DESIGNING DIGITAL SIGNAL PROCESSORS WITH ROCKETCHIP

Paul Rigge, Borivoje Nikolić
{rigge,bora}@eecs.berkeley.edu
UC Berkeley

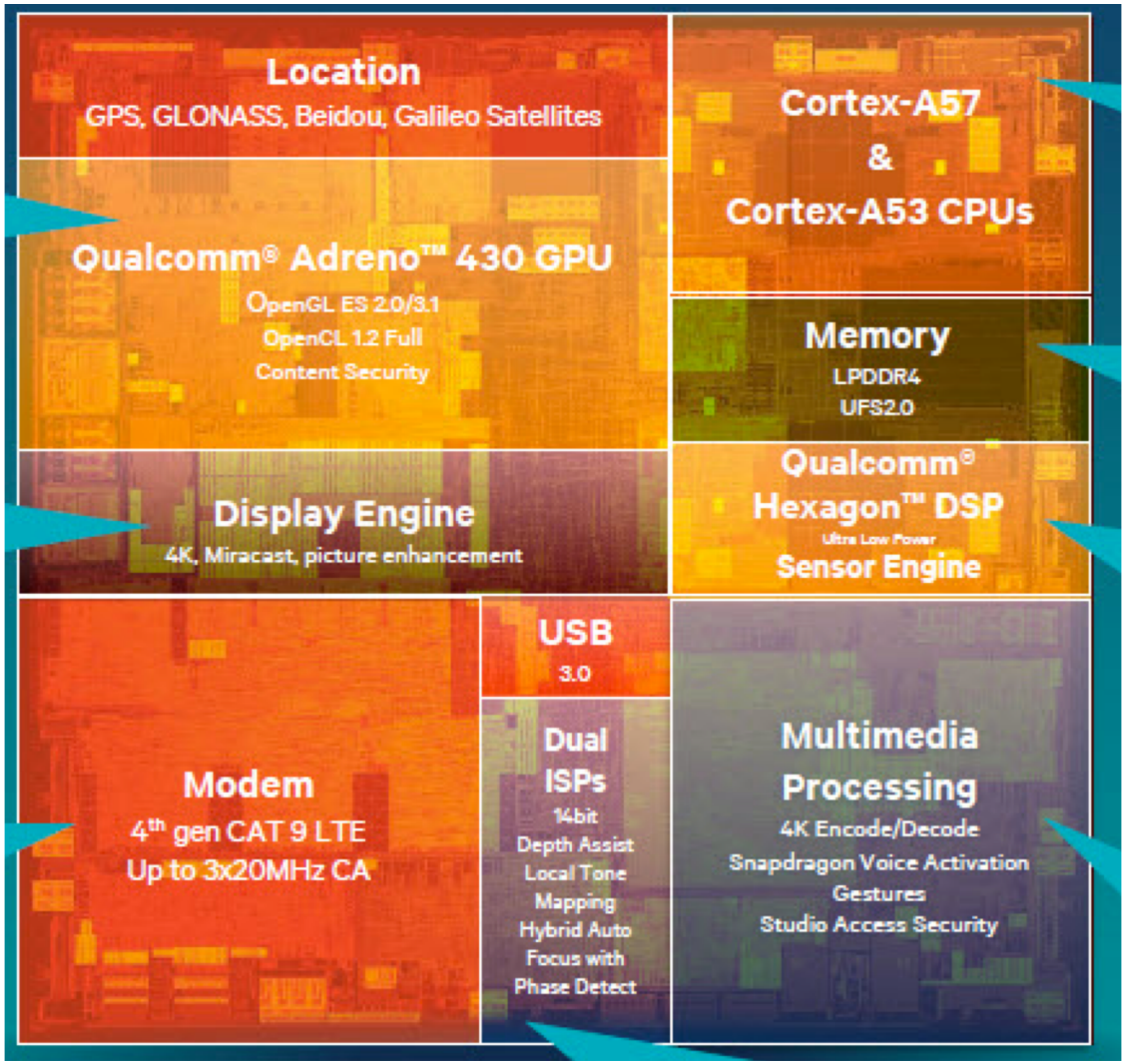
CARRV 2018

June 2, 2018

BWRC

ADEPT

SoCs Combine Programmability + Efficiency



https://telecomtalk.info/qualcomm-announces-64-bit-snapdragon-808-and-snapdragon-810-high-end-mobile-processors/115805/

BWRC

Digital Signal Processing

ADEPT

- SoCs Integrate a lot of signal processing
  - Cellular+WiFi
  - Audio
  - Image Processing
  - GPS
- Strongly benefits from custom hardware
  - Parallelism
  - High locality
  - Trim unneeded bits

BWRC

Designing SoCs is Hard

ADEPT

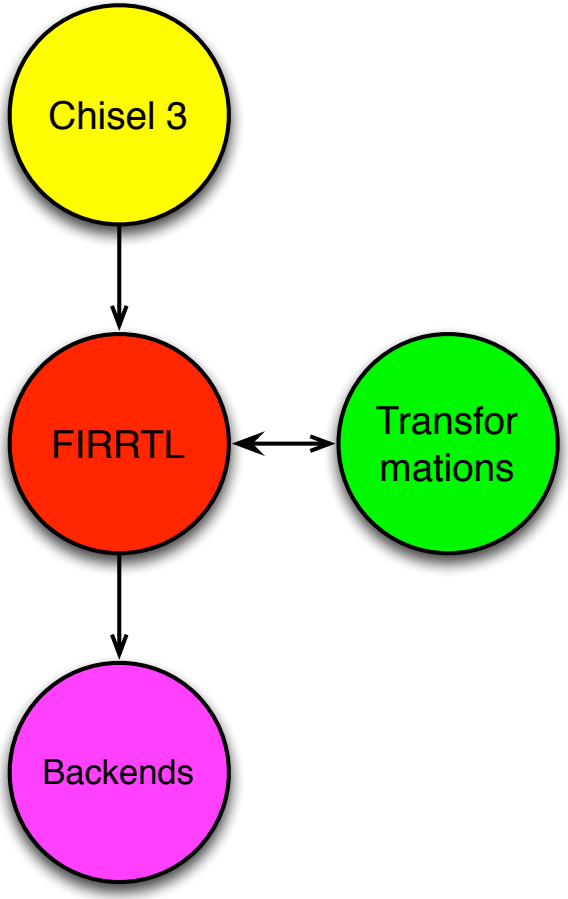
- Long development cycle
- High cost of tools, respins
- Reuse limited
- High NRE only justifiable in high volume

BWRC

Chisel, FIRRTL, Rocketchip

ADEPT

- Chisel: domain specific language (DSL) for writing programs that generate circuits
- FIRRTL: Flexible Intermediate Representation for RTL (LLVM for hardware)
- RocketChip: Open-source RISC-V implementation in Chisel



BWRC

Growing Infrastructure

ADEPT

- Stable cores
- Compilers
- Software Infrastructure
- Accelerators
  - DMA
  - Hwacha
  - Etc.
- Interconnect Generators

