to the GPU work queues.

Open Computing Language (OCL)
- is not much of a language
- is more of an accelerator API.

David.Greaves@cl.cam.ac.uk

CILK, OpenMP & WOOL



```
int fib(int n) {
    int x, y;
    if (n < 2)
        return n;
    else {
       #pragma omp task shared(x)
           x = fib(n-1);              /* A new task */
       #pragma omp task shared(y)
           y = fib(n-2);              /* A new task */
       #pragma omp taskwait           /* Wait for the two tasks
           return x + y;
    }
}
```

Programmer inserts parallelism directives in body of C code.

Program still can run single-threaded by just ignoring the mark up.

Array accesses are essentially unaltered but

- great care over aliasing is needed, and
- no type-system or language-level assistance for correctness.

Join Calculus

```
//A simple join chord:
public class Buffer
{
    public async Put(char c);
    public char Get(bool f) & Put(char c)
                             { return (f)?toupper(c):c; }
}
```

Joins are elegant.

Joins substrate implements workqueues and schedullers.

Queue capacity requires careful dimensioning – too small can deadlock.

A long way from hardware design but probably a good way forward for general parallel programming targeting FPGA!

Hardware join Java: a unified hardware/software language for dynamic partial runtime reconfigurable computing applications - Kearney,.

Mapping the Join Calculus to Heterogeneous Hardware
Peter Calvert,Alan Mycroft

Polyphonic C Sharp – Benton and Cardelli.

Asynchronous Task Calls:
The C#5 / Scala – await primitive.

```
public async Task<int> SumPageSizesAsync(IList<Uri> uris)
  {
      int total = 0;
      foreach (var uri in uris) {
         statusText.Text = string.Format("Found {0} bytes ...", total);
         var data = await new WebClient().DownloadDataTaskAsync(uri);
         total += data.Length;
      }
      statusText.Text = string.Format("Found {0} bytes total", total);
      return total;
  }
```

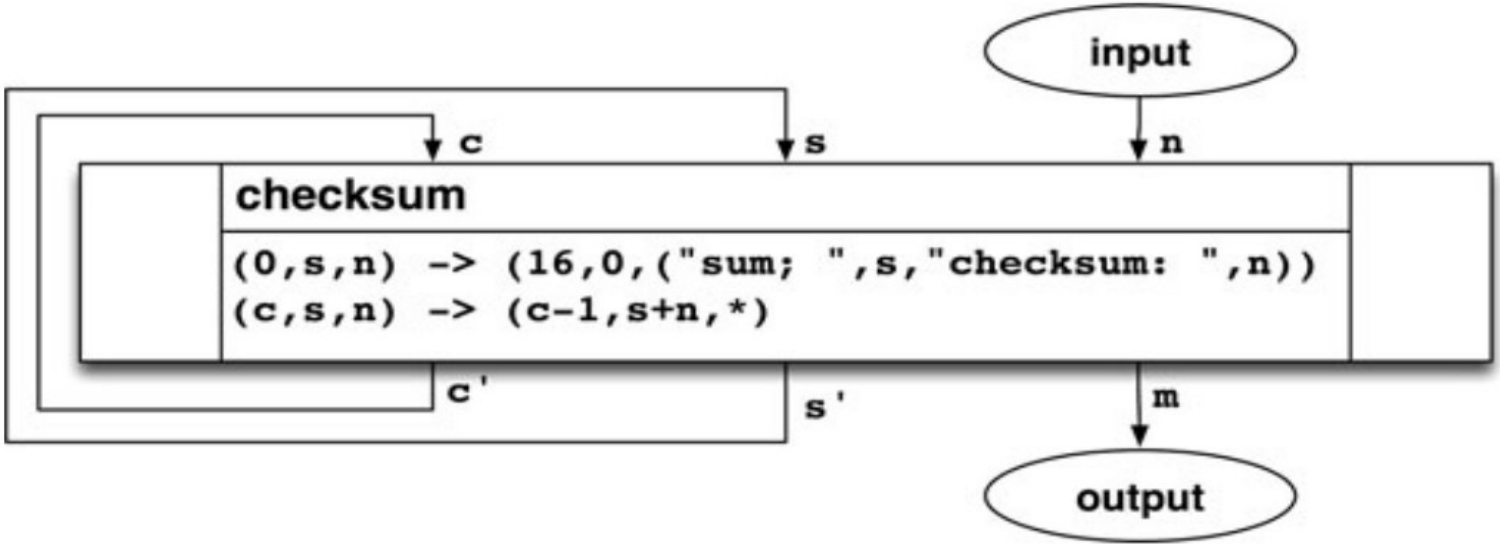
Also in Java as
a FutureTask

A simple version of the full join calculus.

Bounded queue and scheduler overhead => implies => practical for engineers (not like Haskell!)

Suitable for large scale systems. Can adapt to different quantities of execution resource by work stealing etc..

Hume Box Algebra



```
checksum
(0,s,n) -> (16,0,("sum; ",s,"checksum: ",n))
(c,s,n) -> (c-1,s+n,*)
```

Classical textual/ASCII language, but...

Programs are a multitude of connected stateless boxes.

Can be hierarchic with a complete box graph nested inside a single parent's box.

Amenable to algebraic manipulations for time/space folding.

Is there a problem, as always, with large data in DRAM ?
No – use the Erlang/Occam localised arrays solution.

David.Greaves@cl.cam.ac.uk

Conclusion

- Concurrent expression of HLS design intent is a good thing: - makes more parallelism available.

- Large arrays in (D)RAM are the most important entity in all types of computing
  - especially non-stream, non-embarrassingly parallel.

- Parallel programming paradigms must eliminate pointer ambiguity
  - 1. as far as possible,
  - 2. without precluding it for the few algorithms that actually delploy pointer aliasing (and even they are mostly just read pointers).
- Optimising schedulers for concurrent specification languages shall emerge (but engineers need write-time handle on complexity).

David.Greaves@cl.cam.ac.uk