The Zen of SoCs in BSV:
The Threefold Path to SoC Nirvana

Elliot Mednick
Bluespec, Inc

MIT Bluespec Workshop
August 13, 2007

The Problem

- Designing SoCs is hard
- Writing IP is hard
- Changing SoC configurations is hard
- You are suffering

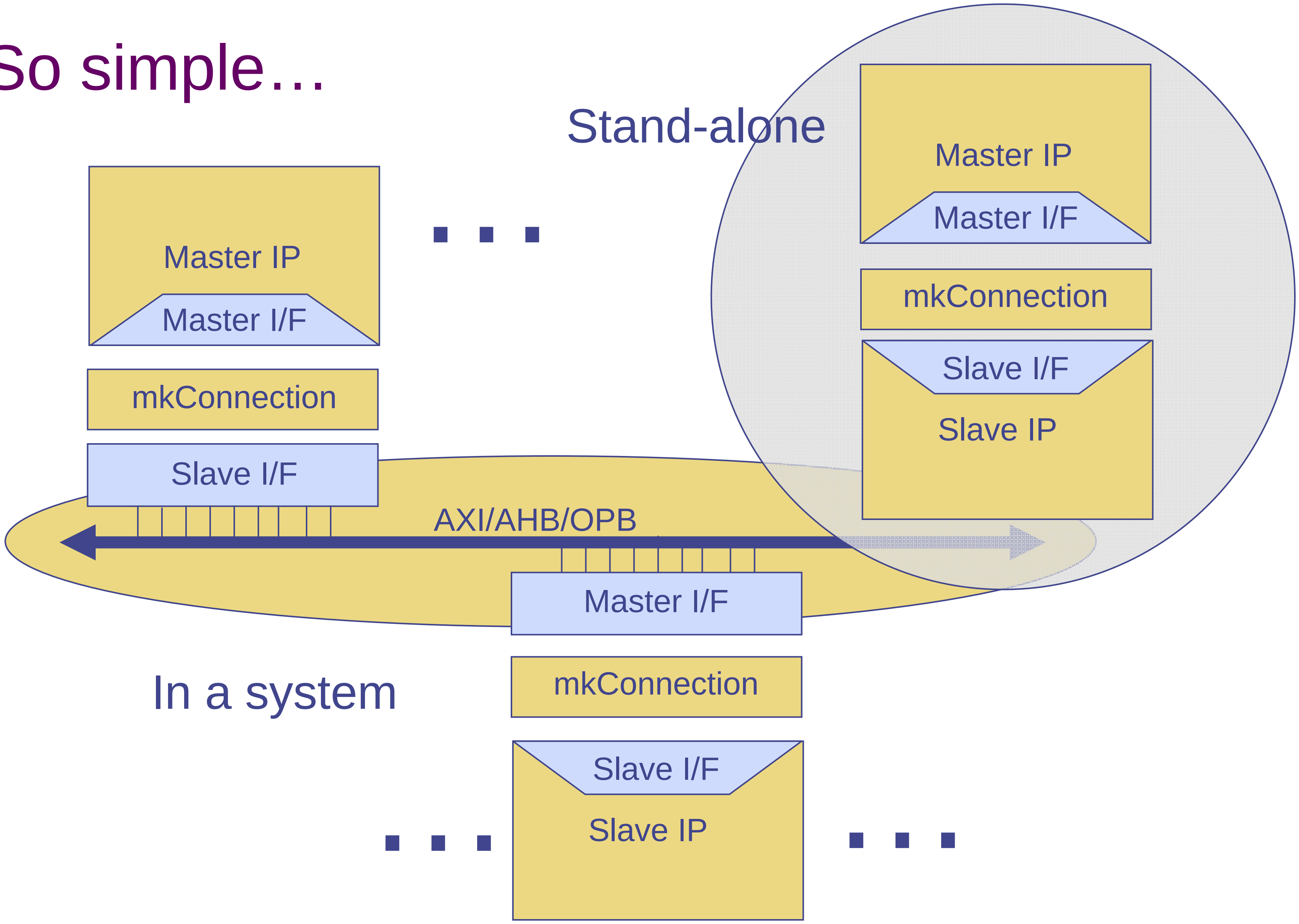
The Threefold Path to SoC Nirvana

1. Right Connections
   - Transactional interfaces
2. Right Abstraction
   - As high as possible, but no higher
3. Right Wrapping
   - Using BSV as a top level, and instantiating Verilog from BSV