BWRC

ADEPT

Outline

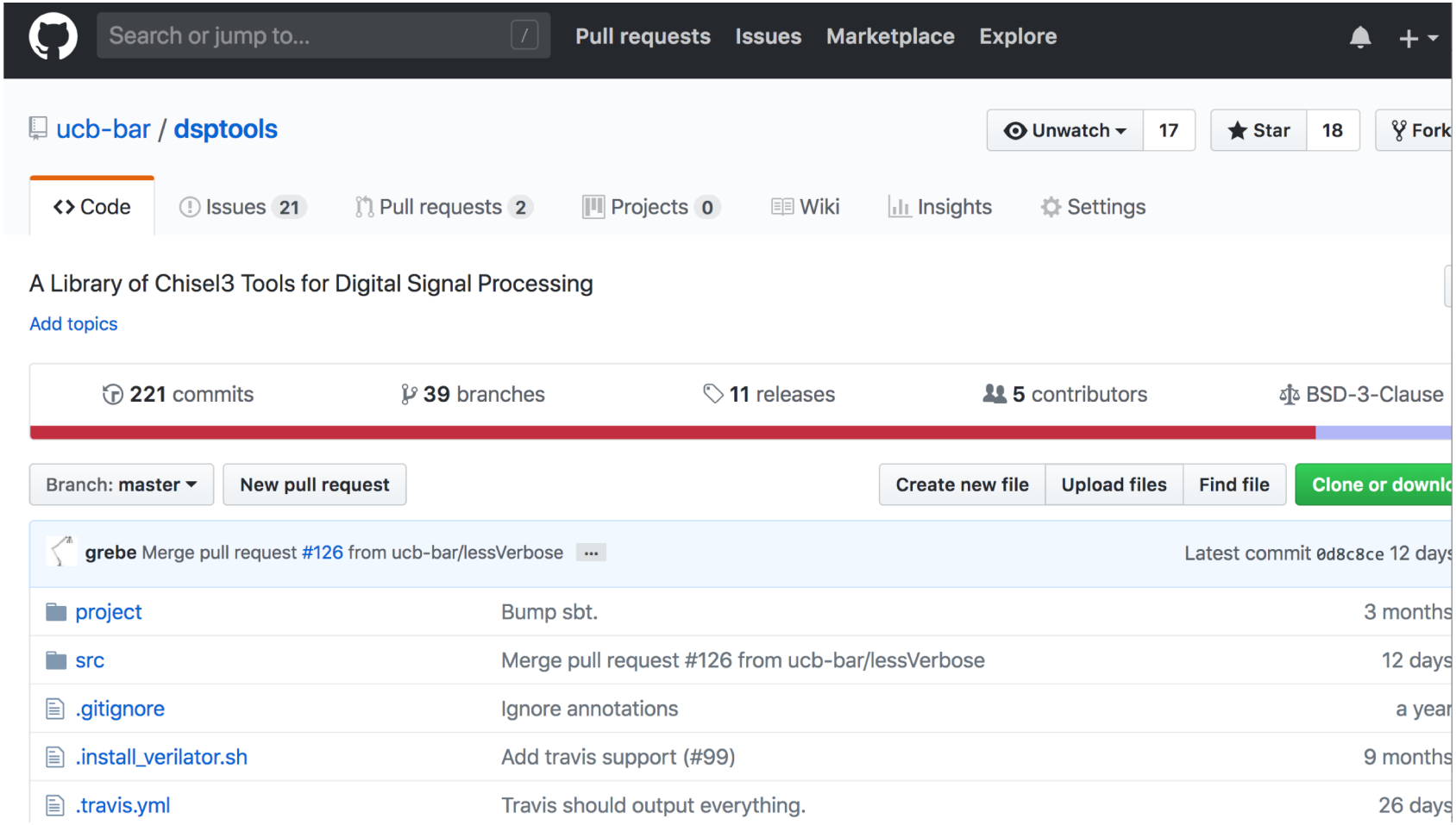
- DSP Generators in Chisel
- AXI-4 Stream + Diplomacy
- Memory-mapped DSP Peripherals
  - Useful building blocks
- Verification
- OFDM Baseband Example

DSP Generators in Chisel

BWRC

dsptools

ADEPT

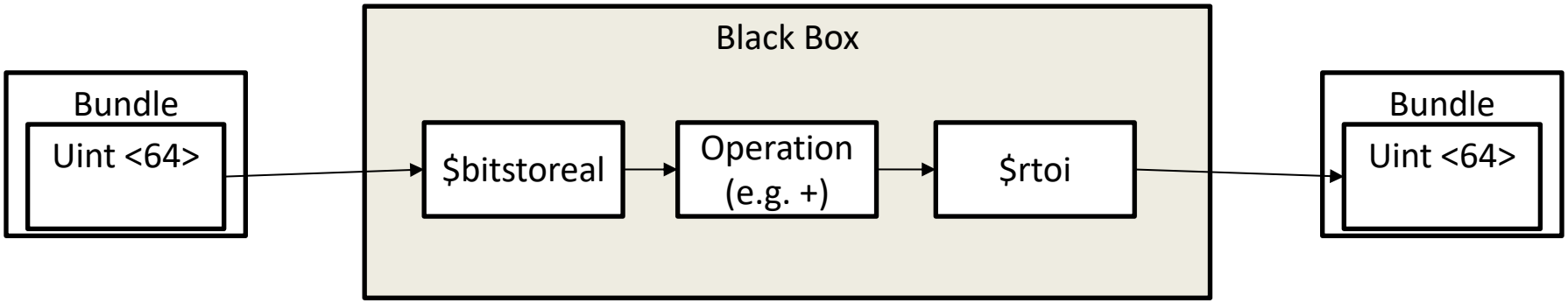


https://github.com/ucb-bar/dsptools

Paul Rigge, Angie Wang, Stevo Bailey, Chick Markley, Adam Izraelevitz

Floating Point

- Start implementing hardware without worrying about precision
  - Validate IO, control logic, algorithm
- Validate floating point hardware implementation against floating point golden model
- Uses Verilog “real” types (non-synthesizeable)



BWRC

Fixed Point

ADEPT

- Fixed point types in Chisel and FIRRTL
- Width inference like UInt, SInt
- Binary point inference

```
val sel = Wire(Bool())
val a = Wire(FixedPoint(width = 10.W,
                        binaryPoint = 9.BP))
val b = Wire(FixedPoint(width = 12.W,
                        binaryPoint = 10.BP))
val reg = Reg(FixedPoint())
when (sel) {
      reg := a
} .otherwise {
      reg := b
}
```

BWRC

Complex Numbers

ADEPT

- dsptools defines a Complex type
- Generic as to underlying type, i.e. DspComplex[SInt] or DspComplex[FixedPoint] or DspComplex[FloatingPoint] OK
- Can choose between 3 or 4 real-multiplies for a complex multiply

