

Bluespec: A New Way of Describing SoC Behavior

Arvind

Computer Science & Artificial Intelligence Lab.
Massachusetts Institute of Technology

ISSoC, Tampere, Finland
November 14, 2006

The biggest SoC drivers

- ◆ Explosive growth in markets for
 - cell phones
 - game boxes
 - sensors and actuators



Functionality and applications are constrained primarily by:

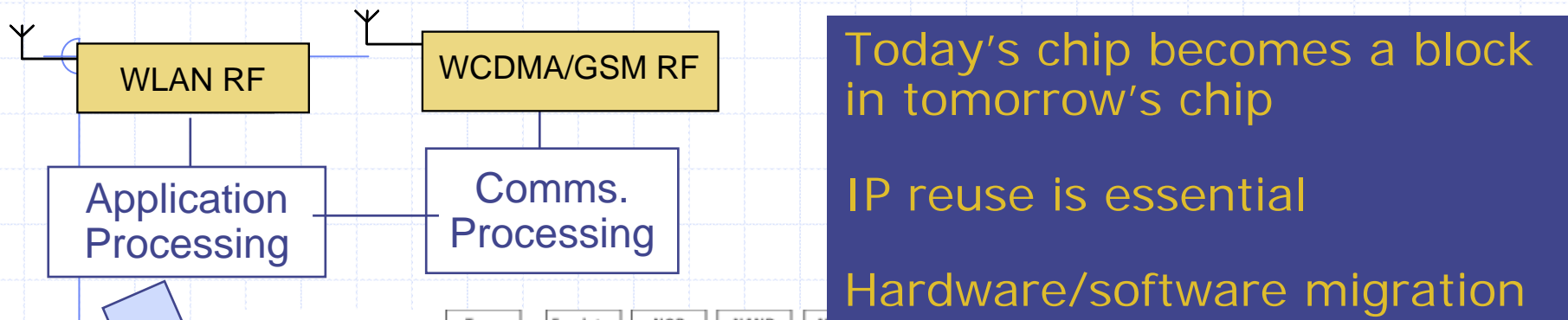
- cost
- power/energy constrains

Will there be a greater or smaller variety of chips in a decade from now?

- ◆ Cell phones, PDAs, sensors, ...
⇒ much greater variety
- ◆ Cost of development, business risks, ...
⇒ smaller variety

New design flows and tools can directly affect the outcome

Current Cellphone Architecture



Today's chip becomes a block in tomorrow's chip

IP reuse is essential

Hardware/software migration



Complex, High Performance, but must not dissipate more than 3 watts

An under appreciated fact

- ◆ If a functionality (e.g. H.264) is moved from a programmable device to a specialized hardware block, the power/energy savings are 100 to 1000 fold

Power savings ⇒ more specialized hardware

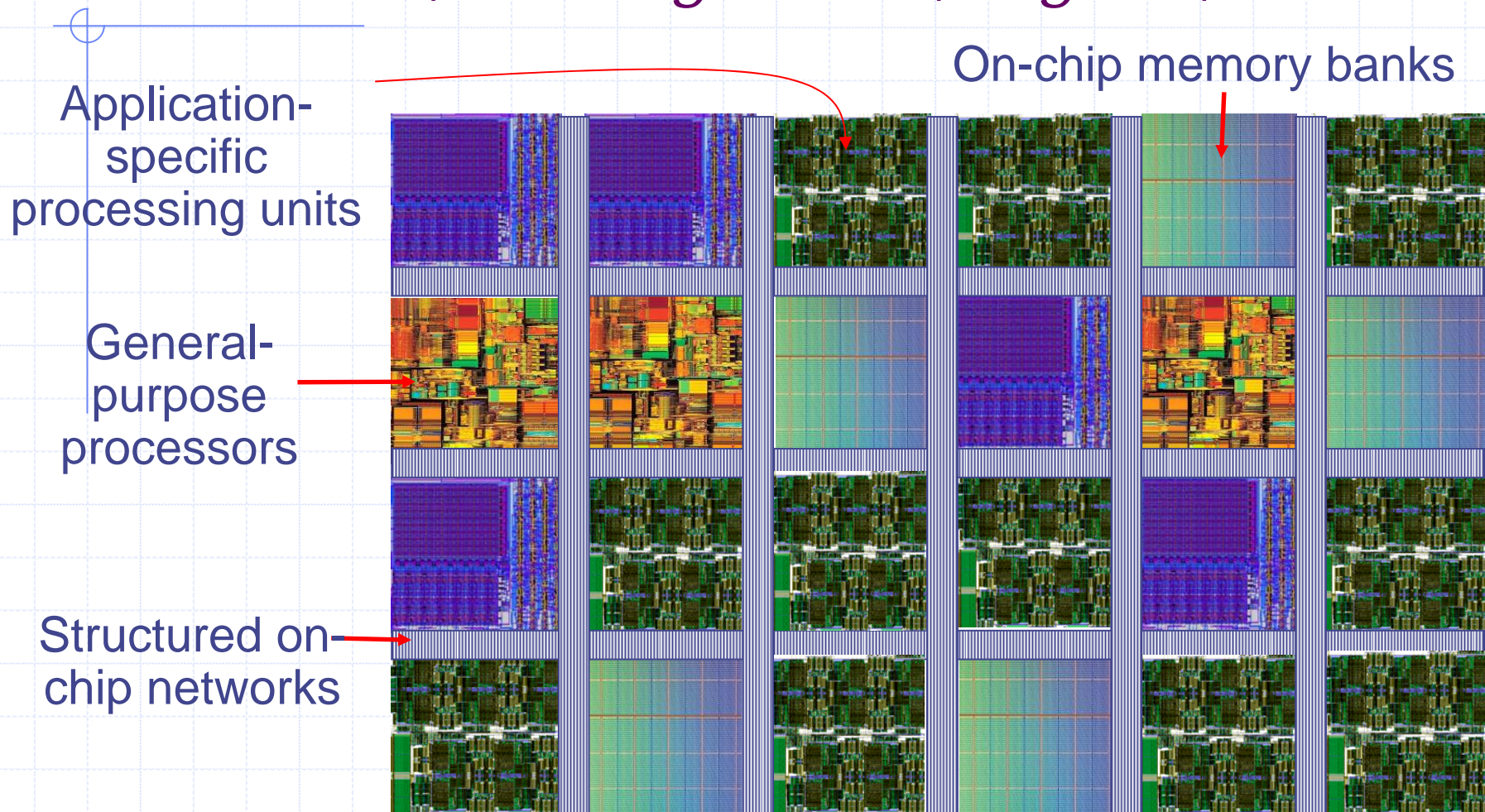
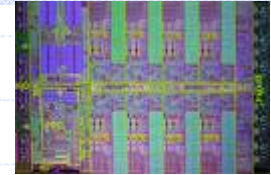
but our mind set

- Software is forgiving
- Hardware design is difficult, inflexible, brittle, error prone, ...

