Bluespec: The need for a new design methodology

Arvind
Computer Science & Artificial Intelligence Lab.
Massachusetts Institute of Technology

February 13, 2008

Real power saving implies specialized hardware

- H.264 implementations in software vs hardware
  - the power/energy savings could be 100 to 1000 fold

but our mind set is that hardware design is

- Difficult, risky
  - Increased time-to-m
- Inflexible, brittle, er
  - How to deal with changing standards, errors

New design flows and tools can change this mind set

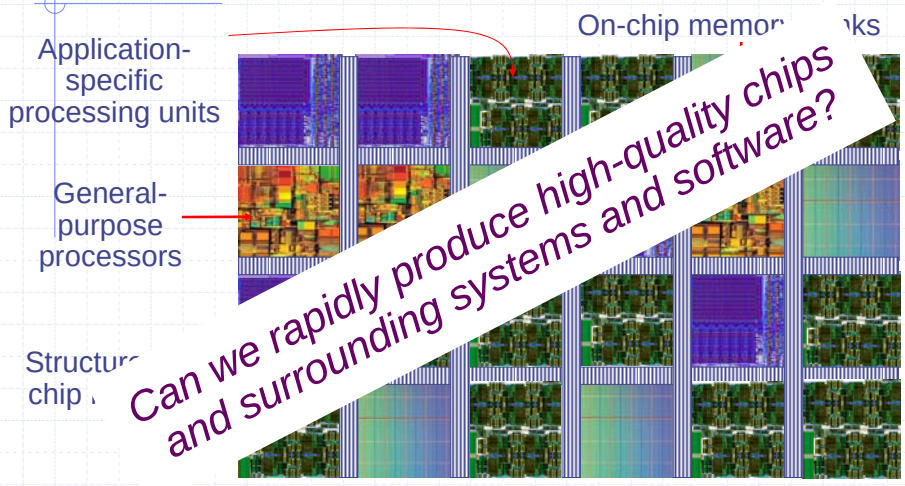
Economic relevance

- Cell phones, PDAs, sensors, …
  - ⇒ Demand a much greater variety of chips
- Cost of development, business risks, …
  - ⇒ Forces us towards specialization primarily through software

New tools can enable a much greater variety of chips

SoC Trajectory:

more application specific blocks



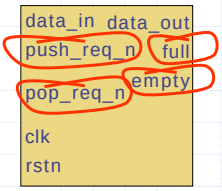
Making hardware design easier

- Extreme IP reuse — "Intellectual Property"
  - Multiple instantiations of a block for different performance and application requirements
  - Packaging of IP so that the blocks can be assembled easily to build a large system (black box model)
  - Whole system simulation to enable concurrent hardware-software development

Need new methods and tools to accomplish this goal

IP Reuse sounds wonderful until you try it ...

Example: Commercially available FIFO IP block



An error occurs if a push is attempted while the FIF[illegible]l.

Thus, there is no conflict in a simul[illegible]hen the FIFO is full. A simultaneous push and pop [illegible]FO is empty, since there is no pop data to prefetch. How[illegible]red in the FIFO.

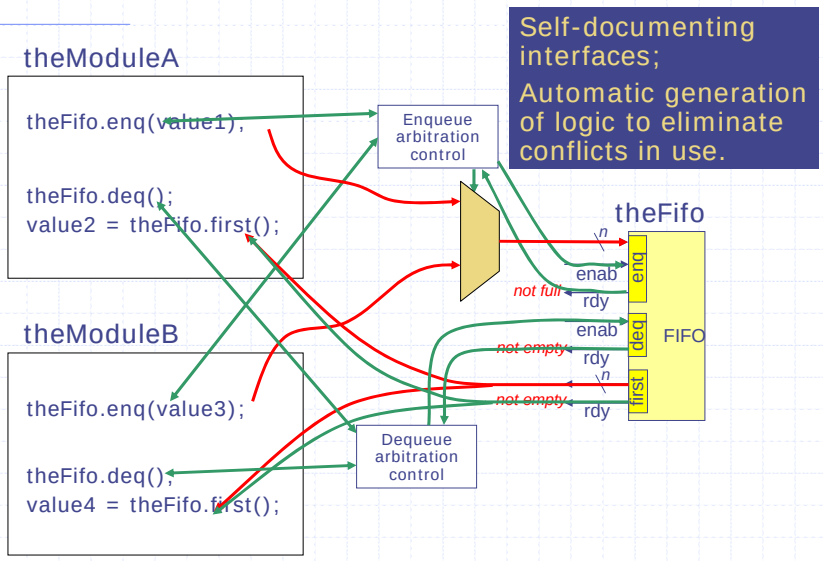
A pop [illegible] pop_req_n is asserted (LOW), as long as the FIFO is not em[illegible]eq_n causes the internal read pointer to be incremented on the nex[illegible] clk. Thus, the RAM read data must be captured on the clk following the a[illegible]on of pop_req_n.