BWRC

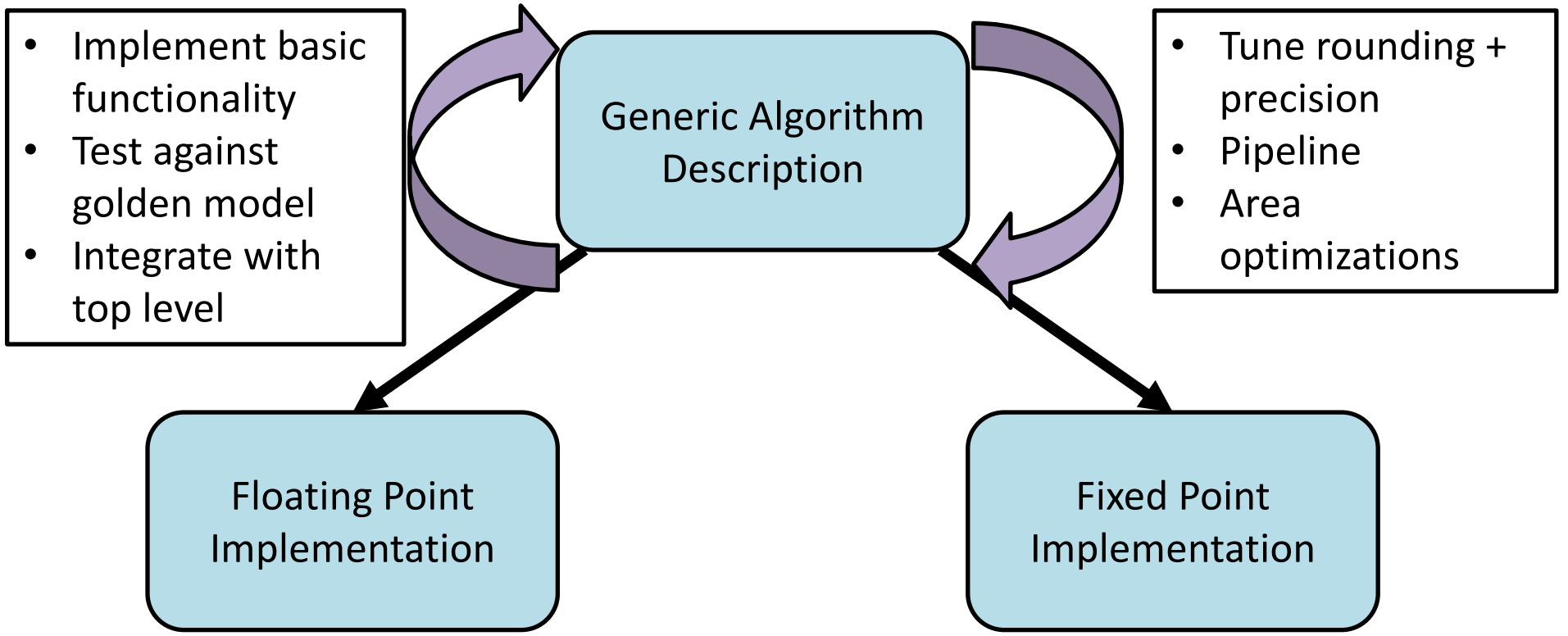
DspContext

ADEPT

- Automatic pipeline register insertion for adds, multiplies
- Rounding
- Precision for literals
- Override global defaults with scope, e.g.

```
DspContext.alter(myContext) {
    val sum = a context_* b // auto-pipelined
}
```

Polymorphic Generators



BWRC

Numeric Type Classes

ADEPT

- Type classes: support ad hoc polymorphism by adding constraints to type variables
- Support using type-generic generators with user-defined types
- Use type classes from Spire numeric library
  - Add new type classes for hardware constructs, especially Bool
  - Hide expensive operations, e.g. division, sqrt

BWRC

Numeric Type Classes

ADEPT

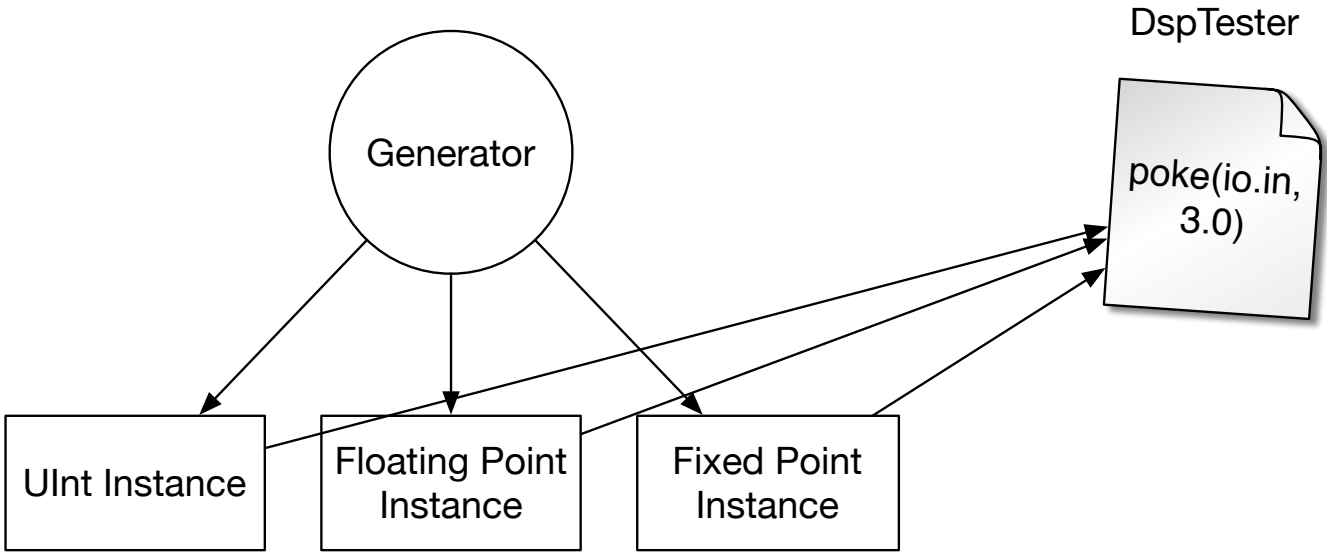
- Ring
  - +, -, *, zero, and one
- Eq
  - === and =!=
- Order (extends Eq)
  - <, >, <=, >=, max, min
- Real (extends Ring with Order with Sign)
  - ceil, floor, round, isWhole
- Integer (extends Real)
  - mod

BWRC

ADEPT

DspTester

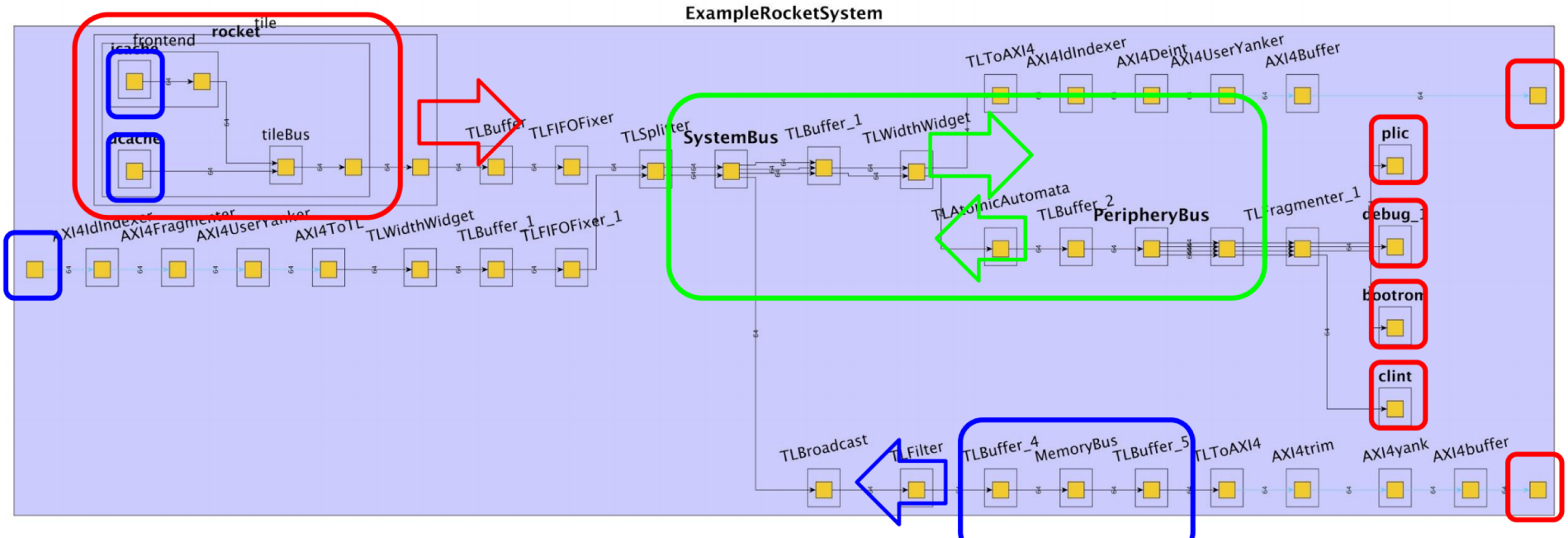
- Verification needs to be as parameterizable as hardware generators
- Type-generic PeekPokeTester
- Assert output is within epsilon (set by type)