SoC Trajectory:

multicores, heterogeneous, regular, ...

IBM Cell
Processor



Can we rapidly produce high-quality chips and surrounding systems and software?

Things to remember

- ◆ Design costs (hardware & software) dominate
- ◆ Within these costs verification and validation costs dominate
- ◆ IP reuse is essential to prevent design-team sizes from exploding

design cost = number of engineers x time to design

Common quotes

- ◆ "Design is not a problem; design is easy"
- ◆ "Verification is a problem"
- ◆ "Timing closure is a problem"
- ◆ "Physical design is a problem"

Mind set

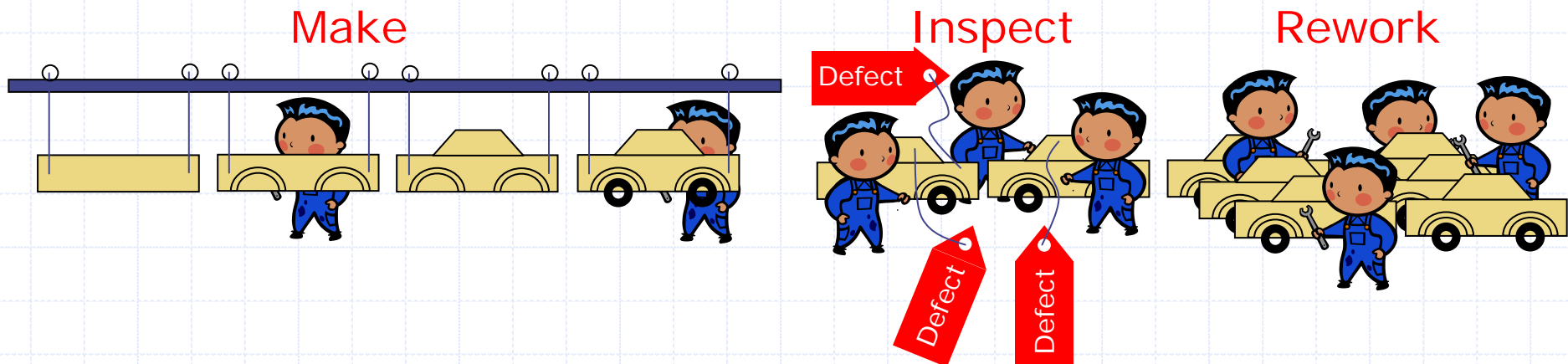
Almost complete reliance on post-design verification for quality

Through the early 1980s:



The U.S. auto industry

- ◆ Sought quality solely through post-build inspection
- ◆ Planned for defects and rework



and U.S. quality was...

... less than world class



- ◆ Adding quality inspectors (“verification engineers”) and giving them better tools, was not the solution
- ◆ The Japanese auto industry showed the way
 - “Zero defect” manufacturing

New mind set:

Design affects everything!

- ◆ A 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

⇒ It is essential to

Design for Correctness

New Design flows & tools

Synthesis as opposed to Decomposition

- ◆ A method of designing and connecting modules such that the functionality and performance are predictable
 - Must facilitate natural descriptions of concurrent systems
- ◆ A method of refining individual modules into hardware or software for SoCs
- ◆ New techniques for producing SoCs so that the development costs are proportional to the changes

New ways of expressing behavior to reduce design complexity

- ◆ Decentralize complexity: *Rule-based specifications (Guarded Atomic Actions)*
 - Lets you think one *rule* at a time
- ◆ Formalize composition: *Modules with guarded interfaces*
 - Automatically manage and ensure the correctness of connectivity, i.e., correct-by-construction methodology

