Research Accelerator for Multiple Processors

Architecture, Language and Compiler

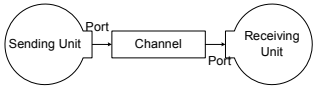
Presented 1/8/2007
BWRC Winter Retreat,
Monterey Bay, CA

Greg Gibeling (gdgib@berkeley.edu)
http://www.eecs.berkeley.edu/~gdgib
http://ramp.eecs.berkeley.edu
Thanks to Krste Asanovic, John Wawrzynek,
Andrew Schultz and Nathan Burkhart

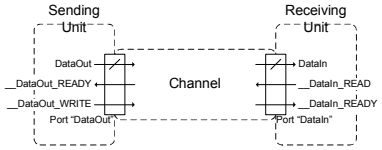
Introducing RAMP & RDL

- Research Accelerator for Multiple Processors (RAMP)
  - A RAMP System is a cross platform architectural simulator.
  - Designed to foster community research (Share & verify results).
  - Orders of magnitude faster than existing solutions.
  - Eases component re-use and integration.
- RAMP Description Language (RDL)
  - A distributed event & message passing system framework.
  - The "Target System," the system being emulated, is captured in RDL and emulated on the "Host System," an FPGA or CPU.
  - Allows virtualization of simulation time to provide cycle accurate simulation results. Compiled by RDLC into Verilog or Java.
- Current Applications of RDL
  - RAMP Blue: A RAMP System built using Xilinx uBlaze processors on the BEE2
  - FLEET: A highly concurrent processor architecture focused on comm. not comp.
  - Overlog/P2: A distributed dataflow description language and execution system

The "Target" Model

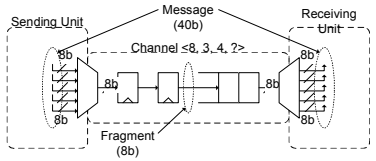


Basic "Target" Model



Hardware Signals for Unit Ports

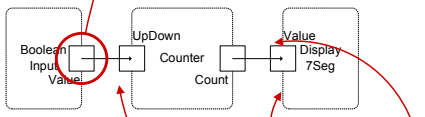
- Units communicate over point-to-point, unidirectional channels
  - A unit would be ~10,000 gates (Processor + L1 cache)
  - Units are implemented in the "host" language, eg. Verilog
  - Existing message passing hardware/software can be ported easily
- Channels Delay Model
  - Allows cycle accurate simulations
  - Staticly typed, variable size messages
  - Bitwidth (Fragment)
  - Latency (Both directions)
  - Buffering
- Benefits
  - Automatic channel implementation
  - Ease of unit re-use and sharing



Message Fragmentation

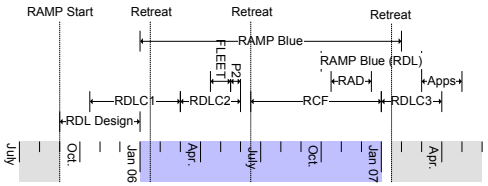
The Counter Example

```
unit {
   input     bit[1]              UpDown;
   output    bit[32]             Count;
}                                Counter;

unit {
   output    bit[1]              Value;
   attribute CaLinx2             "input:1:SW_";
   attribute CaLinx2             "input:1:BTN_";
}                                IO::BooleanInput;
   Platform specific attributes
   (For external connections)
platform {
   language                      "verilog";
   default link                  "SerialLink";
}                                CaLinx2;

map {
   platform  CaLinx2             BasePlatformInst;
   unit      CounterExample      BaseUnitInst;
}                                CaLinx2Counter;
```




```
unit {
   instance  IO::BooleanInput    BooleanInput;
   instance  Counter             Counter;
   instance  IO::Display7Seg     Display7Seg;

   channel   fifopipe[1, 1, 1]   InChannel
      { BooleanInput.Value -> Counter.UpDown };
   channel   fifopipe[32, 1, 1]  OutChannel
      { Counter.Count -> Display7Seg.Value };
}                                CounterExample;
```

Timeline



RDLC1
Verilog & Java Output
Single Platform Projects

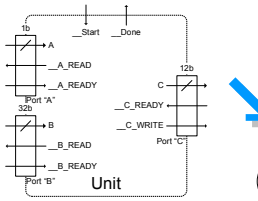
RDLC2
Verilog Output
Multiple Platform Projects
Parameters
Advanced Data Types
Plugins & Generators

RDLC3
Verilog & Java Output
Cross Platform Projects
Static & Dynamic Parameters
Timing & Link Library
Debugging & Control
Improved Project Management

- A simple unit
- No functionality in RDL

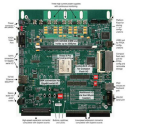
```
unit {
   input     bit[1]    A;
   input     bit[32]   B;
   output    bit[12]   C;
}                      Unit;
```

RDLC (Shell)

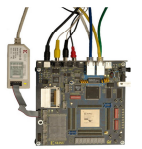


- Unit shell, ready for implementation
- Verilog, Java, etc...

Standard Compiler



Xilinx XUP

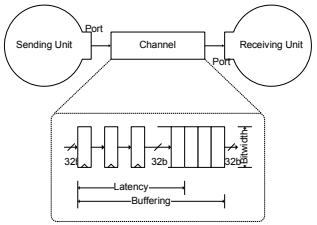


Berkeley CaLinx2

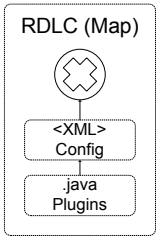
Any computer

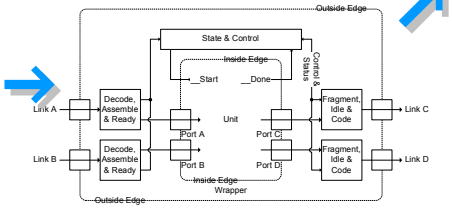
BEE2

- RDL Hierarchical Netlist
- Include channel model



- Support for multiple target languages
- Automatic cross platform implementation





- Complete host implementation

Toolflow