AXI-4 Stream + Diplomacy

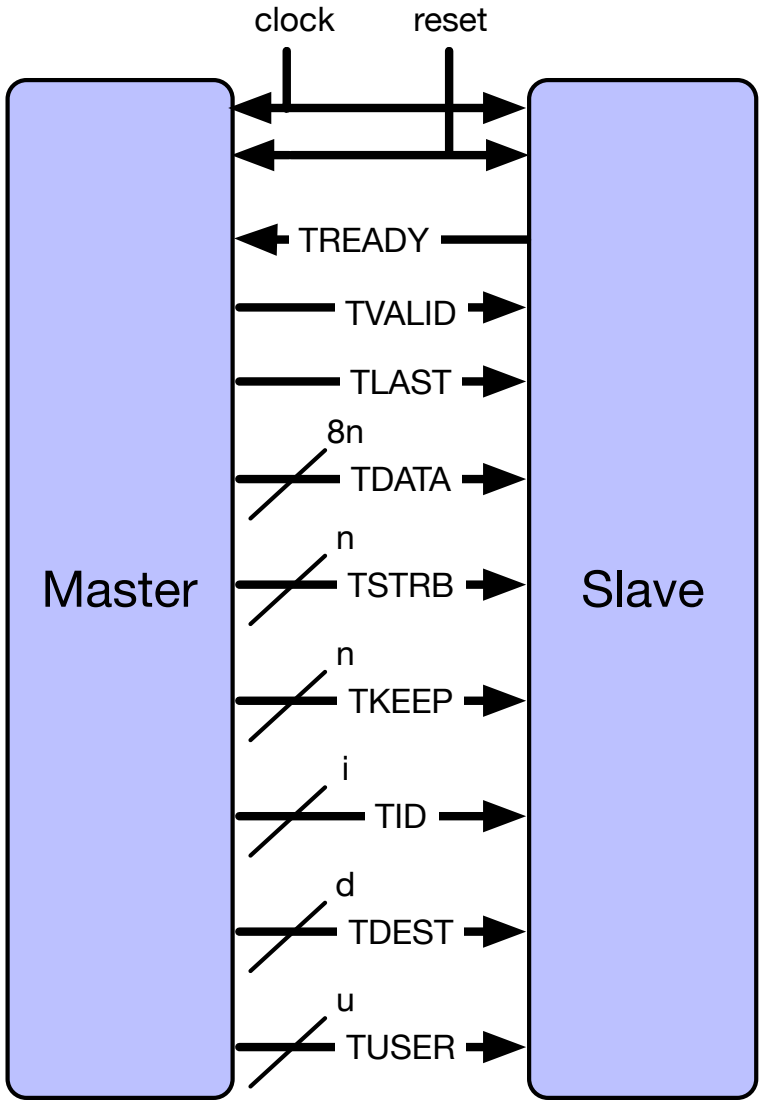
Diplomacy Background

- Generator runs in two phases
  - Negotiate parameters
  - Elaborate hardware
- Parameters flow both “in” and “out”



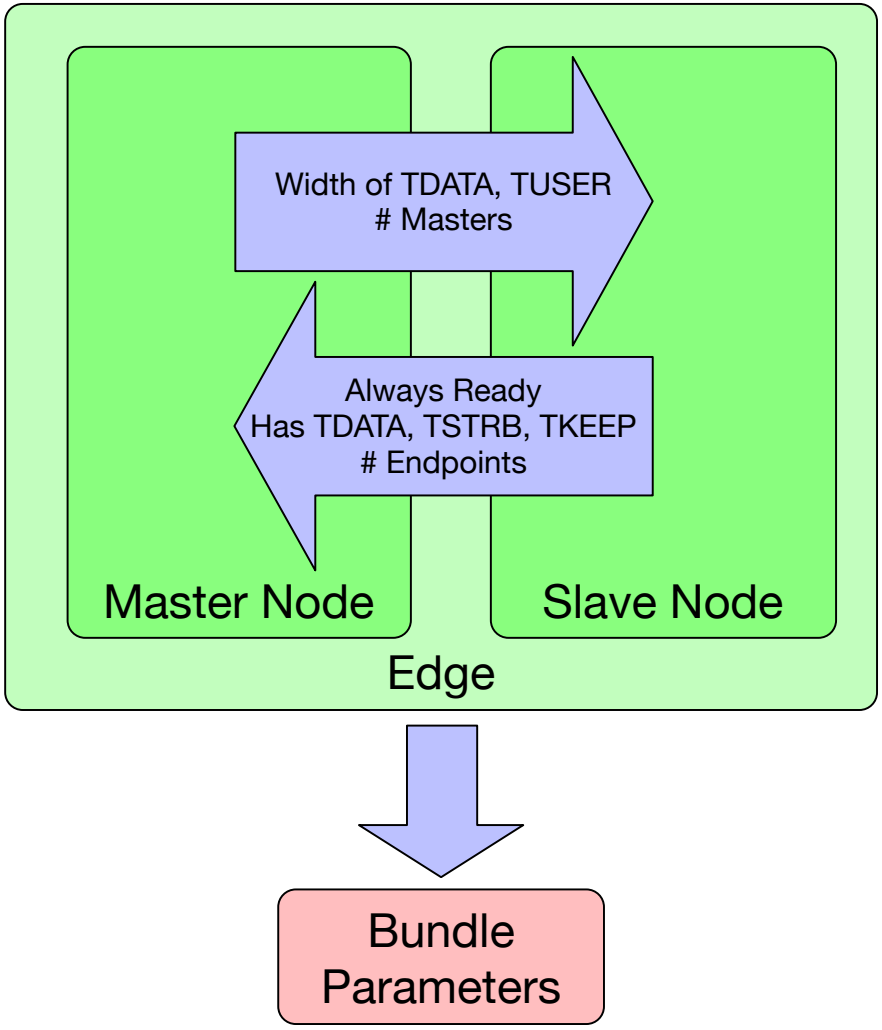
AXI-4 Stream

- AMBA standard for streaming data
- Defines ready/valid handshake semantics
- Most fields optional
  - Even TDATA!