No machine verification of such informal constraints is feasible

These constraints are spread over many pages of the documentation...

Bluespec can change all this

Bluespec promotes composition through guarded interfaces



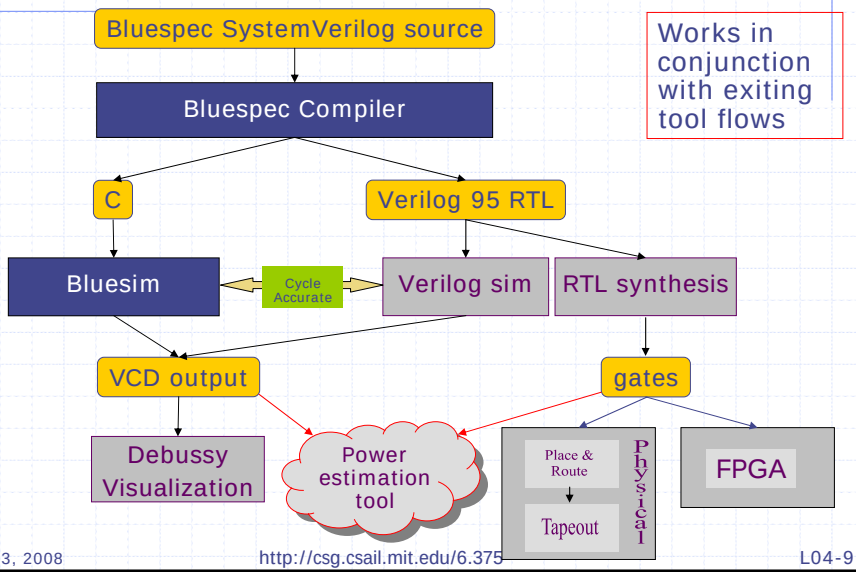
Bluespec: A new way of expressing behavior using Guarded Atomic Actions

- Formalizes composition
  - Modules with guarded interfaces
  - Compiler manages connectivity (muxing and associated control)
- Powerful static elaboration facility
  - Permits parameterization of designs at all levels
- Transaction level modeling
  - Allows C and Verilog codes to be encapsulated in Bluespec modules

➔ *Smaller, simpler, clearer, more correct code*

➔ *not just simulation, synthesis as well*

Bluespec Tool flow



Works in conjunction with exiting tool flows

Recent Applications

- Multiradio OFDM: From WiFi to WiMax
  - 802.11a and 802.16 from the same source
- H.264 Decoder
  - Baseline profile, 720p X ~75 frames
  - FPGA implementation working

Other examples: Processors, Cache Coherence Protocols, IP Lookup, ...

Research sponsors have agreed to publish all designs done at MIT under the MIT open source license

Importance of Publishing Bluespec Designs

- Enables whole community to undertake much more ambitious projects
  - We already see the effects in 6.375 projects
- Enables derivative designs, specializations and variety at a fraction of the development cost

Multi-radio OFDM workbench

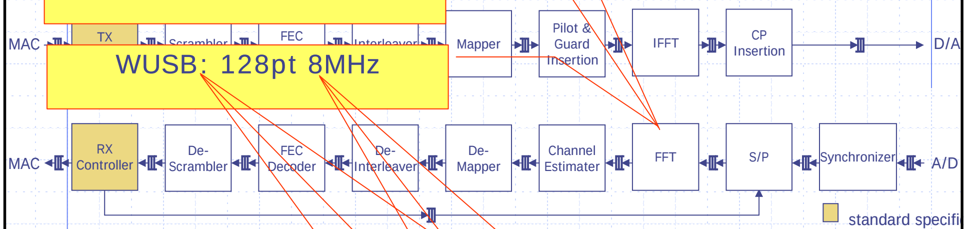
[MEMOCODE 2006, MEMOCODE 2007]

WiFi: 64pt @ 0.25MHz

WiMAX: 256pt @ 0.03MHz

WUSB: 128pt 8MHz

Parameterized modules
Example OFDM based protocols



Convolutional

Reed-Solomon

Turbo

- Reusable algorithm with different parameter settings
- 85% reusable code between WiFi and WiMAX
  From WiFi to WiMAX in 4 weeks
- Different algorithms

(Alfred) Man Chuek Ng, ...

