Modular Design in Bluespec Using Asim / AWB

Joel Emer[‡†], Michael Adler[‡], Michael Pellauer[‡†]

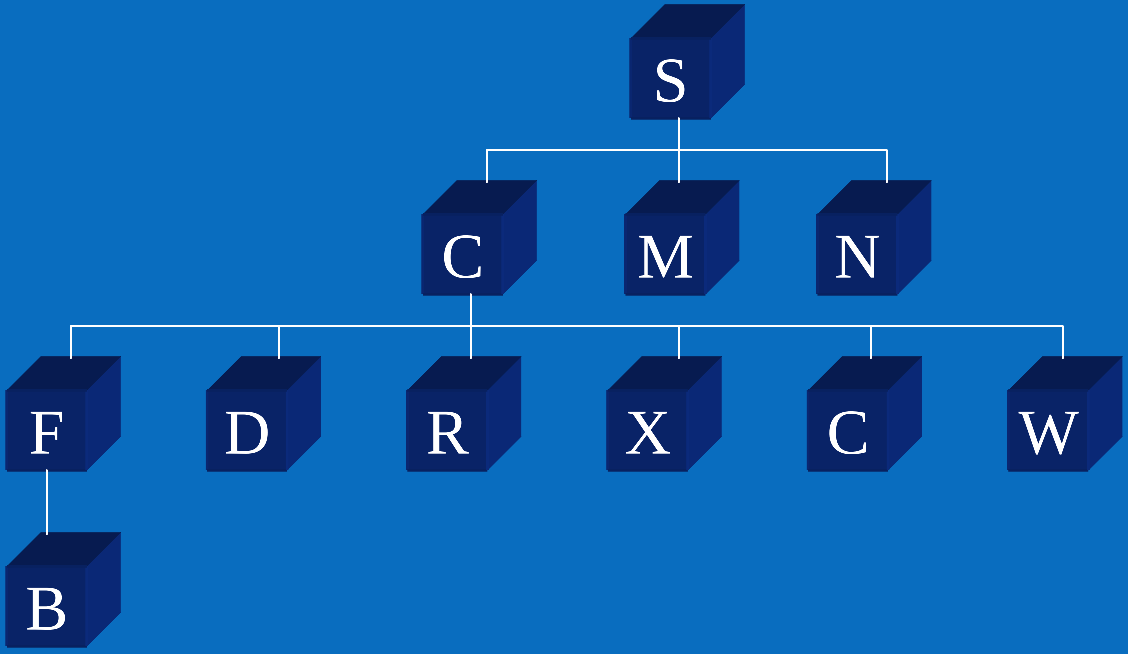
[‡]VSSAD Group
Intel

[†]CSG - CSAIL
MIT

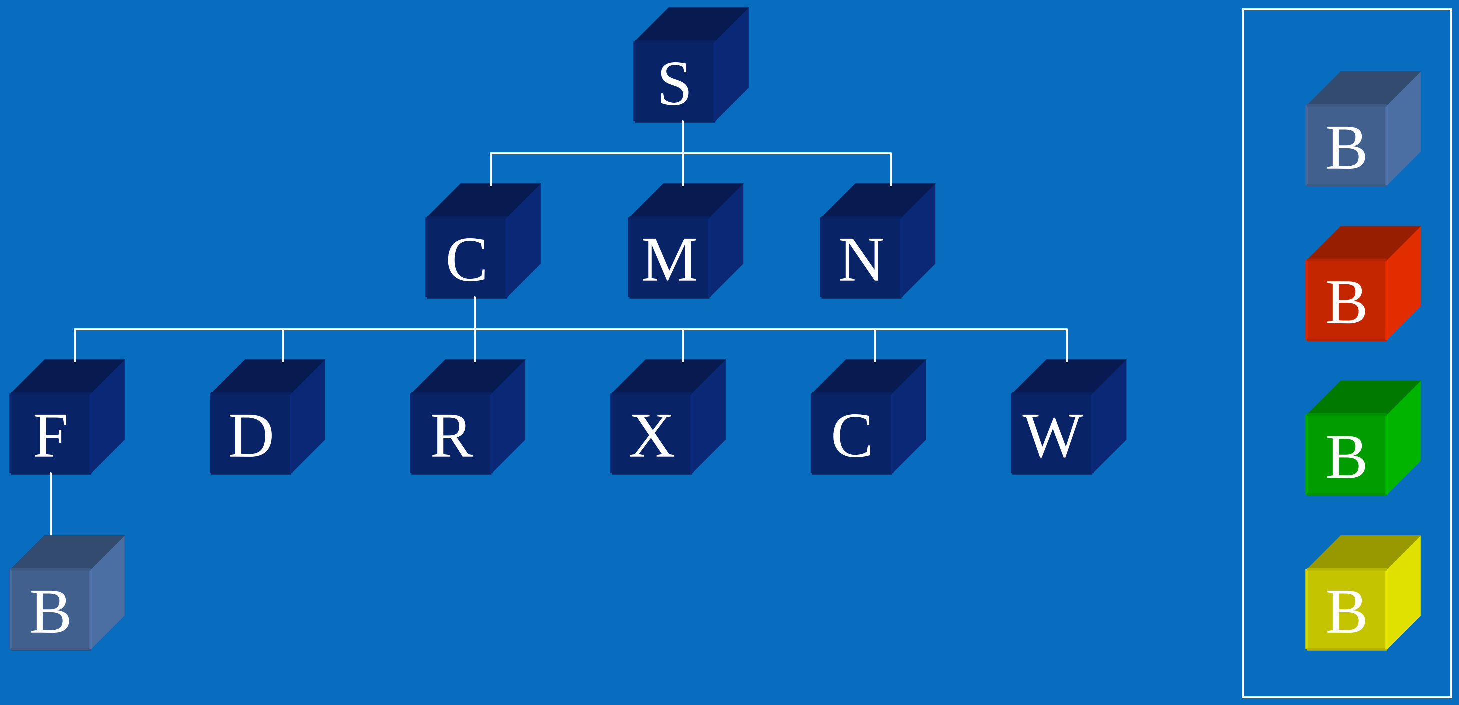
Why modularity?