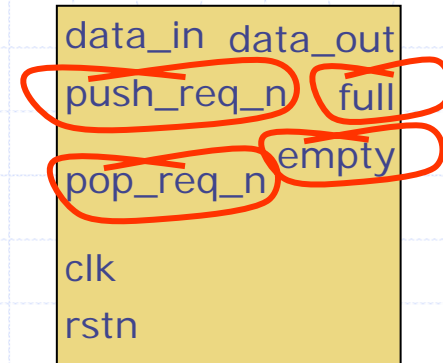
Bluespec

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

→ *not just simulation, synthesis as well*

Reusing IP Blocks

Example: Commercially available
FIFO IP block



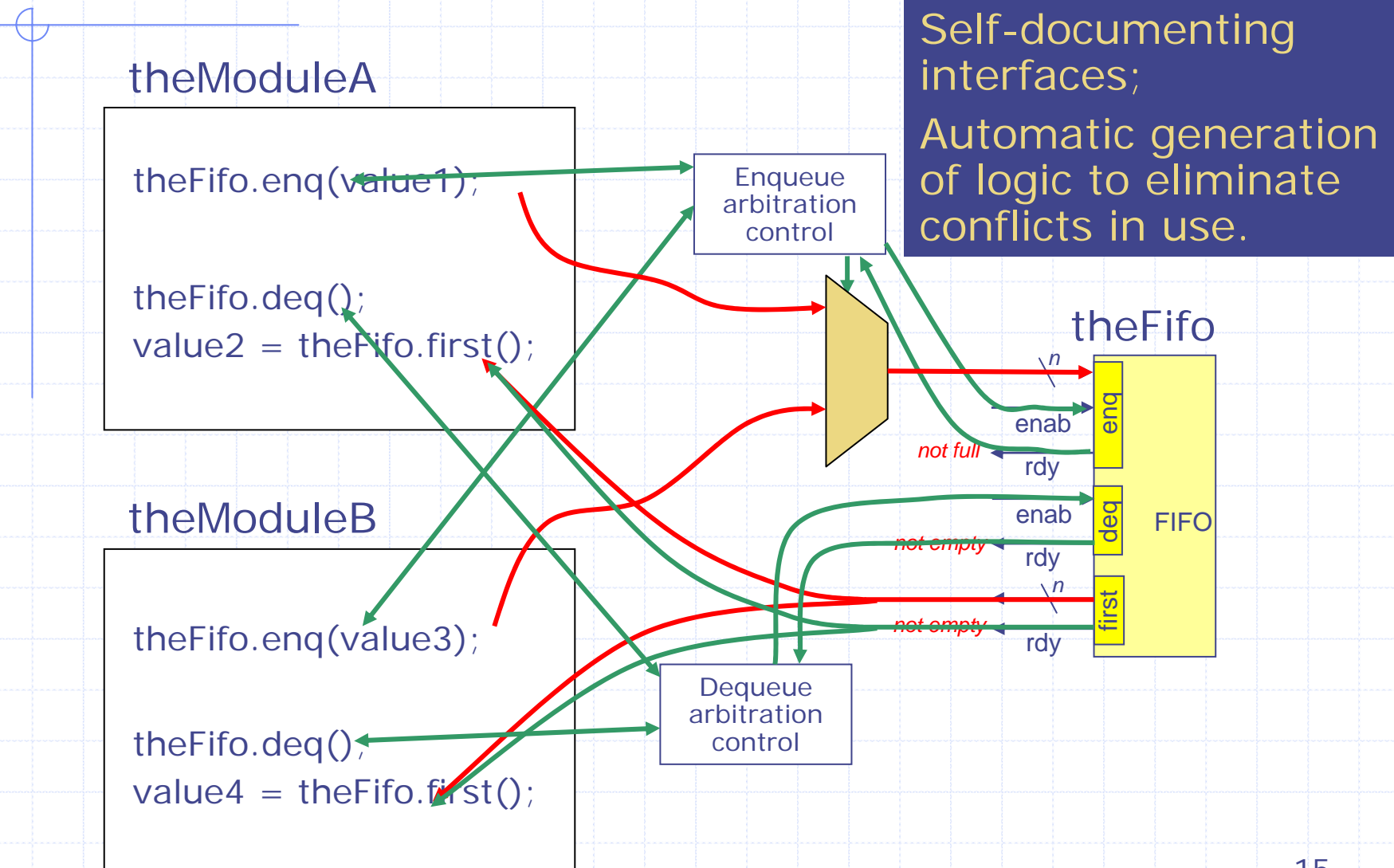
An error occurs if a push is attempted while the FIFO is full.

Thus, there is no conflict in a simultaneous push and pop operation when the FIFO is full. A simultaneous push and pop operation is possible when the FIFO is empty, since there is no pop data to prefetch. However, a push is not allowed when the FIFO is full.

A pop operation is allowed when pop_req_n is asserted (LOW), as long as the FIFO is not empty. pop_req_n causes the internal read pointer to be incremented on the next rising edge of clk. Thus, the RAM read data must be captured on the clk following the assertion of pop_req_n.

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

Bluespec promotes composition through guarded interfaces



Bluespec

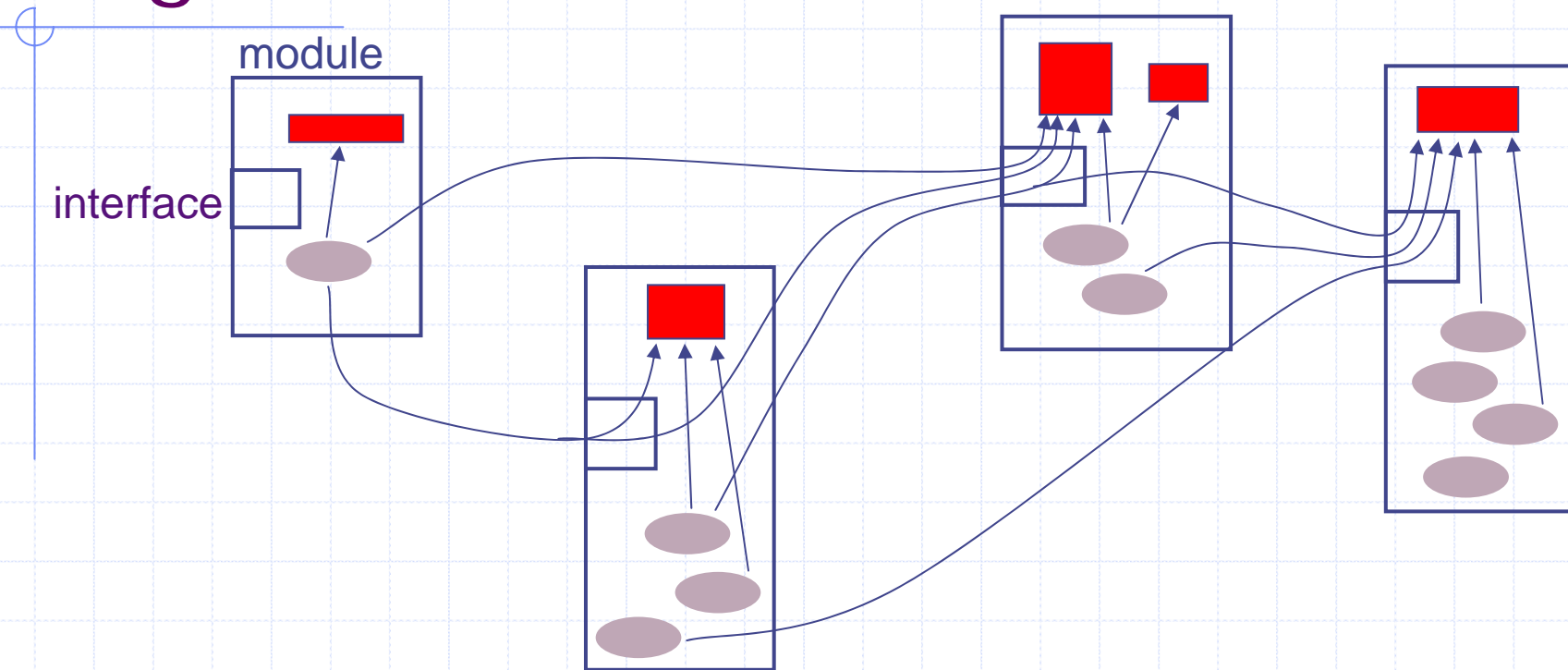
- ◆ What is it?
- ◆ Programming with Rules
 - Example GCD
- ◆ Architectural Exploration
 - Example 802.11a transmitter
- ◆ Example: H.264

Bluespec is available in two versions:

BSV – Bluespec in System Verilog

ESEPro – Bluespec in SystemC

Bluespec: State and Rules organized into *modules*



All *state* (e.g., Registers, FIFOs, RAMs, ...) is explicit.