AXI-4 Stream Diplomacy

- Nodes exchange parameters
- Edge resolves parameters, chooses bundle parameters



BWRC

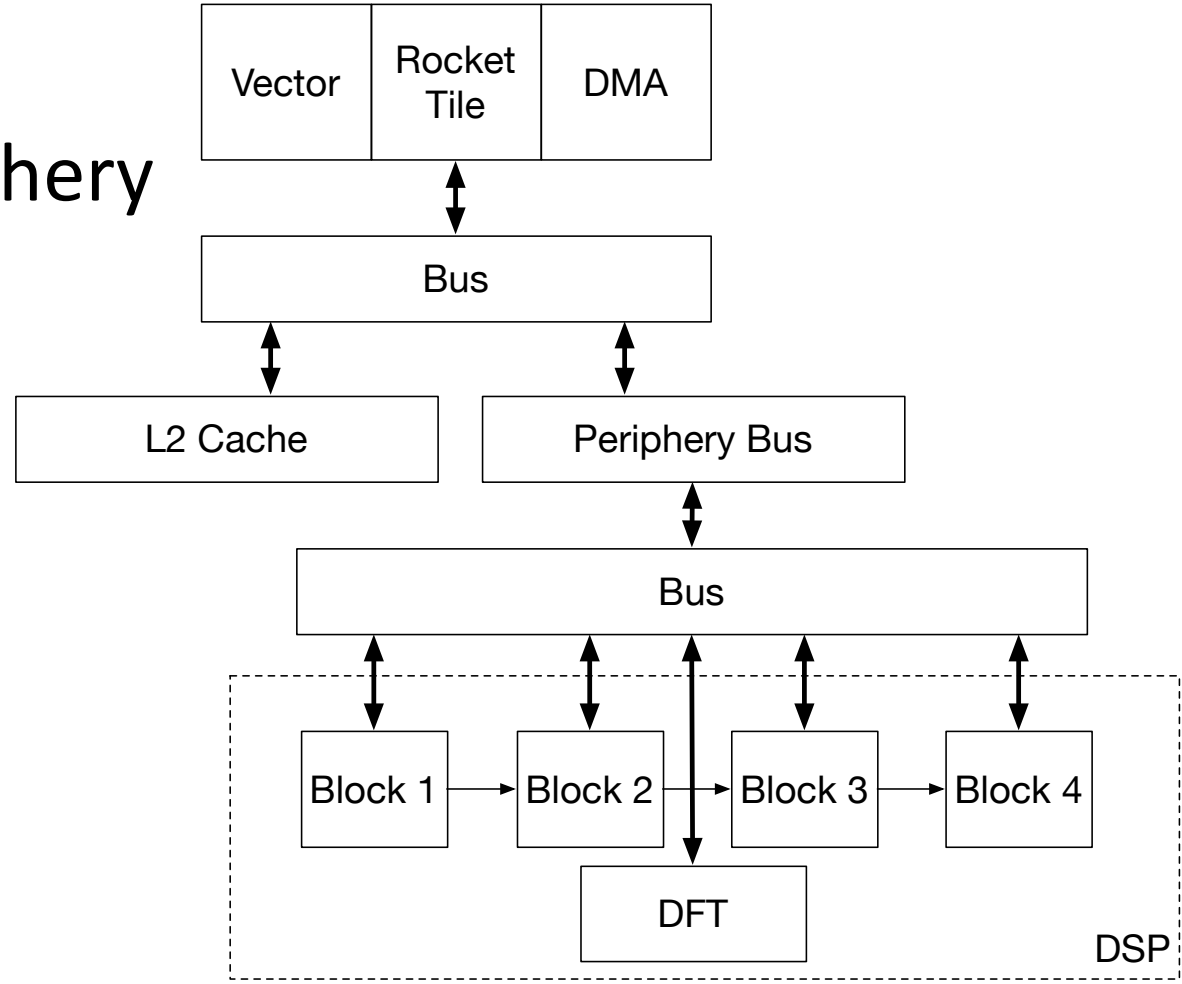
ADEPT

Memory-mapped DSP Peripherals

BWRC

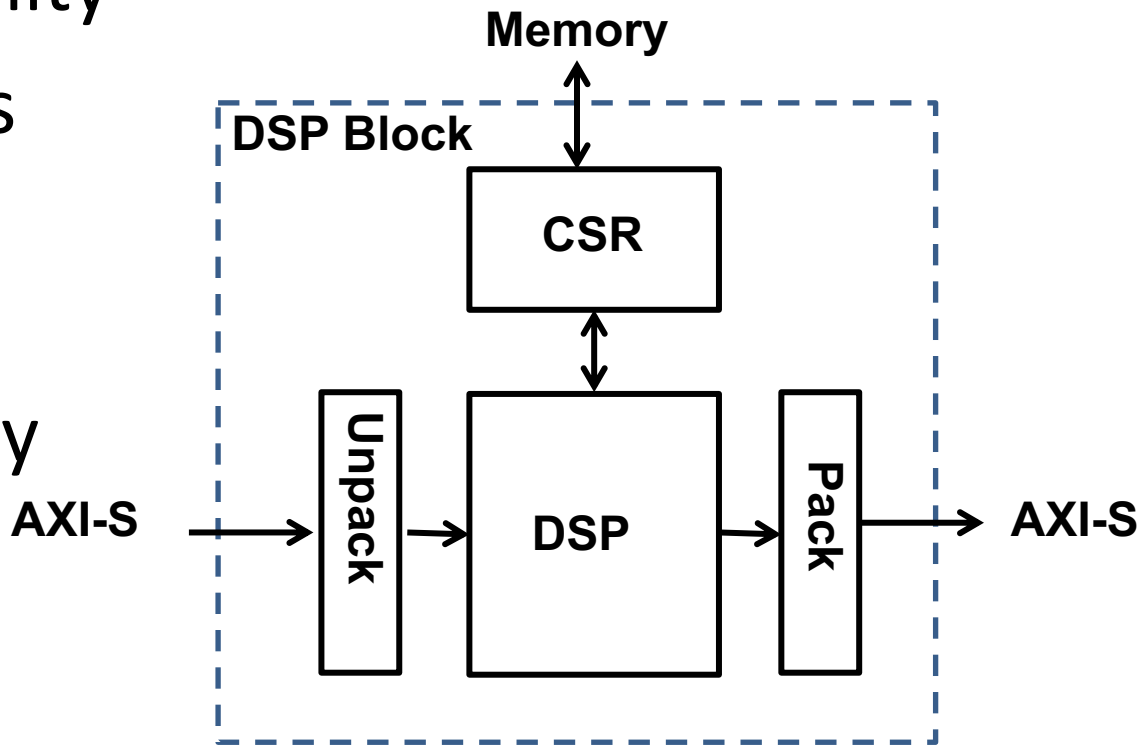
ADEPT

- Connect DSP accelerators to Rocket via Periphery Bus



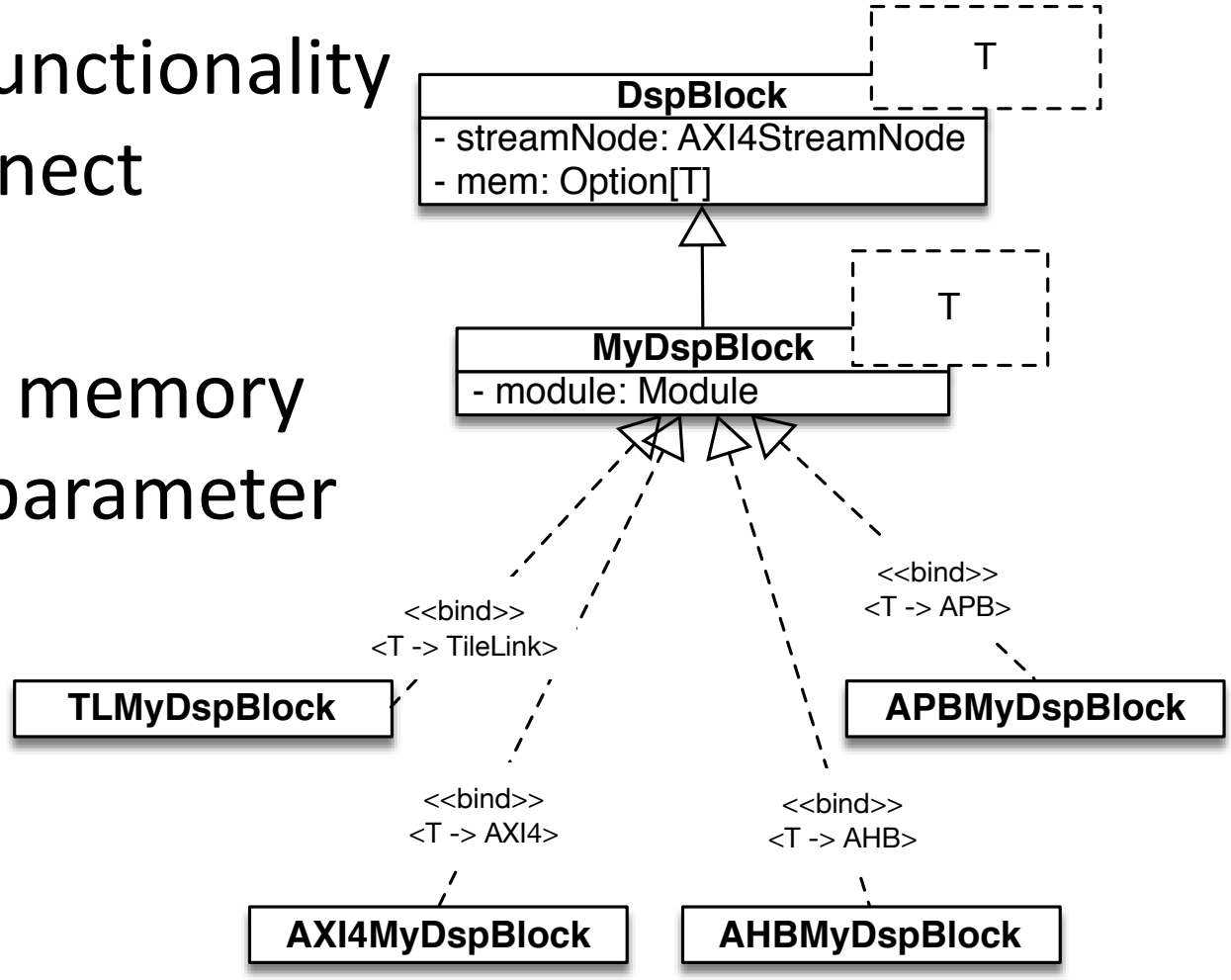
DspBlock

- Basic building block of DSP functionality
- Streaming inputs and outputs (any number)