- Speed of development
- Shared components between products
- Reuse across generations
- Improved fidelity
- Incremental refinement
- Facilitates area/speed trade-offs
- Architectural experimentation
- Factorial development and evaluations
- Sharing

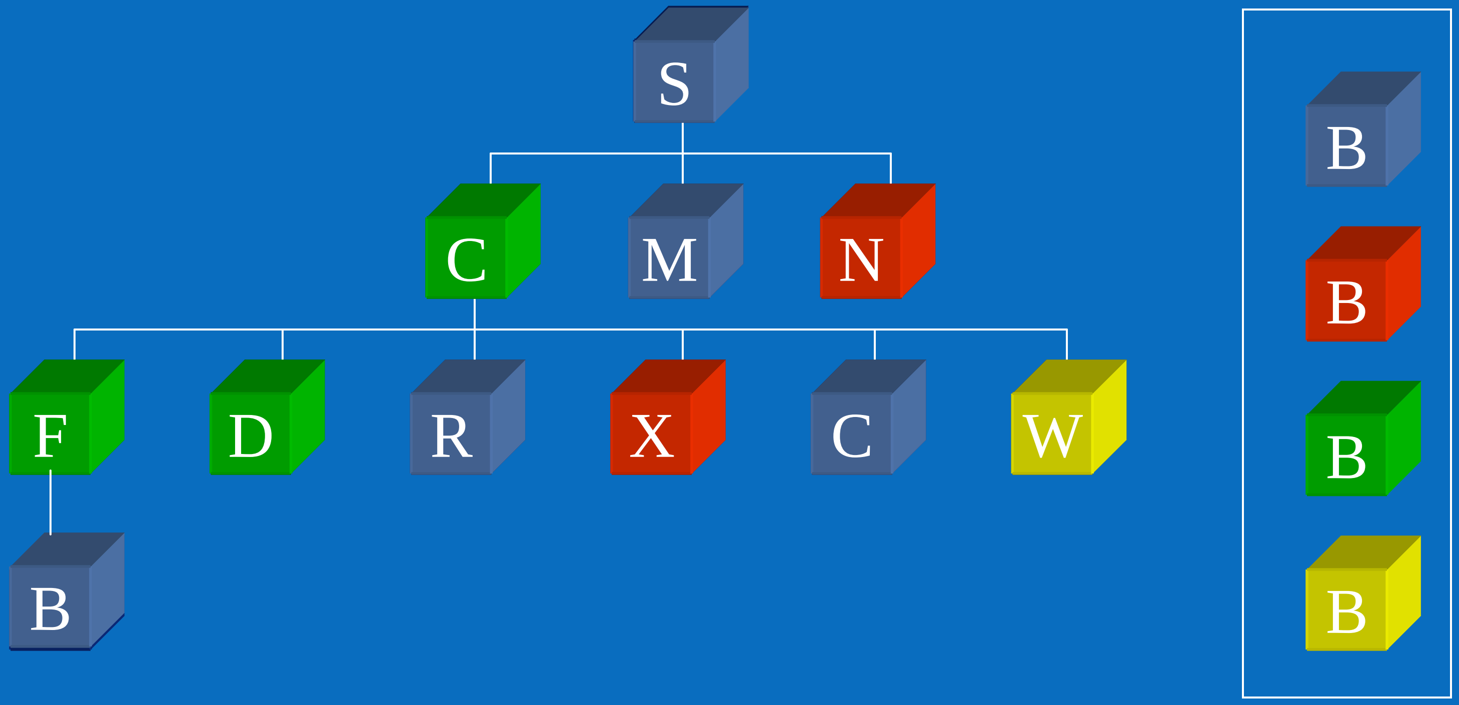