Behavior is expressed in terms of atomic actions on the state:

Rule: condition → action

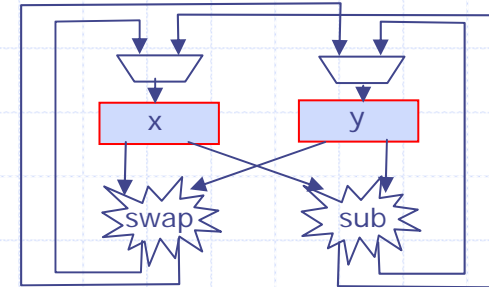
Rules can manipulate state in other modules only *via* their interfaces.

Programming with rules: A simple example

Euclid's algorithm for computing the Greatest Common Divisor (GCD):

15	6	
9	6	<i>subtract</i>
3	6	<i>subtract</i>
6	3	<i>swap</i>
3	3	<i>subtract</i>
0	<i>answer:</i> 3	<i>subtract</i>

GCD in BSV



```
module mkGCD (I_GCD);
```

```
  Reg#(int) x <- mkRegU;  
  Reg#(int) y <- mkReg(0);
```

State

```
rule swap ((x > y) && (y != 0));  
  x <= y; y <= x;
```

```
endrule
```

```
rule subtract ((x <= y) && (y != 0));  
  y <= y - x;
```

```
endrule
```

Internal behavior

```
method Action start(int a, int b) if (y==0);
```

```
  x <= a; y <= b;
```

```
endmethod
```

```
method int result() if (y==0);
```

```
  return x;
```

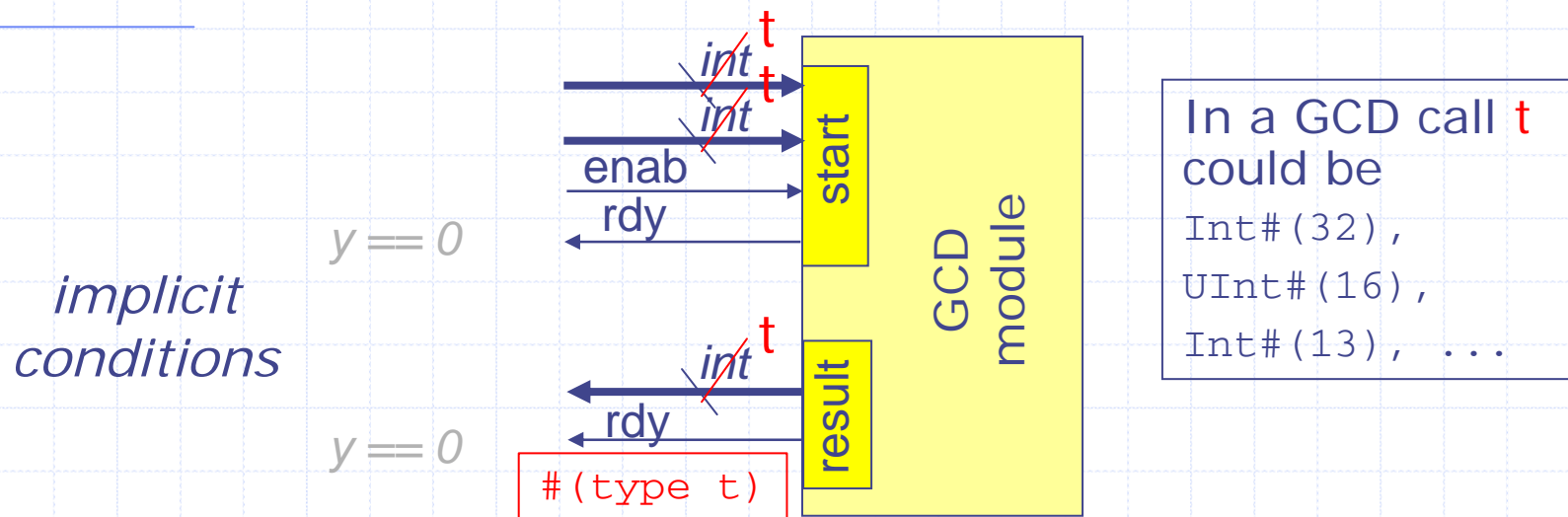
```
endmethod
```

External Interface

```
endmodule
```

Assumes $x \neq 0$ and $y_1 \neq 0$

GCD Hardware Module

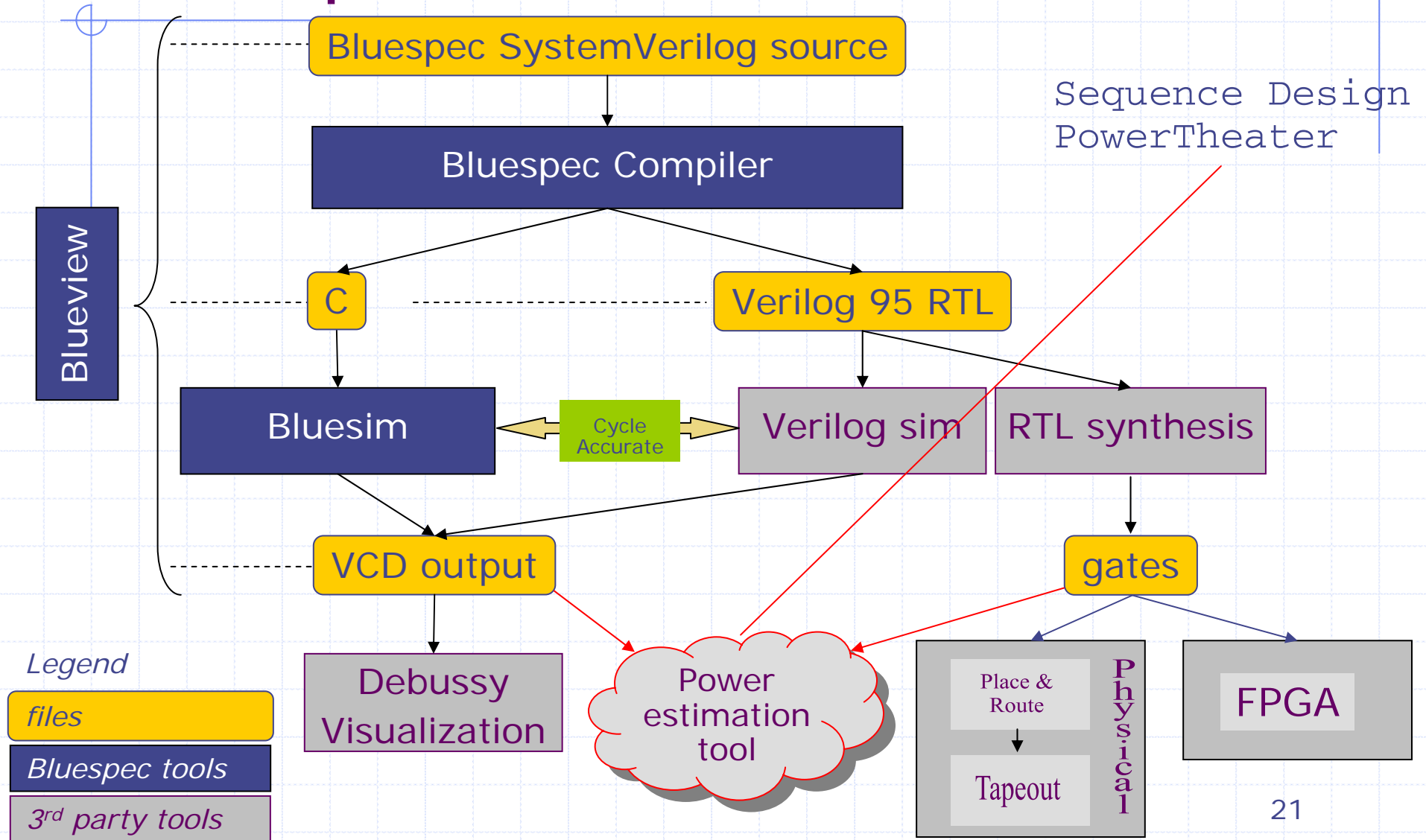


```
interface I_GCD;
    method Action start (intt a, intt b);
    method intt result();
endinterface
```