802.11a Architectural Exploration
(Only the IFFT block is changing) [MEMOCODE 2006]

These designs were done in ~ 3 man-days

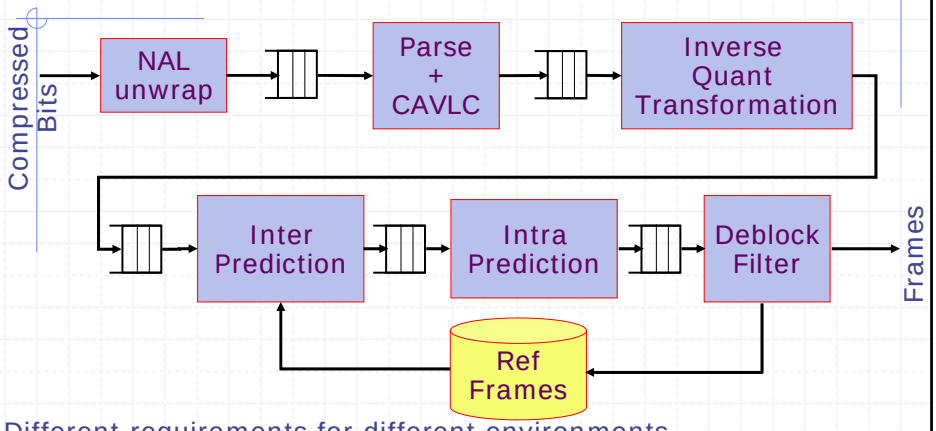
IFFT Design	Area (mm^2)	Symbol Latency (CLKs)	Throughput Latency (CLKs/sym)	Min. Freq Required	Average Power (mW)
Pipelined	5.25	12	04	1.0 MHz	4.92
Combinational	4.91	10	04	1.0 MHz	3.99
Folded (16 Bfly-4s)	3.97	12	04	1.0 MHz	7.27
Super-Folded (8 Bfly-4s)	3.69	15	06	1.5 MHz	10.9
SF(4 Bfly-4s)	2.45	21	12	3.0 MHz	14.4
SF(2 Bfly-4s)	1.84	33	24	6.0 MHz	21.1
SF (1 Bfly4)	1.52	57	48	12 MHZ	34.6

TSMC .18 micron; numbers reported are before place and route. (DesignCompiler), Power numbers are from Sequence PowerTheater

Video Codec: H.264

Chun-Chieh Lin (MIT MS thesis 2006)
Kermin Elliott Fleming

H.264 Video Decoder



Different requirements for different environments
- QVGA 320x240p (30 fps)
- DVD 720x480p
- HD DVD 1280x720p (60-75 fps)

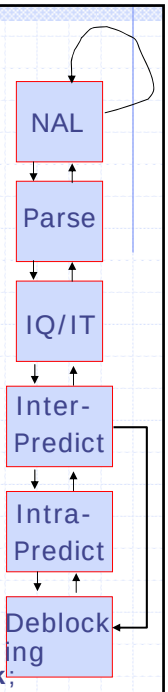
May be implemented in hardware or software depending upon ...

Sequential code

from ffmpeg

```
void h264decode(){
      int stage = S_NAL;
 while (!eof()){
  createdOutput = 0; stallFromInterPred = 0;
  case (stage){
    S_NAL: try_NAL();
        if (createdOutput) stage = S_Parse; break;
    S_Parse: try_Parse();
        stage=(createdOutput) ? S_IQIT: S_NAL; break;
    S_IQIT: try_IQIT();
        stage=(createdOutput) ? S_Parse:S_Inter; break;
    S_Inter: try_Inter();
        stage=(createdOutput) ? S_IQIT:S_Intra;
        if (stallFromInterPred) stage=S_Deblock; break;
    S_Intra: try_Intra();
        stage=(createdOutput) ? S_Inter:S_Deblock; break;
    S_Deblock: try_deblock(); stage=S_Intra; break } } }
```

20K Lines of C out of 200K



Parallelizing the C code

First step towards hardware generation from C

- Control structure is totally over specified and unscrambling it is beyond the capability of current compiler techniques
- Program structure is difficult to understand
- Packets are kept and modified in a global heap

Some of these problems can be avoided by providing the programmer a few parallel constructs

H.264 Learnings

- Productivity: Base profile
  - Effort: Less than one-man year
  - 8K lines of Bluespec (contrast 20k to 80K lines of C)
  - First draft decoded 720p @ ~32fps, (Available C codes do not meet this performance)
- Architectural Exploration: Many improvements made over a period of several months to increase performance and reduce area
  - Process several samples / cycle
  - Adjust FIFO depths
  - Pipeline modules: Interpolator, Deblocking filter
  - After improvements decodes 720p @ ~95fps (180nm)