ASIM Module Hierarchy



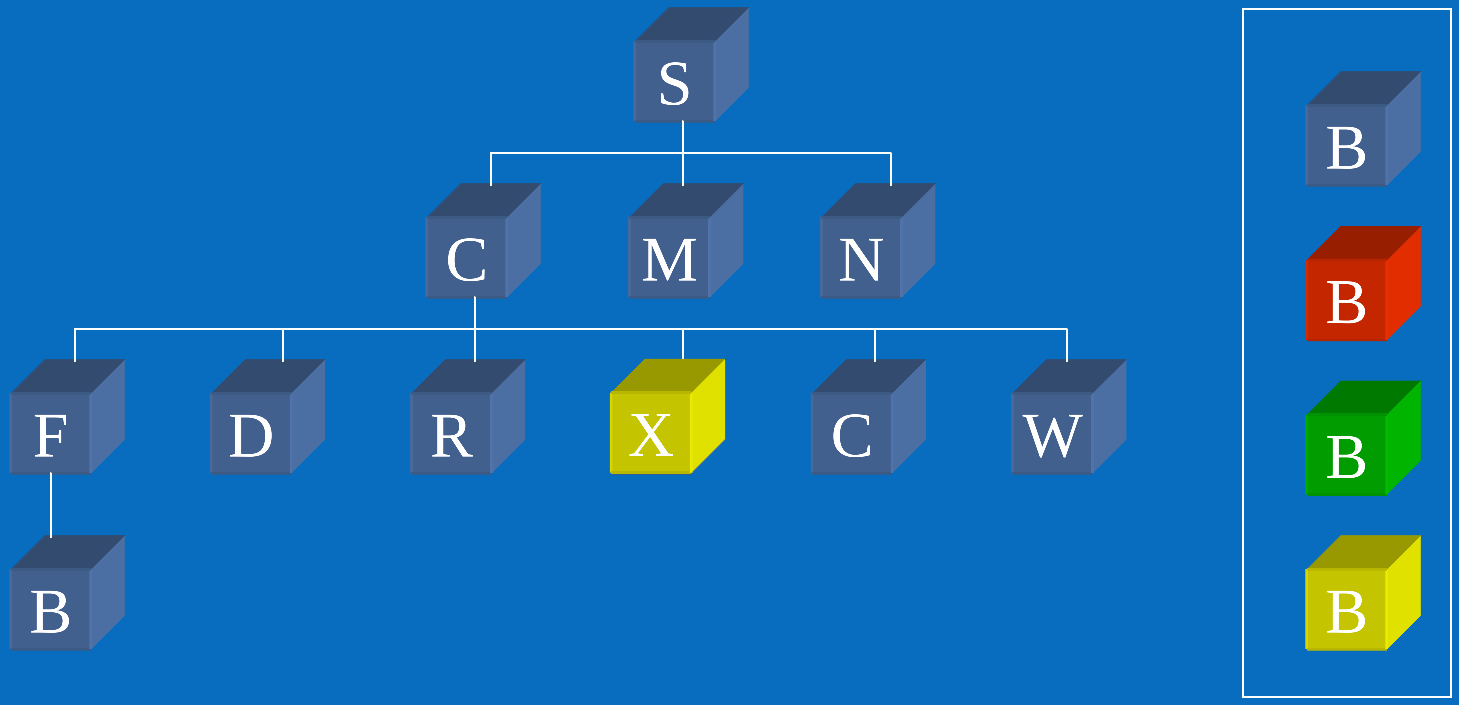
ASIM Module Selection



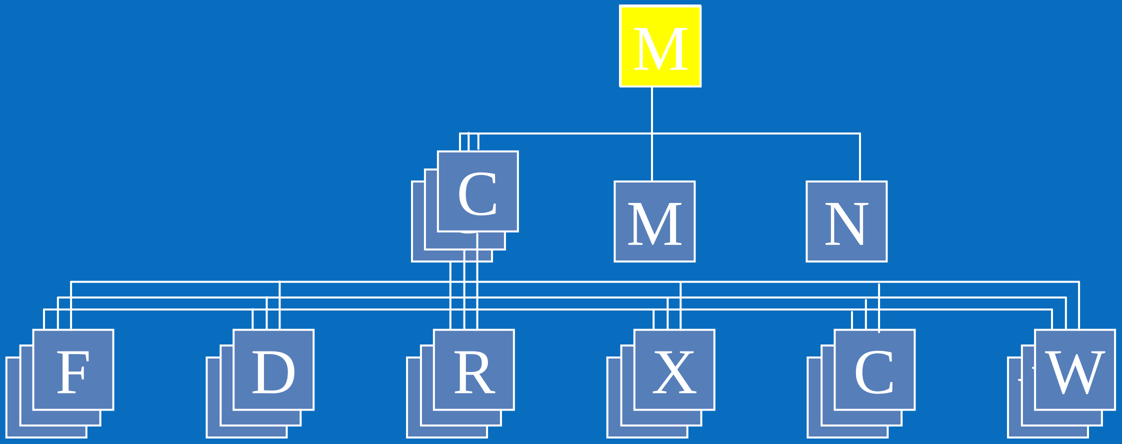
Module Selection



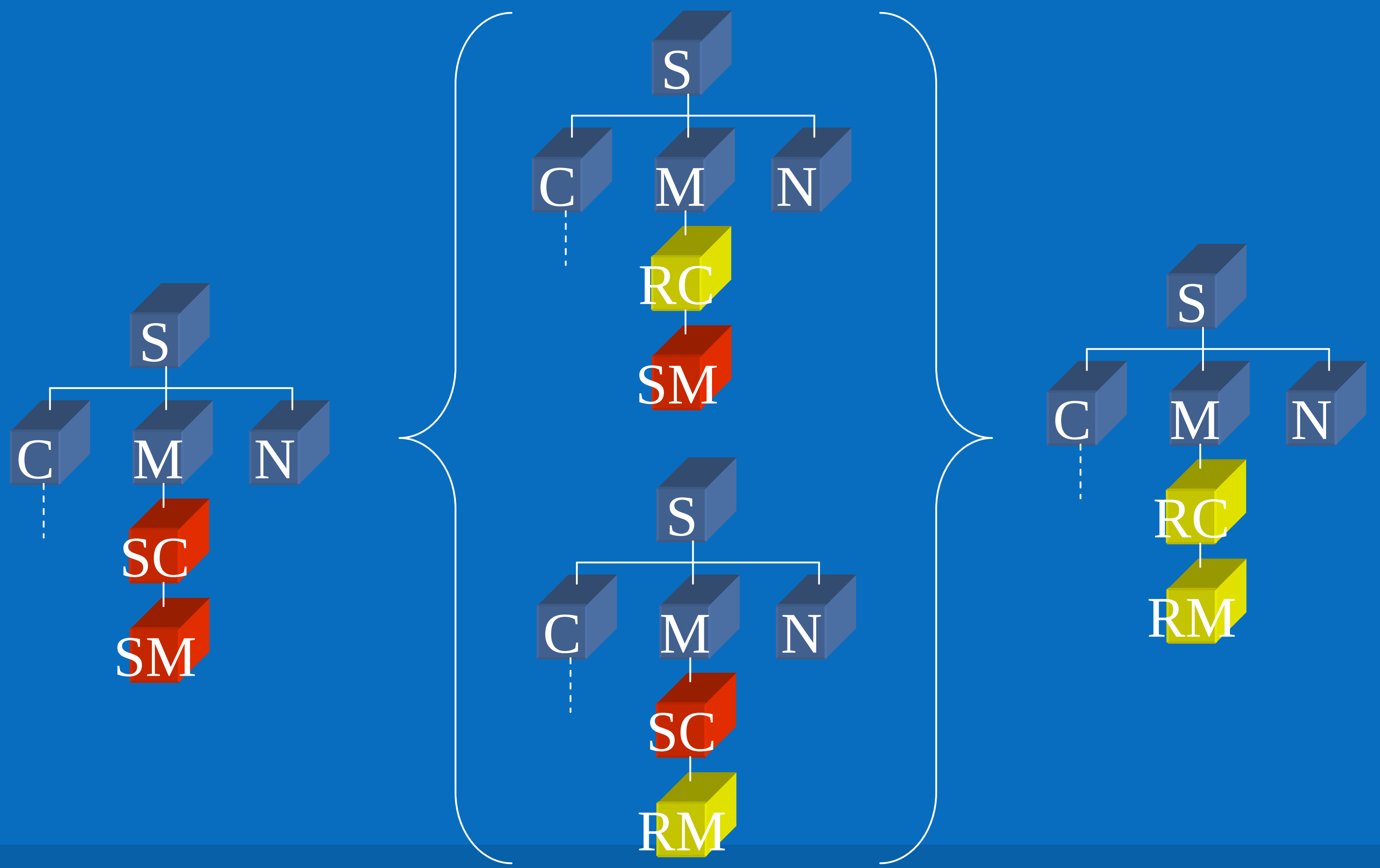
Module Replacement



Module Instantiation



Factorial Coding / Experiments