- ◆ The module can easily be made polymorphic
- ◆ Many different implementations can provide the same interface:


```
module mkGCD (I_GCD)
```

Bluespec Tool flow

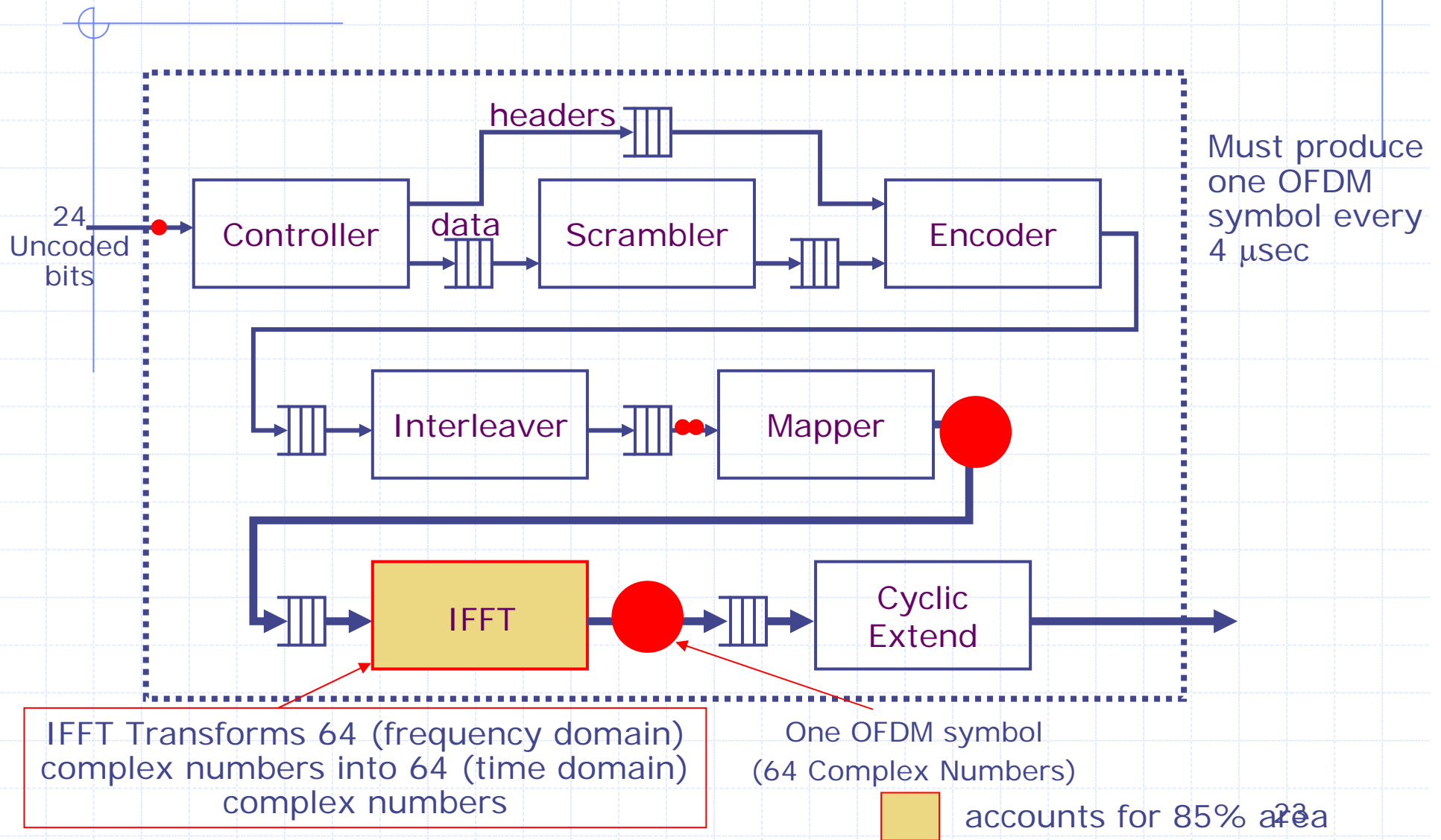




Architectural Exploration:

Area-Power tradeoff in 802.11a transmitter design

802.11a Transmitter Overview



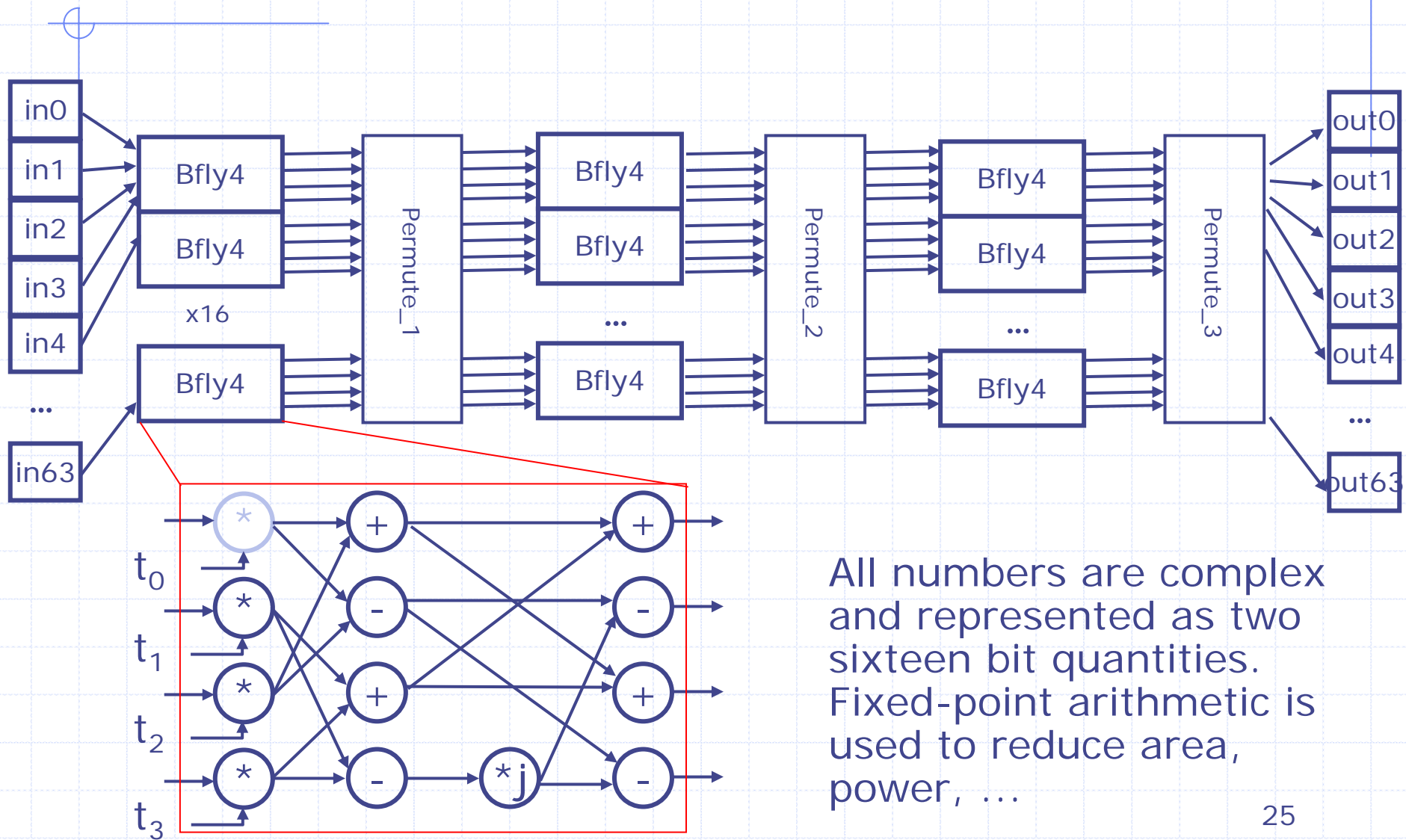
Preliminary results

[MEMOCODE 2006] Dave, Gerding, Pellauer, Arvind

Design Block	Lines of Code (BSV)	Relative Area
Controller	49	0%
Scrambler	40	0%
Conv. Encoder	113	0%
Interleaver	76	1%
Mapper	112	11%
IFFT	95	85%
Cyc. Extender	23	3%

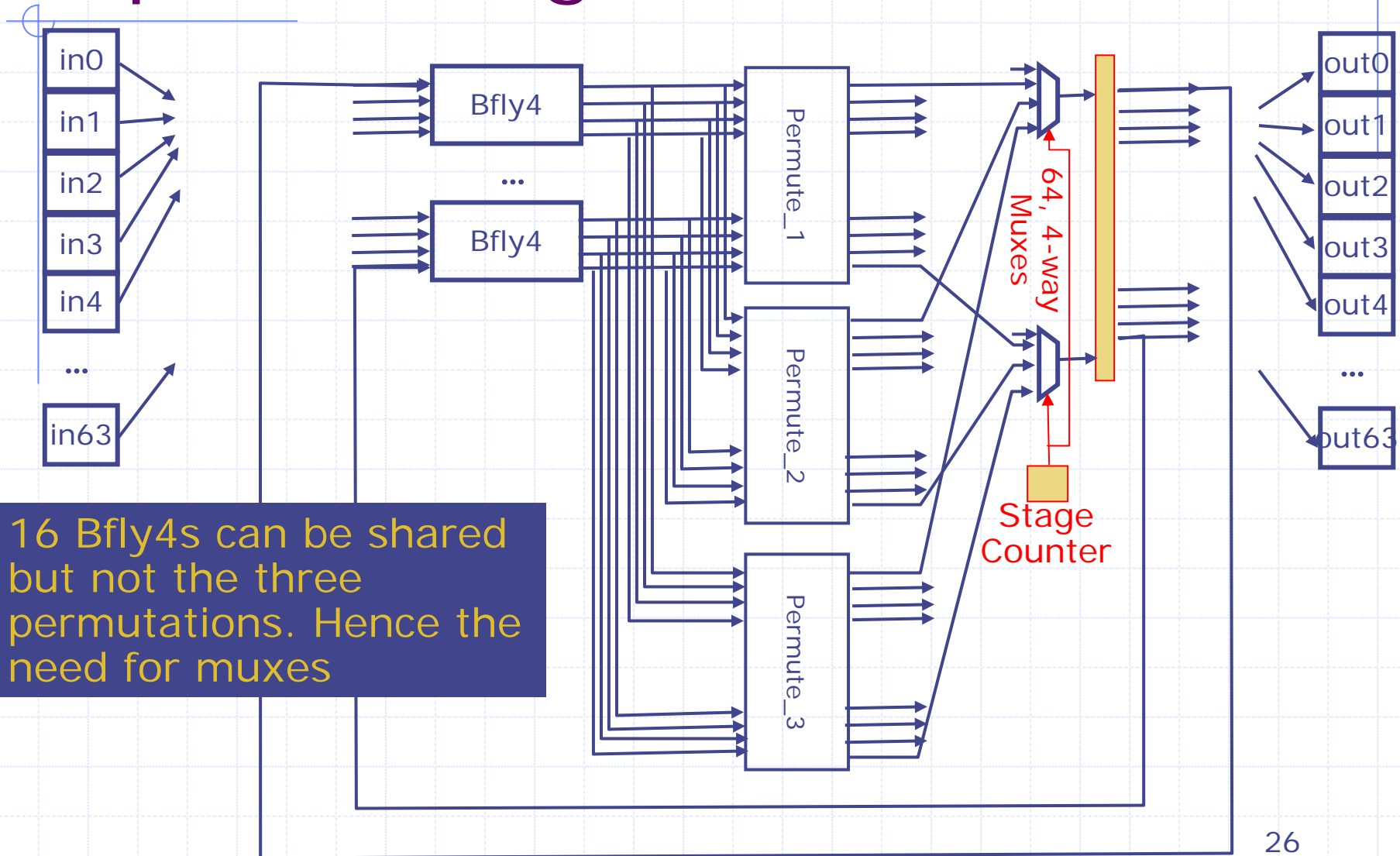
Complex arithmetic libraries constitute another 200 lines of code