The Firstfold Step: Right Interconnects

- All connections through transactional get/put interfaces
  - Master-to-Slave
  - Master-to-fabric, fabric-to-Slave
- Allows nearly trivial IP *and interconnect(!)* swaps
- Already in AzureIP library
- Currently supports AXI, AHB buses
- OPB to be added next
- Synthesizability unique to BSV

So simple…


Stand-alone
Master IP
Master I/F
mkConnection
Slave I/F
Slave IP
Master IP
Master I/F
mkConnection
Slave I/F
. . .
AXI/AHB/OPB
Master I/F
mkConnection
Slave I/F
Slave IP
In a system
. . .
. . .

The Secondfold Step: Right Abstraction

- Use a mix of abstractions, where each is the highest (reasonable) level of abstraction
- Most here familiar with Bluespec implementations – and how high-level they are relative to RTL
- With this effort, also show synthesizable models:
  - Write IP quickly, as if it were a “C” model
  - Ignore (mostly) hardware performance optimizations, like pipelining
  - For example: PowerPC “synthesizable ISS”
    - One rule per instruction, mostly
    - Very fast Bluesim simulation speed (1-10MIPs)
    - “Good Enough” hardware performance (100MIPs)
    - “Good Enough” area (<1000 LUTs)
    - Easy to implement
  - A test: PCI-E
    - Is there a subset that will work on a PC host that is fast enough to meet timing using high-abstraction techniques?
    - We will find out
  - Synthesizability using to BSV