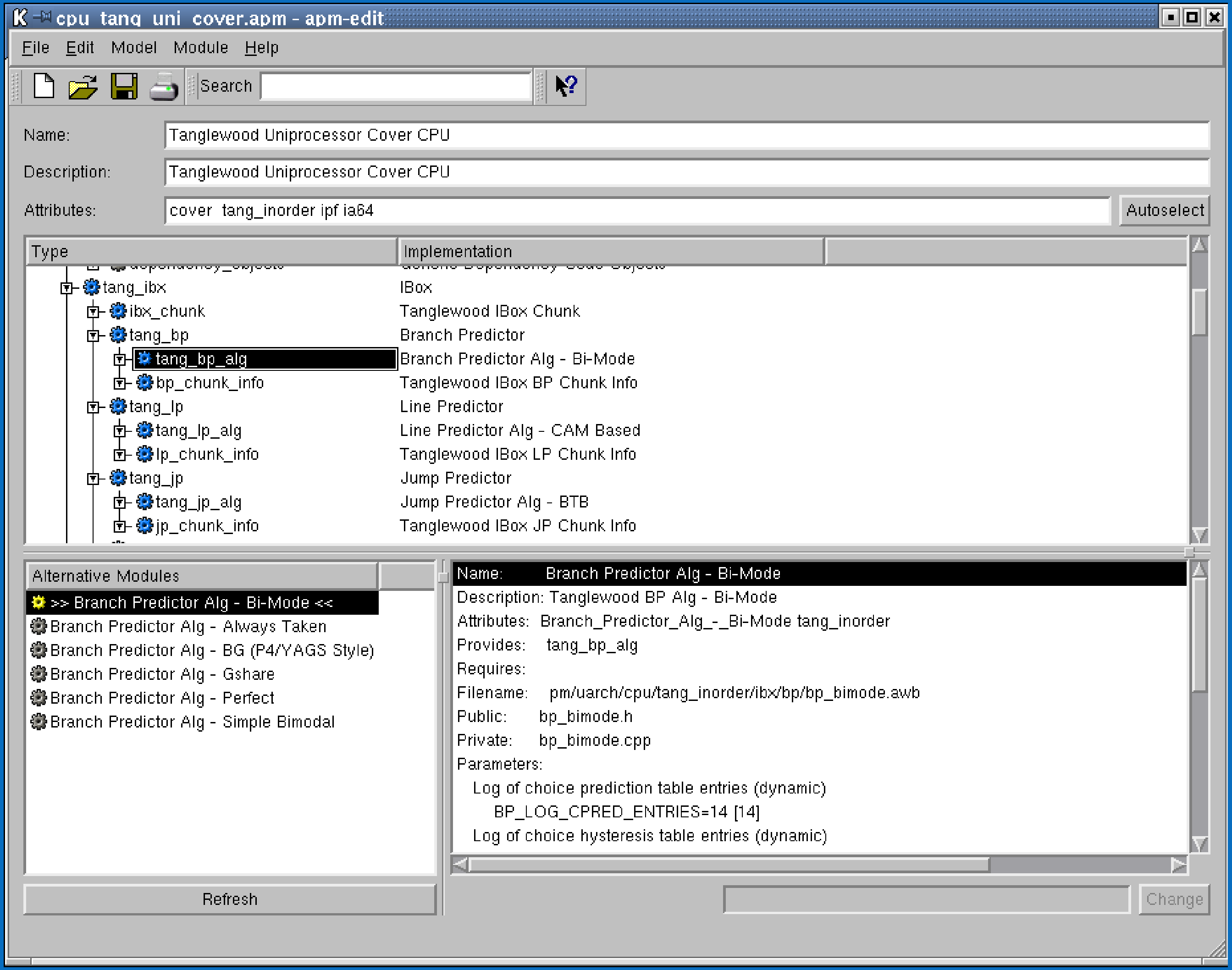
Module Description (.awb file)

```
%name SMIPS R10K Superscalar Decode Stage
%desc SMIPS R10K Superscalar Decode Stage
%attributes s10k smips hasim

%provides hasim_pipe_decode
%requires hasim_rob hasim_branch_pred

%public Decode.bsv
```