Combinational IFFT



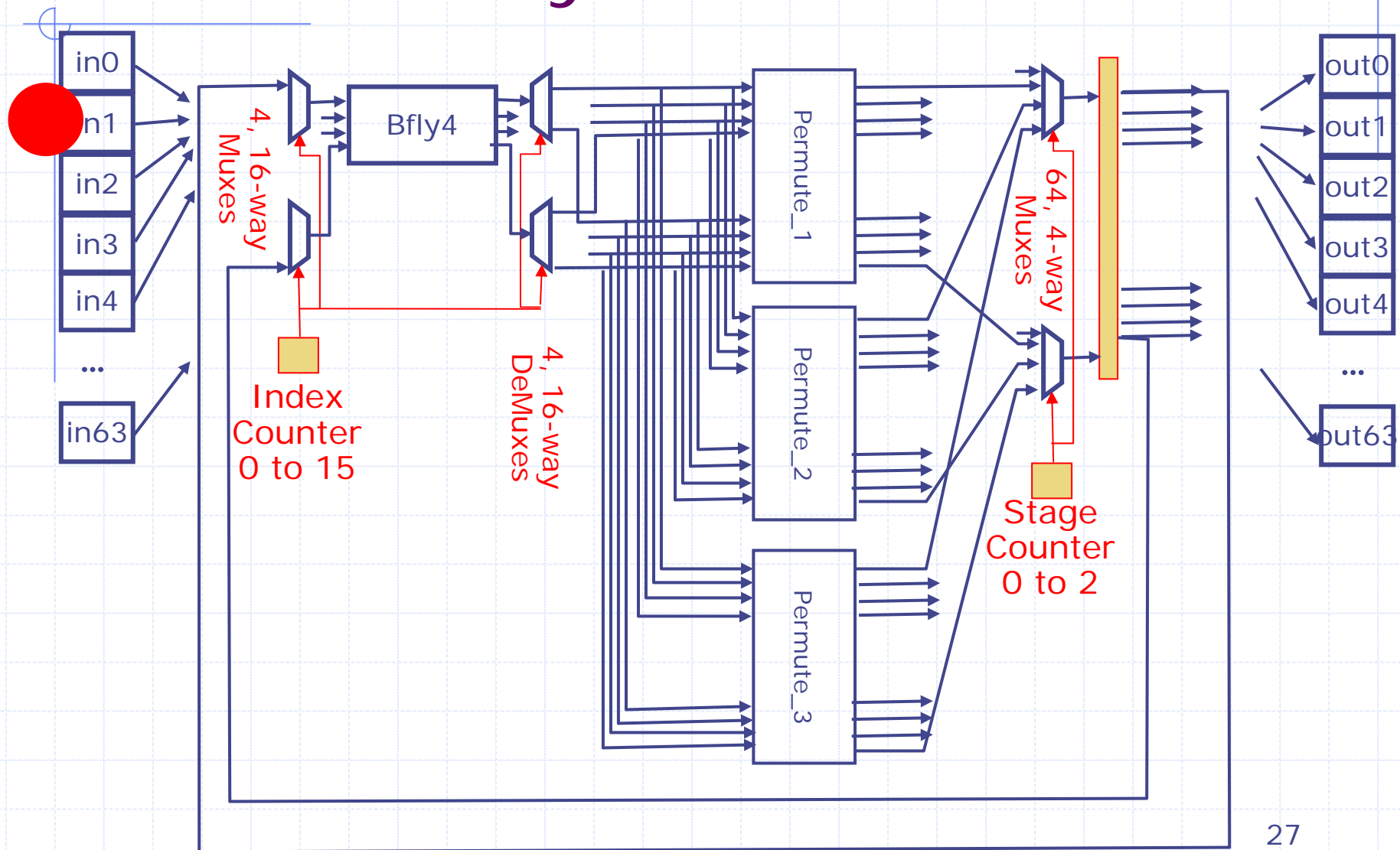
All numbers are complex and represented as two sixteen bit quantities. Fixed-point arithmetic is used to reduce area, power, ...

Circular pipeline: Reusing the Pipeline Stage



16 Bfly4s can be shared but not the three permutations. Hence the need for muxes

Superfolded circular pipeline: Just one Bfly-4 node!



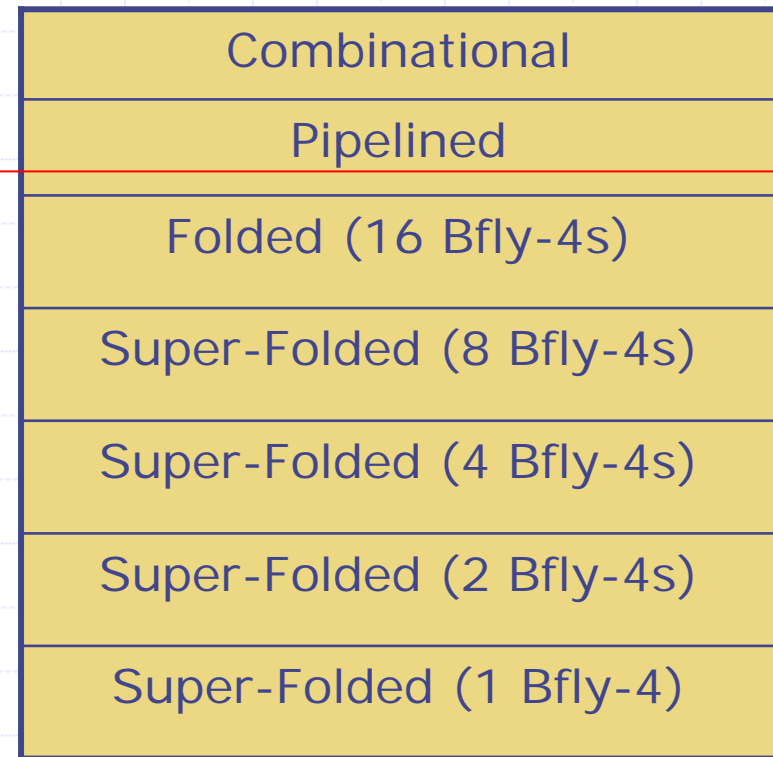
Which design consumes the least energy to transmit a symbol?