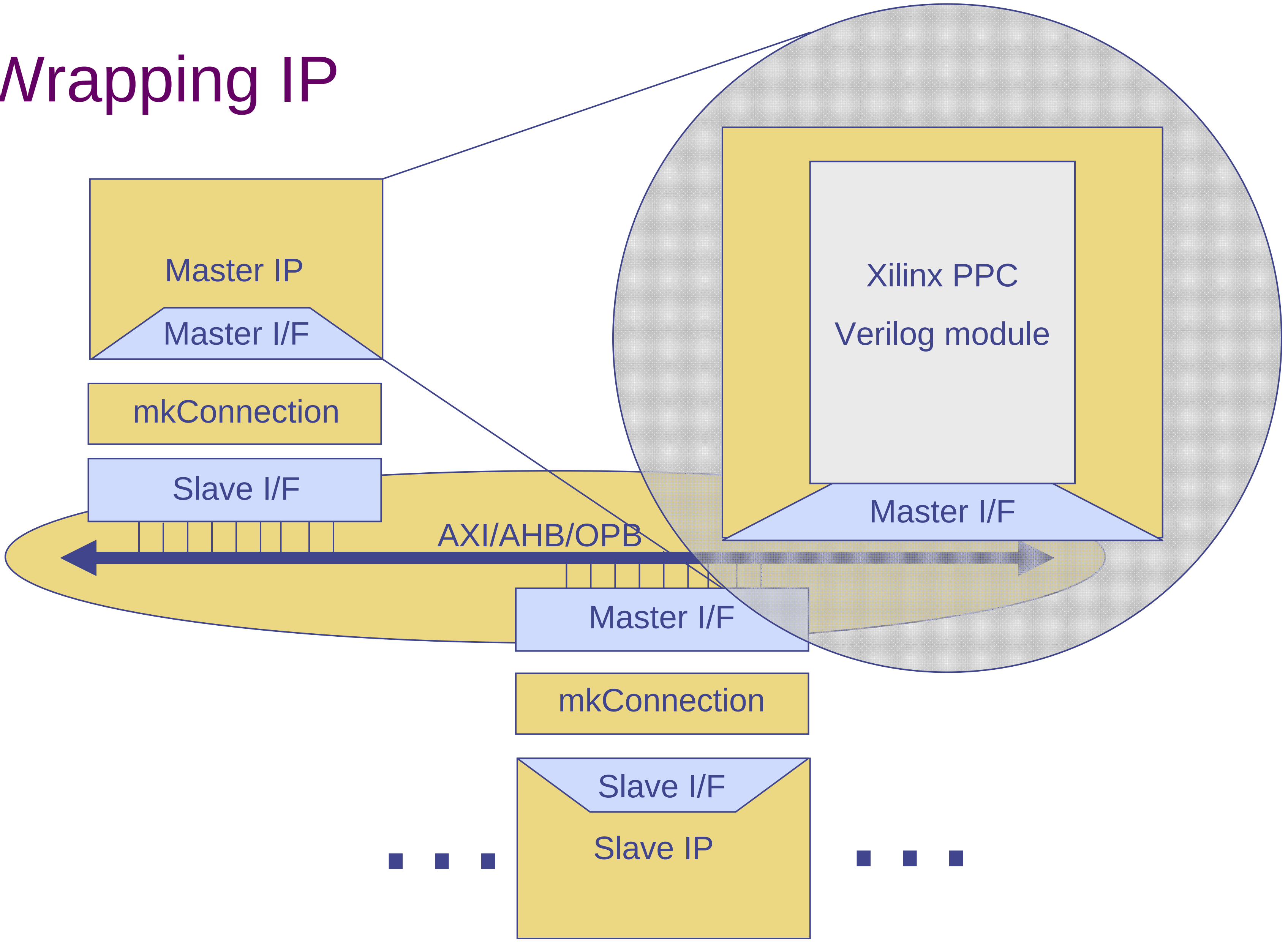
Sample code snippet

```
rule compare_log_imm(check_opcode(pc)==Cmpli);
   …
endrule

rule cntlzw (check_opcode(pc) == Cntlzw);
   …
endrule

rule cror (check_opcode(pc) == Cror);
   …
endrule

rule crxor (check_opcode(pc) == Crxor);
   …
endrule
```

The Thirdfold Step: Right Wrapping

- Use ImportBVI to access legacy Verilog models from BSV
- First test: Wrap Xilinx’s PPC to connect it to AzureIP interfaces
- Second test: Wrap Xilinx’s Ethernet MAC
- Also need to wrap Xilinx-specific libraries
    - Distributed RAM Blocks
    - Queues
    - MPMC2?
- Later: Develop wrapping methodology
- A little more later: write a tool to do it, *a la* SWIG
- Synthesizability unique to BSV