(H)ASIM Module Hierarchy


cpu_tang_uni_cover.apm - apm-edit
File Edit Model Module Help
Search

Name: Tanglewood Uniprocessor Cover CPU

Description: Tanglewood Uniprocessor Cover CPU

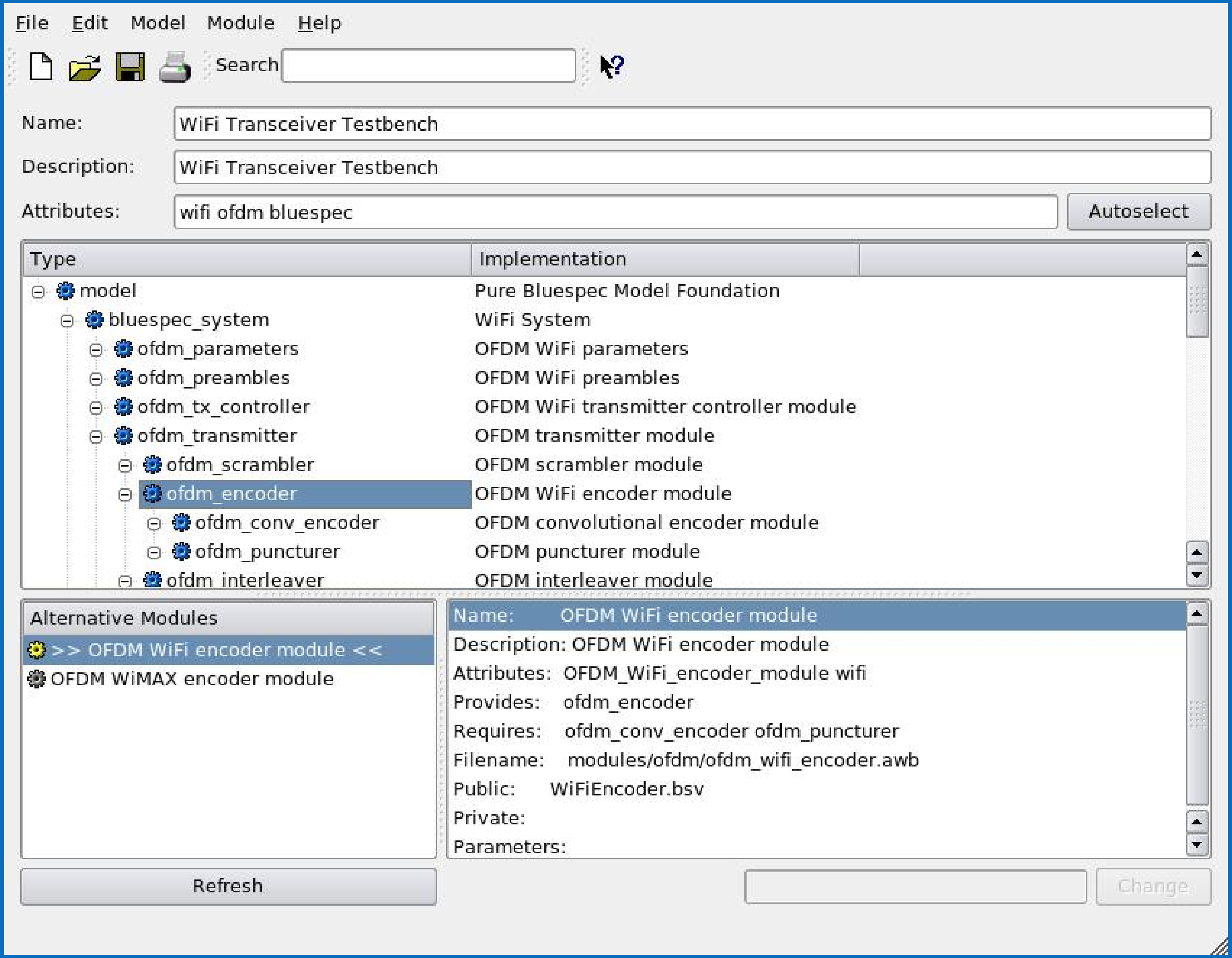
Attributes: cover tang_inorder ipf ia64

Type	Implementation
tang_ibx	IBox
ibx_chunk	Tanglewood IBox Chunk
tang_bp	Branch Predictor
tang_bp_alg	Branch Predictor Alg - Bi-Mode
bp_chunk_info	Tanglewood IBox BP Chunk Info
tang_lp	Line Predictor
tang_lp_alg	Line Predictor Alg - CAM Based
lp_chunk_info	Tanglewood IBox LP Chunk Info
tang_jp	Jump Predictor
tang_jp_alg	Jump Predictor Alg - BTB
jp_chunk_info	Tanglewood IBox JP Chunk Info

Alternative Modules
- >> Branch Predictor Alg - Bi-Mode <<
- Branch Predictor Alg - Always Taken
- Branch Predictor Alg - BG (P4/YAGS Style)
- Branch Predictor Alg - Gshare
- Branch Predictor Alg - Perfect
- Branch Predictor Alg - Simple Bimodal

```
Name:        Branch Predictor Alg - Bi-Mode
Description: Tanglewood BP Alg - Bi-Mode
Attributes:  Branch_Predictor_Alg_-_Bi-Mode tang_inorder
Provides:    tang_bp_alg
Requires:
Filename:    pm/uarch/cpu/tang_inorder/ibx/bp/bp_bimode.awb
Public:      bp_bimode.h
Private:     bp_bimode.cpp
Parameters:
  Log of choice prediction table entries (dynamic)
     BP_LOG_CPRED_ENTRIES=14 [14]
  Log of choice hysteresis table entries (dynamic)
```