- Optional memory interface



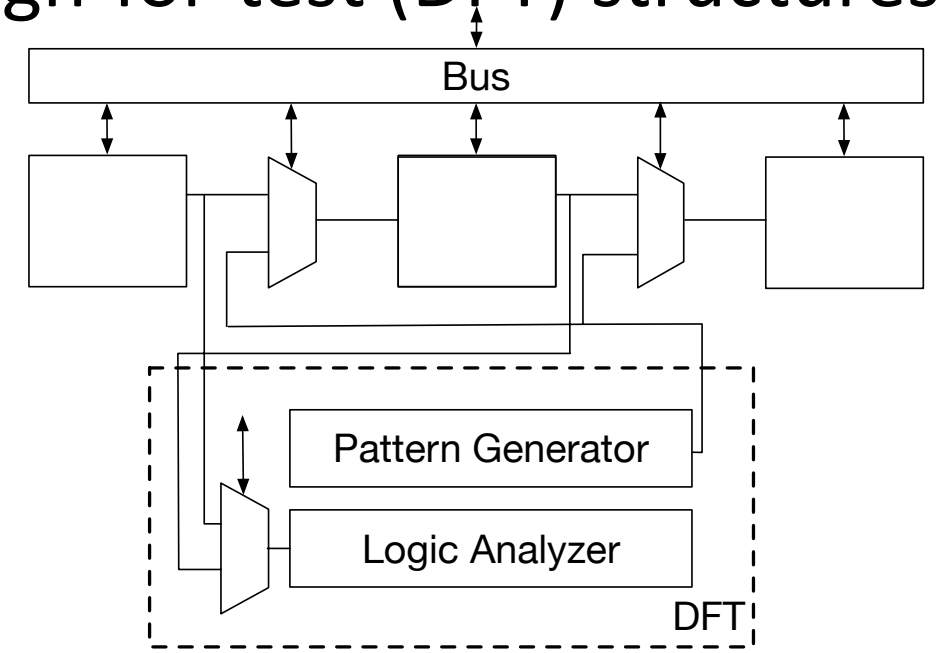
Type-generic DSP Blocks

- Define DSP functionality and interconnect separately
- Treat type of memory interface as parameter



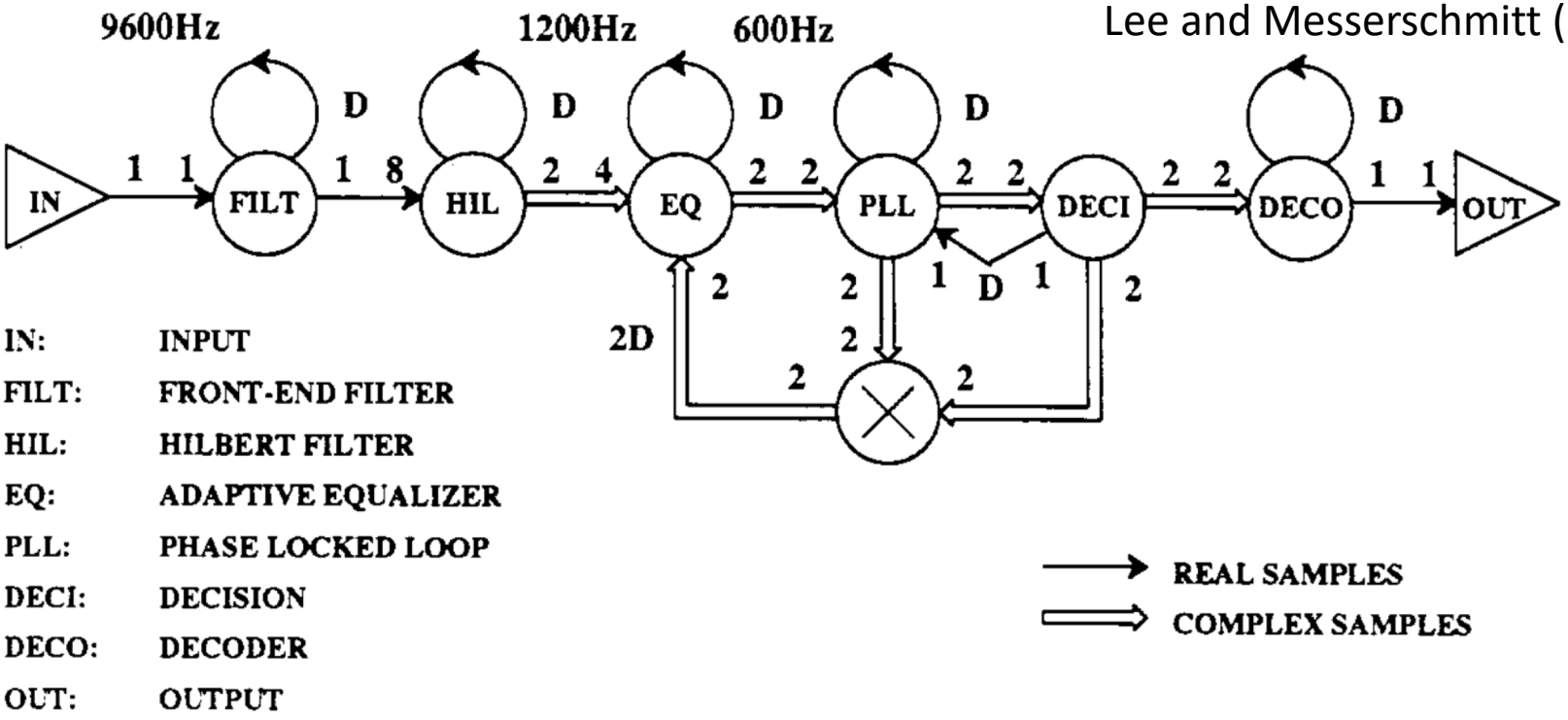
DspChain

- DspBlock composed of many internal DspBlocks
- Generate memory interconnect, connections between blocks
- Add design-for-test (DFT) structures



Synchronous Data Flow

Lee and Messerschmitt (1987).