- ◆ Can we quickly code up all the alternatives?
 - single source with parameters?

Not practical in traditional hardware description languages like Verilog/VHDL

Expressing these designs in Bluespec was easy

- ◆ All these designs were done in less than *one* day!
 - Designers were experts in Bluespec
- ◆ Area and power estimates?



The same source code

802.11a Transmitter Synthesis results (Only the IFFT block is changing)

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 Power Theater

Note ...

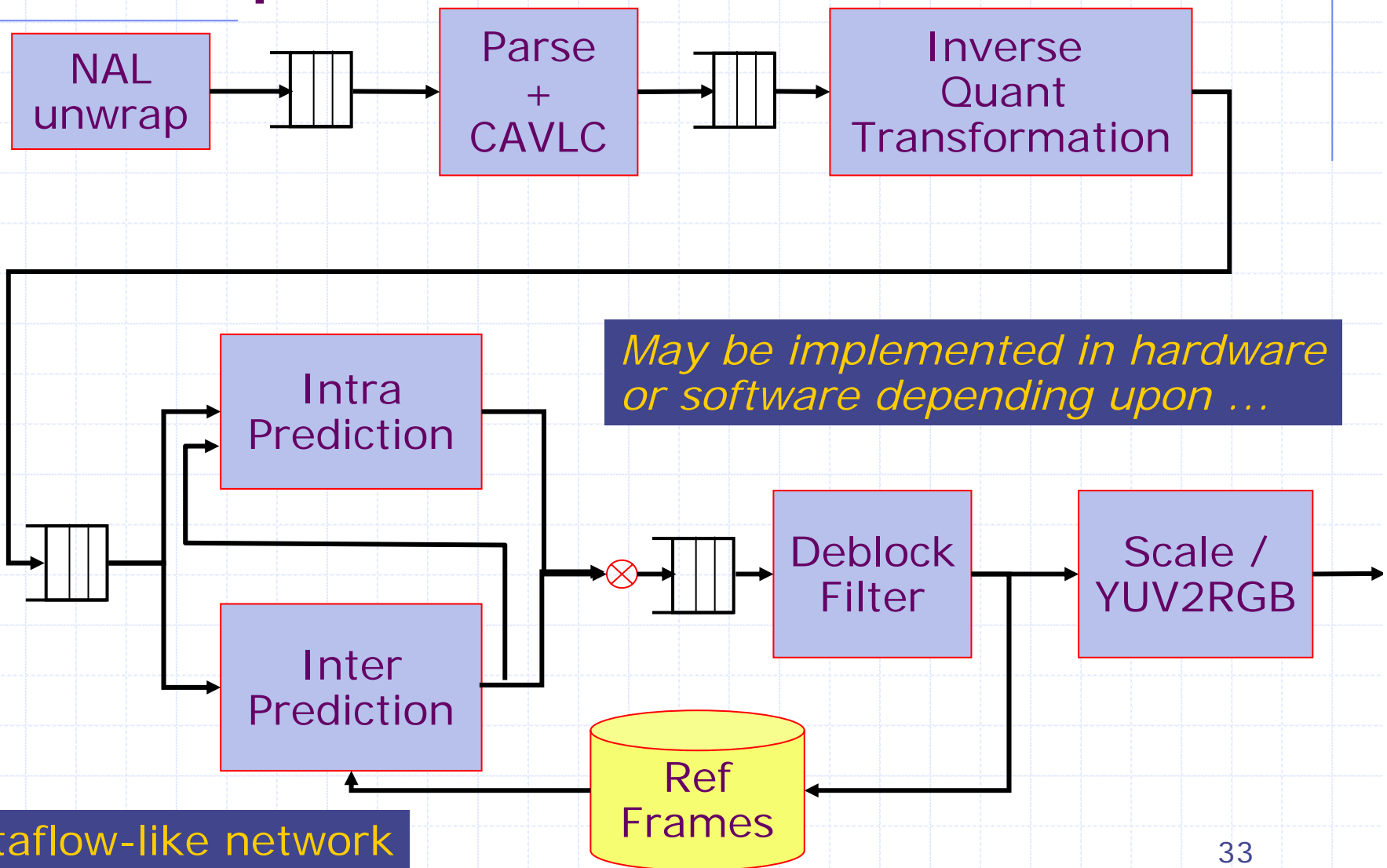
- ◆ Bluespec language and tools have no knowledge of FFT or complex arithmetic ...
- ◆ Bluespec promotes concise implementation descriptions and synthesis, enabling

rapid architectural exploration



Video Codec: H.264

Example: H.264 Decoder



A dataflow-like network

Available codes *(not multithreaded)*

- ◆ Reference code
 - 80K lines, awful coding style, slow
- ◆ ffmpeg code for Linux
 - 200K lines, mixed with other codecs
- ◆ Codes don't reflect the dataflow structure
 - Pointers to data structures are passed around and modified. Difficult to figure out which block is modifying which parts
 - No model of concurrency. Even the streaming aspect gets obscured by the code

The code can be written in a style which will serve both hardware and software communities.

H.264 Decoder in Bluespec

Work in Progress - Chun-Chieh Lin et al

- ◆ Baseline profile, no inter-prediction
- ◆ Decodes 720p @ 55fps
- ◆ 7.3K lines of Bluespec Concise
 - Has been synthesized into hardware (RTL)
 - Any module can be implemented in software
 - Each module can be separately refined
 - Hopefully a much easier source code for multicores

1. Each module embodies its own resources
2. The behaviors are composable in a concurrent setting

Summary

- ◆ Market forces are demanding a much greater variety of SoCs
- ◆ The design cost for SoCs has to be brought down dramatically by facilitating IP reuse
- ◆ High-level synthesis tools are essential for architectural exploration and IP development
- ◆ Bluespec facilitates the new design flow

Thanks