(H)ASIM Module Hierarchy



Name: WiFi Transceiver Testbench

Description: WiFi Transceiver Testbench

Attributes: wifi ofdm bluespec

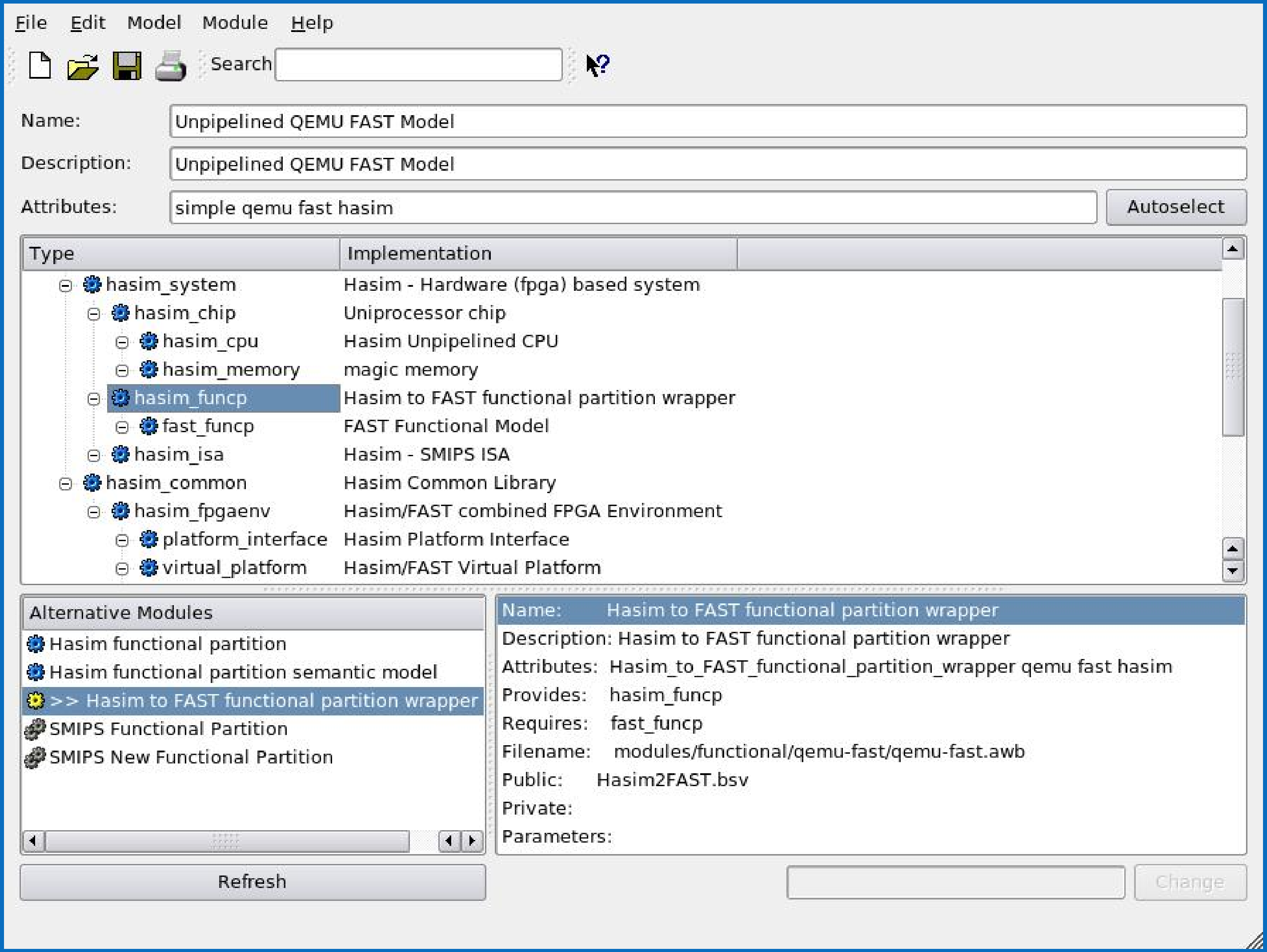
Type	Implementation
model	Pure Bluespec Model Foundation
bluespec_system	WiFi System
ofdm_parameters	OFDM WiFi parameters
ofdm_preambles	OFDM WiFi preambles
ofdm_tx_controller	OFDM WiFi transmitter controller module
ofdm_transmitter	OFDM transmitter module
ofdm_scrambler	OFDM scrambler module
ofdm_encoder	OFDM WiFi encoder module
ofdm_conv_encoder	OFDM convolutional encoder module
ofdm_puncturer	OFDM puncturer module
ofdm_interleaver	OFDM interleaver module

Alternative Modules

- >> OFDM WiFi encoder module <<
- OFDM WiMAX encoder module

Name:	OFDM WiFi encoder module
Description:	OFDM WiFi encoder module
Attributes:	OFDM_WiFi_encoder_module wifi
Provides:	ofdm_encoder
Requires:	ofdm_conv_encoder ofdm_puncturer
Filename:	modules/ofdm/ofdm_wifi_encoder.awb
Public:	WiFiEncoder.bsv
Private:	
Parameters:	

(H)ASIM Module Hierarchy



Name: Unpipelined QEMU FAST Model
Description: Unpipelined QEMU FAST Model
Attributes: simple qemu fast hasim

Type	Implementation
hasim_system	Hasim - Hardware (fpga) based system
hasim_chip	Uniprocessor chip
hasim_cpu	Hasim Unpipelined CPU
hasim_memory	magic memory
hasim_funcp	Hasim to FAST functional partition wrapper
fast_funcp	FAST Functional Model
hasim_isa	Hasim - SMIPS ISA
hasim_common	Hasim Common Library
hasim_fpgaenv	Hasim/FAST combined FPGA Environment
platform_interface	Hasim Platform Interface
virtual_platform	Hasim/FAST Virtual Platform

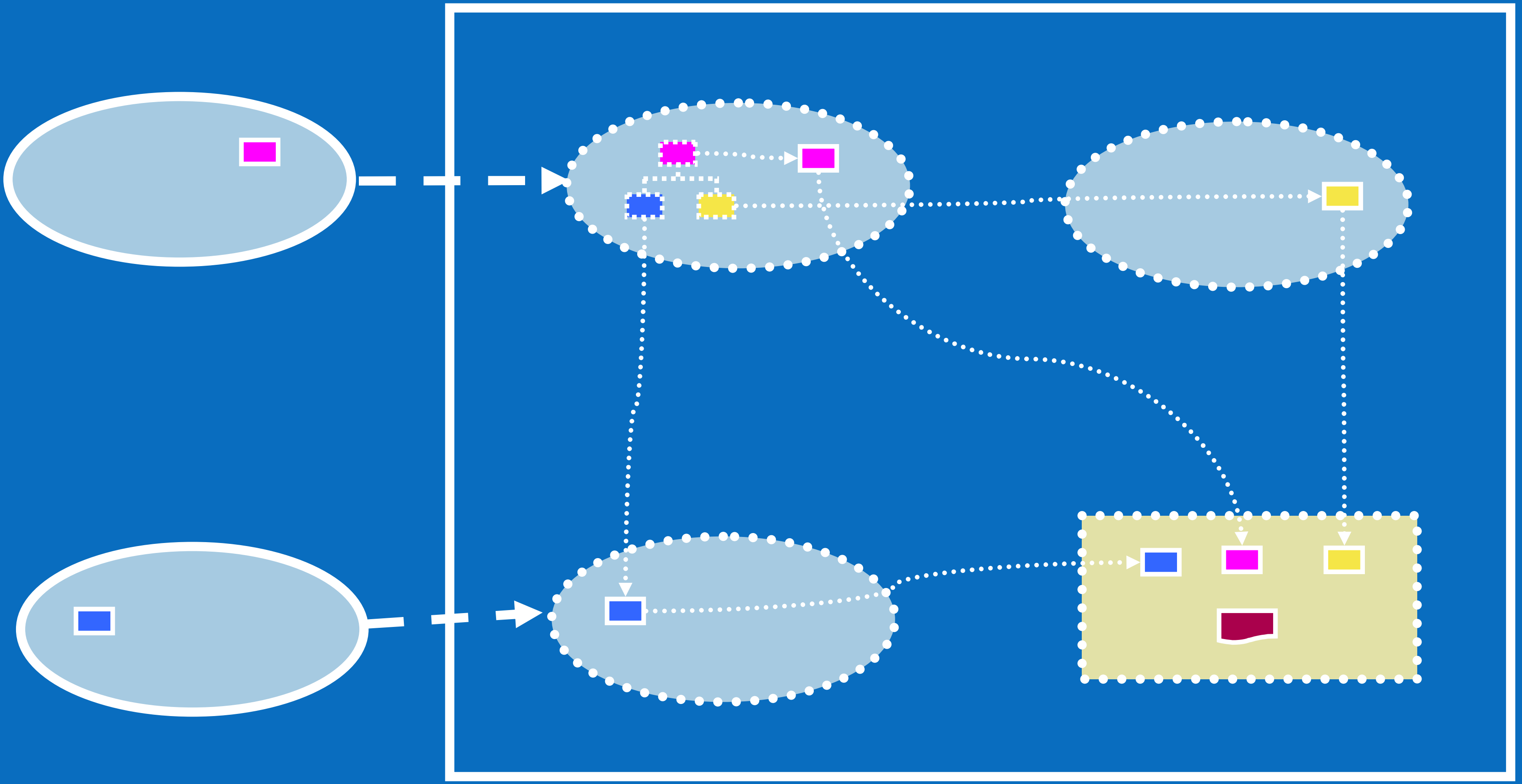
Alternative Modules
- Hasim functional partition
- Hasim functional partition semantic model
- >> Hasim to FAST functional partition wrapper
- SMIPS Functional Partition
- SMIPS New Functional Partition