- Represent computation as digraph
- Samples produced/consumed by each node known a priori

BWRC

DspRegister, DspQueue

ADEPT

- Building blocks for SDF-style design
- DspRegister
  - Register with programmable vector length
  - Stream in and out simultaneously
  - Processor has visibility into contents
- DspQueue
  - Throw interrupt when entries exceeds programmable threshold
  - Support real-time

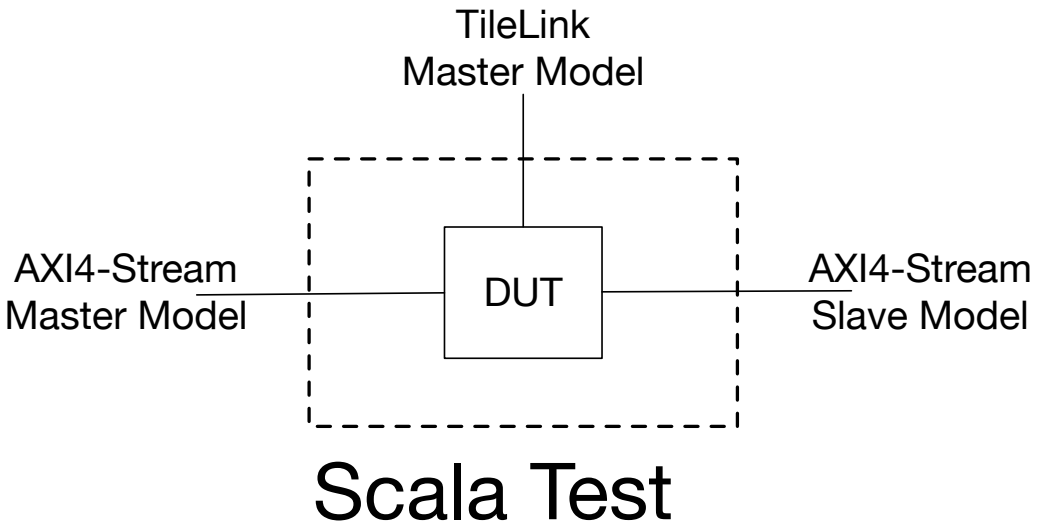
BWRC

ADEPT

Verification

Unit Tests

- PeekPokeTester
  - Type-generic with DspTester
- FIRRTL Interpreter very fast for small tests

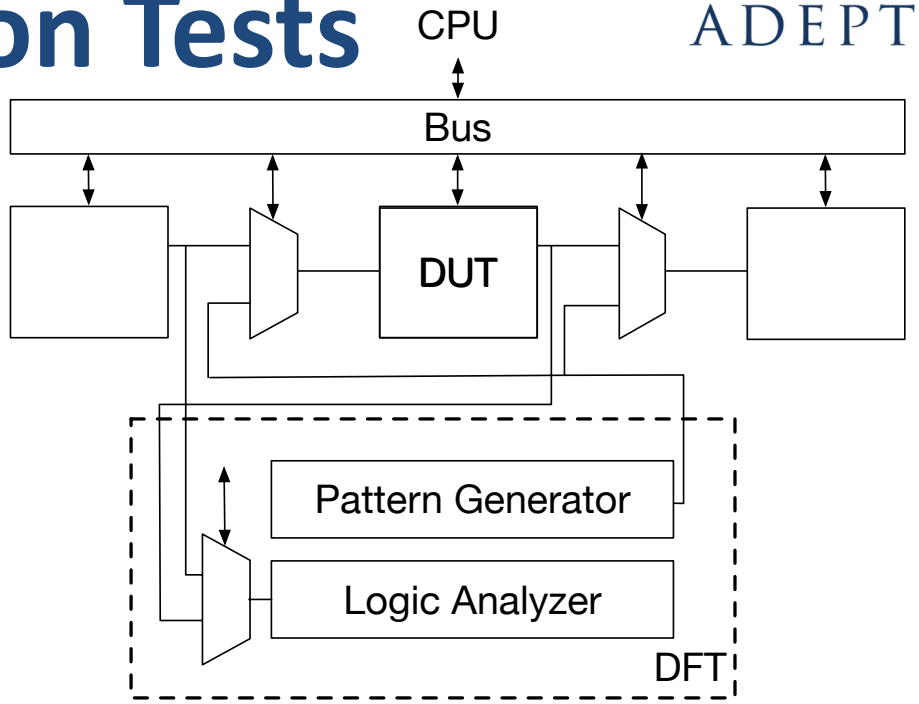


Scala Test

```
// initialize DFT
bindMaster(dut.io.in).addTransactions(testVector)
val slave = bindSlave(dut.io.out)

// Interact with DUT
val out = tlReadWord(dutAddr0)
if (out != 0) {
  tlWriteWord(dutAddr1, 3)
}
assert(tlReadWord(dutAddr2) != 0)

// Get output
val output = slave.getTransactions()
```

Integration Tests

- Write C programs to run on Rocket, use Rocket test harness
- Generate design-for-test (DFT) structures
  - Load test vectors, set muxes, store outputs
- Same binary for simulation and bring-up



C Test

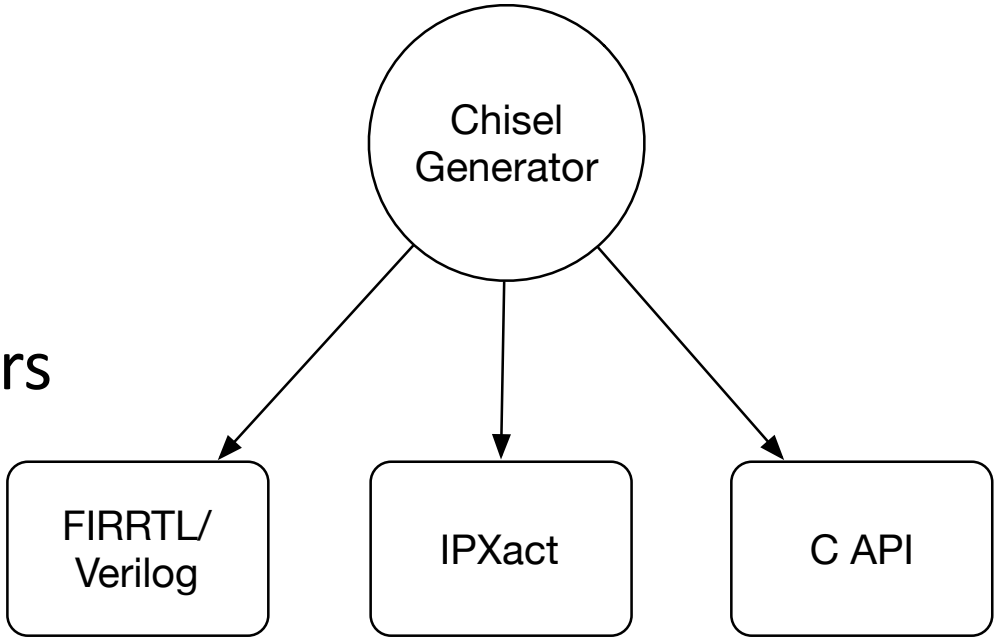
```
// Initialize DFT
pg_load_data(pg_addr, test_vector);
initiate_capture(cap_addr);

// Interact with DUT
out = read_reg(dut_addr0);
if (out != 0) {
  write_reg(dut_addr1, 3);
}
assert(read_reg(dut_addr2) != 0);

// Get output
get_output(cap_addr, output);
```