Modular refinement is both feasible and essential

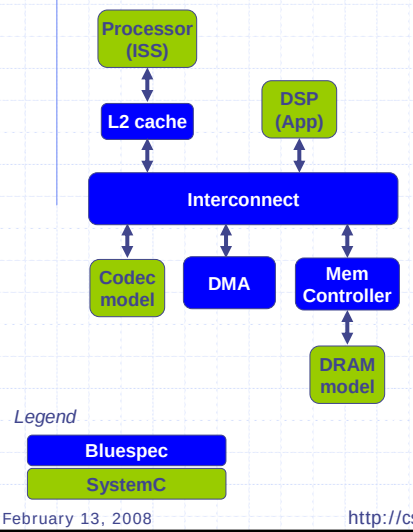
H.264 Design Exploration

	Area (mm^2)	Cycles /pixel	Cycle time (ns)	FPS 1280x720
First draft	5.44	2.90	11.81	31.66
4 samples / FIFO elt	5.32	1.65	14.53	45.24
4 samples / cycle	5.45	1.53	11.87	59.62
Larger FIFOs	6.04	1.32	11.82	69.67
Interpred in parallel	6.09	1.28	11.73	72.20
Pipelined interp	6.88	1.24	13.14	66.46

Tower 180nm library

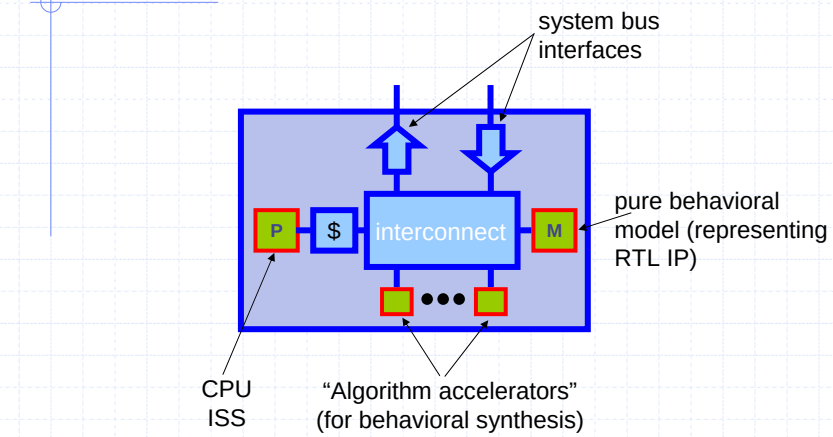
Bluespec for System Modeling and Synthesis

A typical SoC model



- The model may contain a mixture of SystemC and Bluespec modules
- Typical SystemC modules:
  - CPU ISS models
  - Existing SystemC IP
  - Behavioral models in C or C++ targeted for synthesis
- Bluespec modules:
  - Complex control – difficult to model in SystemC
  - Hardware - realistic architectural exploration

Modeling Concurrency



Legend

Bluespec

SystemC

Programming the interconnect without an accurate timing model is slightly bogus

Modular refinement

- Is it easy to build Bluespec wrappers for a class of C codes
- Bluespec modules can be introduced early because they
  - Can be written at a very high level,
  - Can interface to other SystemC TLM modules
  - Can be refined into hardware/RTL
  - System-level testbenchs can be reused at all levels

Other ongoing collaborative projects

- Performance modeling on FPGAs
  - with Joel Emer at Intel
  - Speeding up the software performance model of IA-32 from 10Kips to 1-10Mips using FPGAs
- PowerPC model for FPGAs
  - with K. Ekanadham & Jessica Tsang at IBM
  - Boot Unix on an RTL model of a multi-threaded, multicore PowerPC on FPGAs
- Turbo decoder
  - with Jamey Hicks & Gopal Raghavan at Nokia
  - Integration of a parameterized Turbo decoder into an existing commercial design flow
- Accelerated test benches via FPGA
  - With Suhas Pai at Qualcomm
  - You will hear about it later in the course

Hardware synthesis: C-based tools vs Bluespec

- The goal of C-based tools (e.g., Catapult-C) is to generate good hardware given some area, timing, power or performance constraints
  - The tool explores the design space to come up with the "right" design
  - Language extensions are provided to overcome some of the limitations of C
- The goal of Bluespec is to enable the designer to generate a good implementation by letting him/her express the design at a high-level and explore alternatives via parameterization or refinement
  - No automatic exploration of the design space

Designer knows best – the tool automates some of the tedious and error-prone part of the hardware design process

Current research

- Make the path to hardware design easier
  - FPGA emulation infrastructure
  - Set up an infrastructure to study power related optimizations
  - Hardware-software interaction: test benches, device drivers, transaction-level modeling
  - Continue to explore new examples
- Semantic extensions and associated compiling schemes
  - The sequential connective: Control over scheduling, Multi-cycle atomic actions
  - Recursive method calls
- Exploratory: Compiling Bluespec for multicores

Bluespec promotes good Design methodology

- Can keep up with changing specs
- Permits architectural exploration
- Facilitates verification and debugging
- Eases changes for timing closure
- Eases changes for physical design
- Promotes reuse

Design for Correctness