Wrapping IP

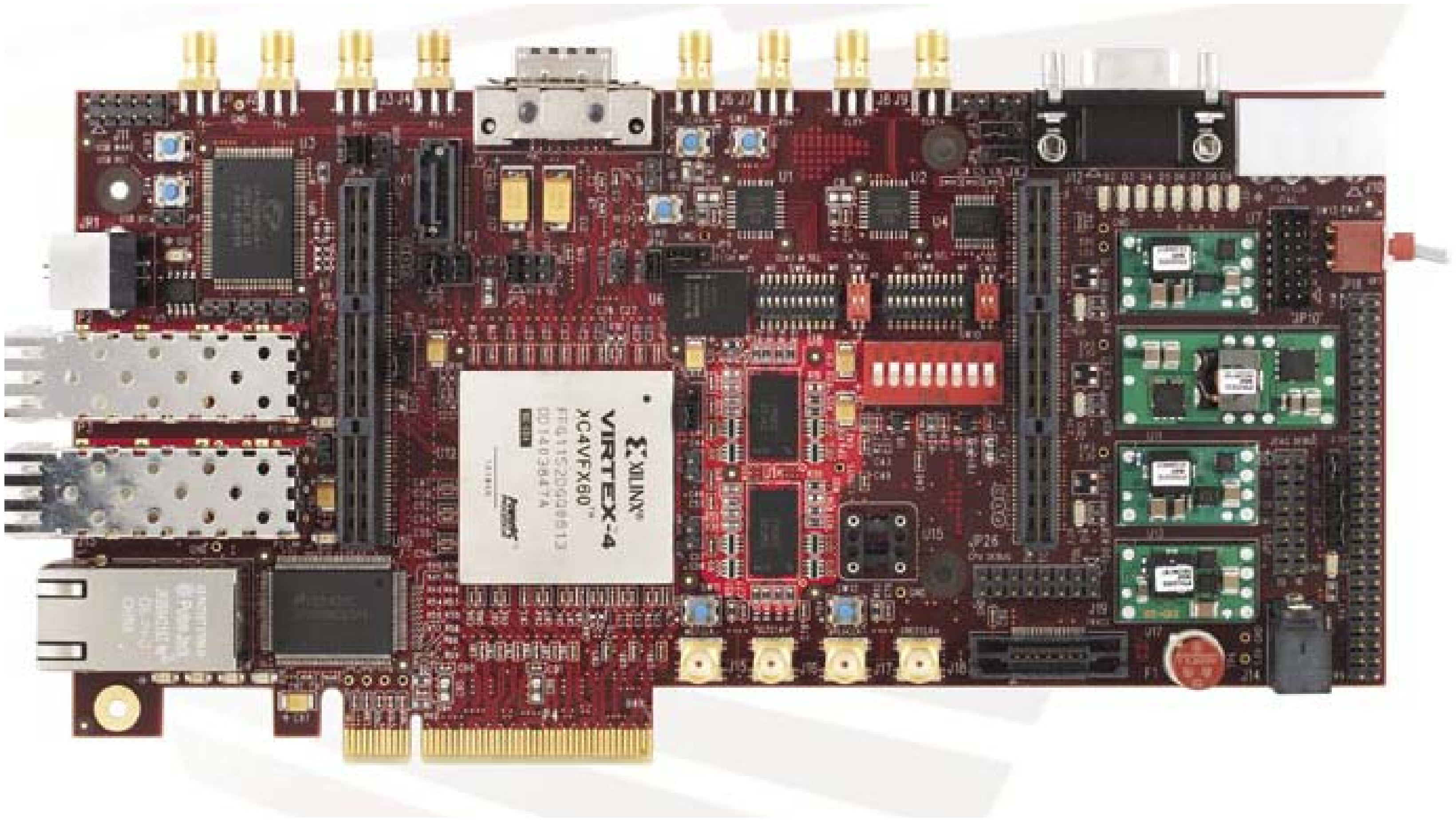

Master IP
Master I/F
mkConnection
Slave I/F
AXI/AHB/OPB
Xilinx PPC
Verilog module
Master I/F
Master I/F
mkConnection
Slave I/F
Slave IP
. . .
. . .

The Demonstration Vehicle

- Substrate: Xilinx Virtex-4 FX (100)
- Processor: PowerPC 405
    - Xilinx and Bluespec/BSV versions
- OS: Linux 2.6.x
- Application 1: FTP server
    - IP: Ethernet MAC, flash/SDRAM memory, interconnect
    - AzureIP interconects
    - Xilinx MAC
- Application 2: Digital Video Recorder
    - Additional IP: H.264, PCI-Express
        - BSV H.264 from MIT
        - BSV PCI-E by Bluespec
- SoC Nirvana

The Platform

- Avnet PCI-E Development Board



Current Status

- Just starting
- FTP Server by Q4/07
- DVR basics by Q1/08
- Nirvana by tbd

Contact info

Elliot Mednick

elliot@bluespec.com