Name: Hasim to FAST functional partition wrapper
Description: Hasim to FAST functional partition wrapper
Attributes: Hasim_to_FAST_functional_partition_wrapper qemu fast hasim
Provides: hasim_funcp
Requires: fast_funcp
Filename: modules/functional/qemu-fast/qemu-fast.awb
Public: Hasim2FAST.bsv
Private:
Parameters:

Module Interfaces

- Plumbing Modules
- Algorithm Modules
- Message Modules
- Library Modules

AWB Operation



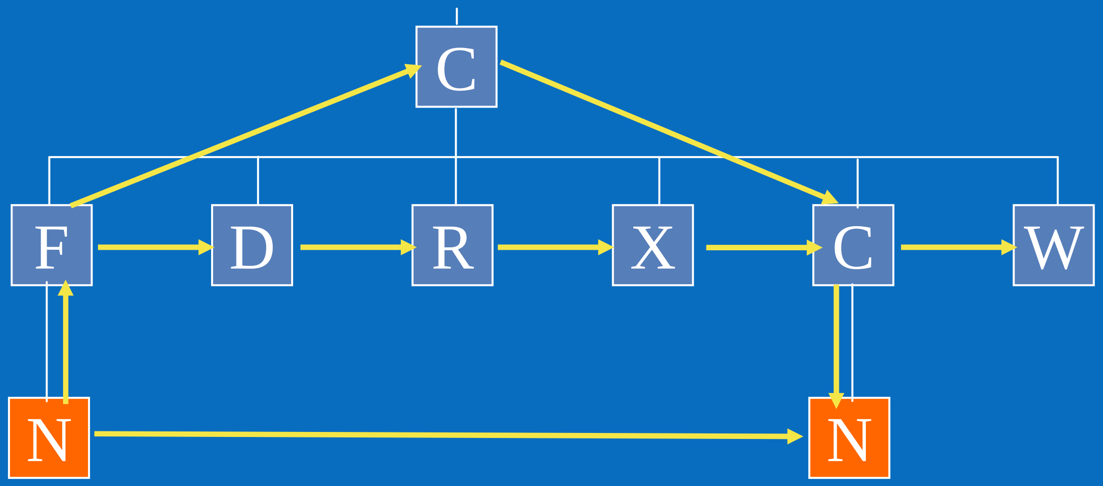
Repositories

Workspace

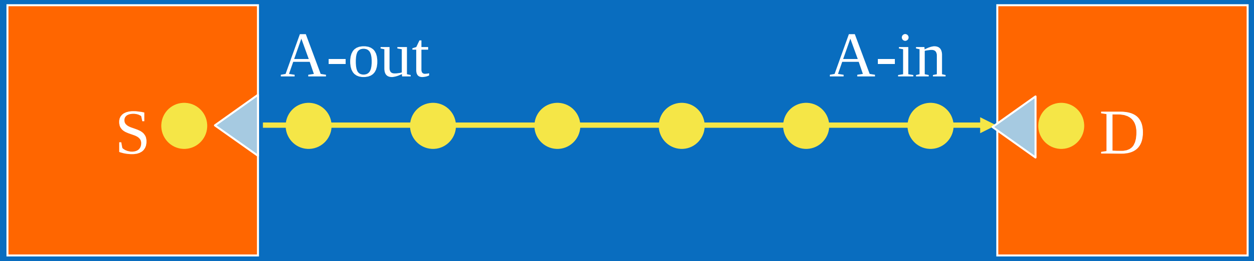
Build Features

- Model level parameter specification
- Automatic Makefile creation from templates (L2)
- Bluespec module dependence analysis
- Easy to specify synthesis boundaries (L3a)
- Support for parallel builds (L3b)
- Allows BDPI and Verilog modules (L7)
- Support for hybrid hardware/software modules
- Targets bitfile, iverilog, Bluesim