BWRC

ADEPT

IPXact

- XML schema describing circuit metadata
  - Port mappings
  - interface types
  - generator parameters
- Use external tools
  - Python test vector generation + Verification Workbench



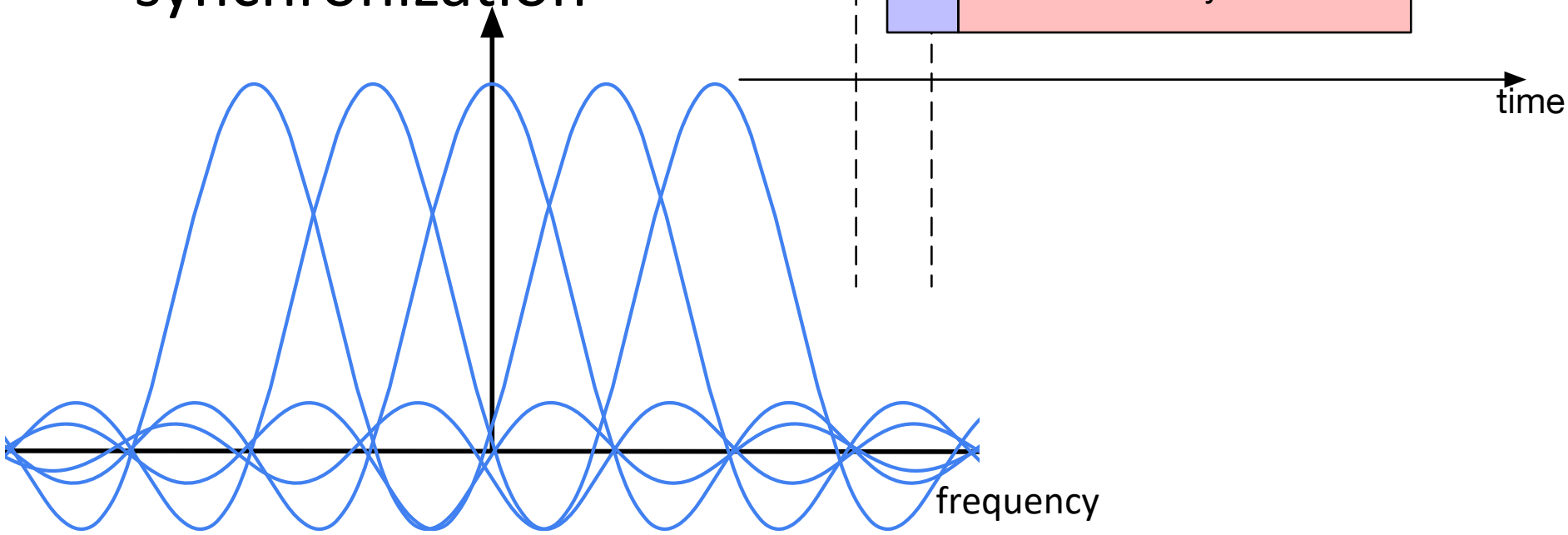
BWRC

ADEPT

OFDM Baseband Example

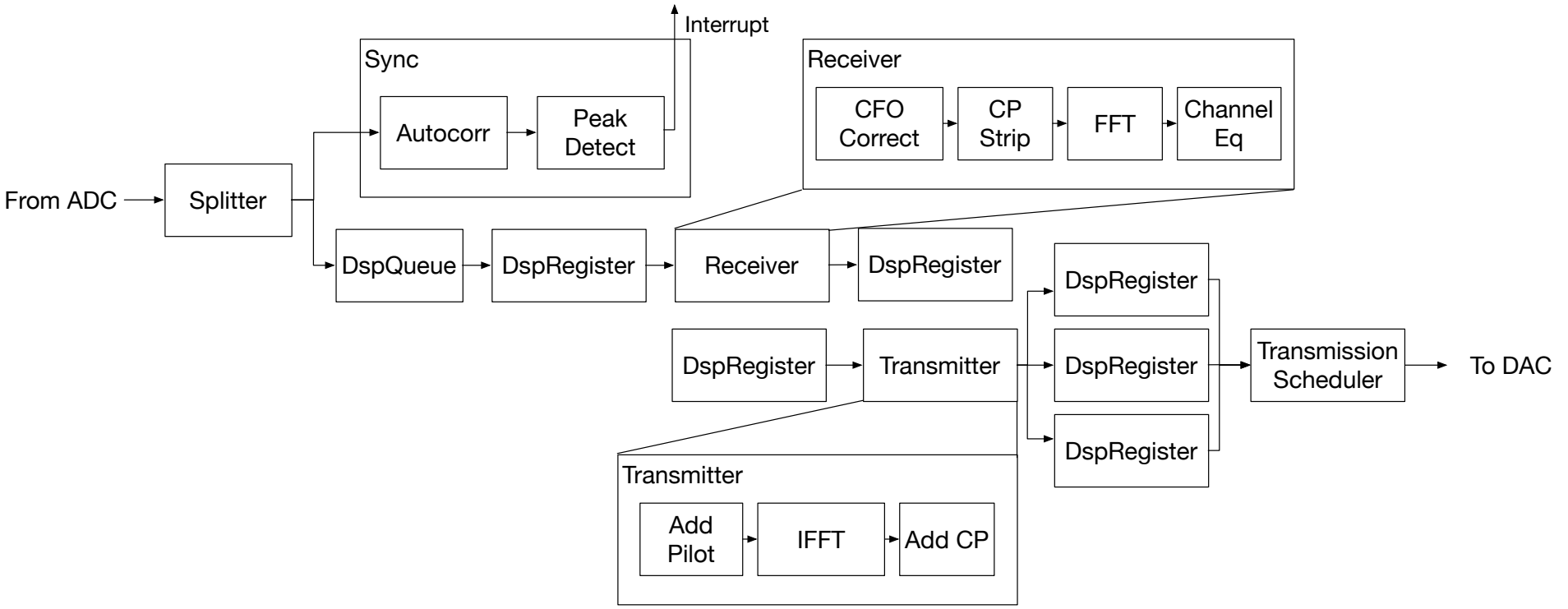
OFDM Background

- Frequency domain equalization
- Uses FFT
- Relax time domain synchronization



OFDM Baseband

- Transmitter and receiver



BWRC

ADEPT

Conclusion

- Chisel + FIRRTL + dsptools help building DSP
- RocketChip is not just a processor, library of useful components
  - Diplomacy
  - Interconnect
  - Utilities
- Can build and verify complex SoCs with RocketChip

BWRC

Thank You

ADEPT

- Collaborators
  - Stevo Bailey, Angie Wang, Adam Izraelevitz, Chick Markley, Colin Schmidt, Timo Joas, and Jim Lawson
  - UCB BAR
- Support
  - NSF GRFP
  - Adept and BWRC