Communication:
A modularity speedbump



Soft Connections: Flattening the speedbumps



Soft Connections

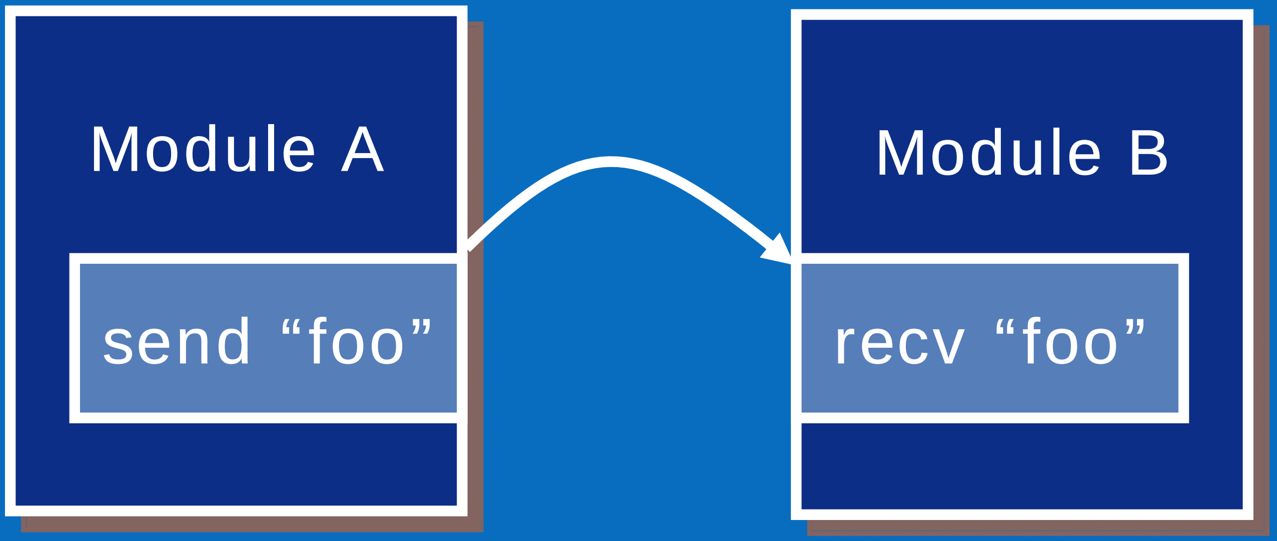
- Use “ModuleCollect” to collect connection names:

```
let my_con <- mkConnection_Send(“dec_to_exe”);
```

- Use static elaboration to find/join ends. Pseudo-code:

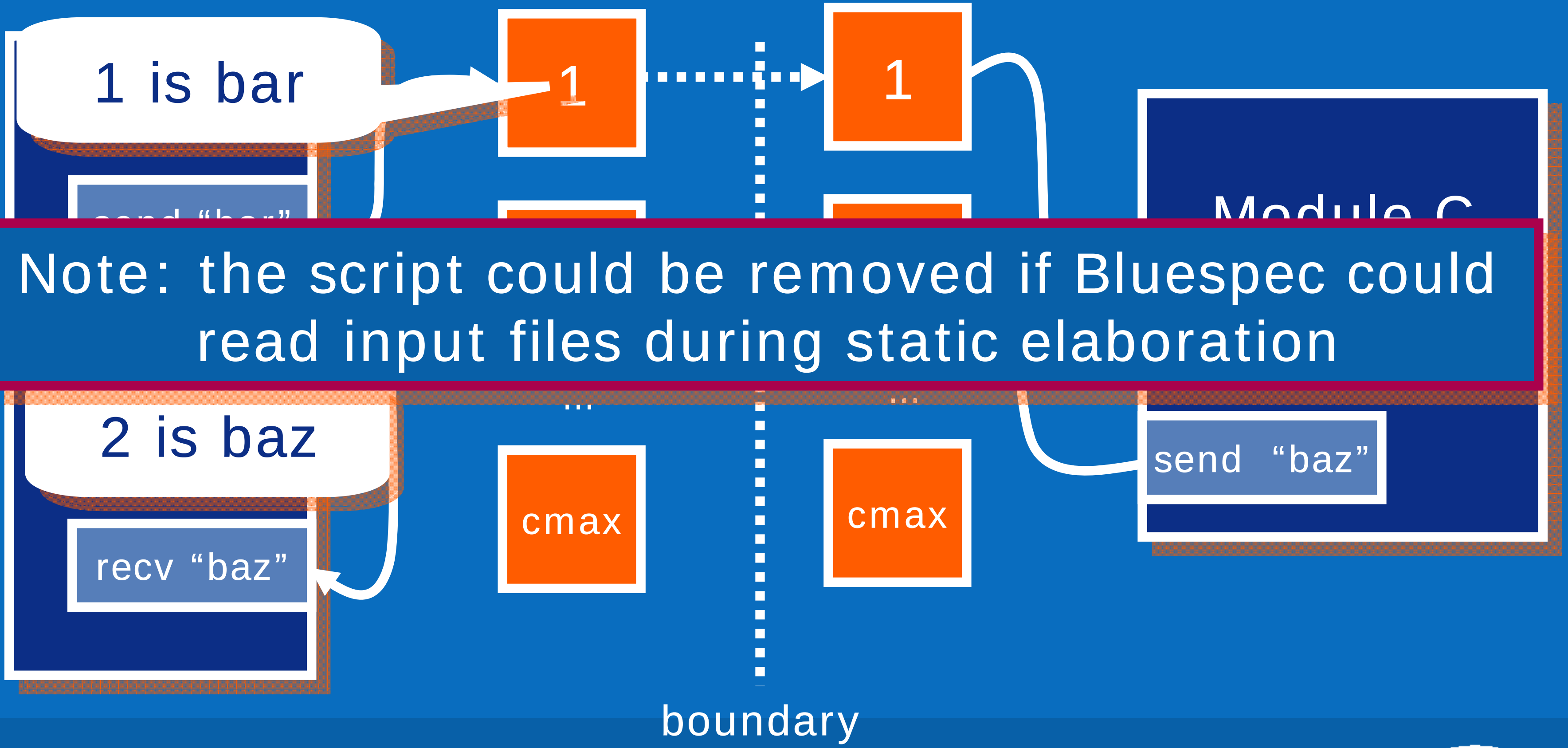
```
let cons <- getCollection(toplevel);               //Get the connections
match {.sends, .recs} = splitConnections(ld);     //Split into sends, recs
match {.dang_sends, .dang_recs, .cncts} = groupByName(sends, recs);
foreach {.send, rec} in cncts
       mkConneciton(send, rec);
if (dang_sends != nil || dang_recs != nil)
       error “Dangling Connections at top level!
```

Connections



Connections Across Synthesis Boundaries (L3)

4. Use the previous scheme to connect as normal



Note: the script could be removed if Bluespec could read input files during static elaboration

Acknowledgments

- David Goodwin
- Artur Klauser
- Martha Mercaldi
- Toni Juan
- Srilatha Manne
- Nate Binkert
- Shubu Mukherjee
- Angshu Parashar
- Ramon Matas
- Arvind
- Saila Parthasarathy
- Krishna Rangan
- Brian Slechta

Soon…

http://asim.csail.mit.edu

intel®

Leap ahead™

Backup