FPGA-Assisted Verification

Team A1: Xiran Wang, Ali Hoffmann, Grace Fieni

Problem: the tortoise vs. the hare

What is verification?
Why do we use hardware simulation (**tortoise**)?

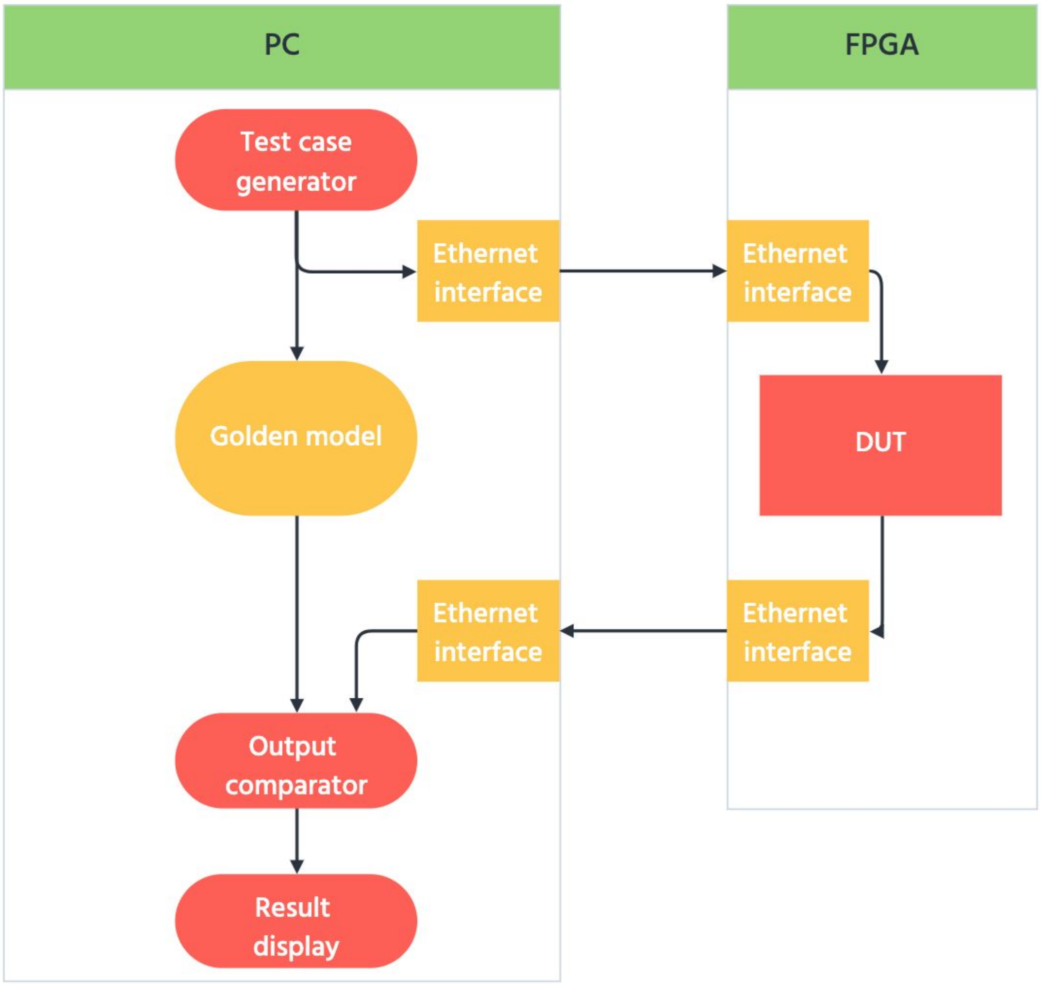
Why do we want to synthesize (**hare**)?

Solution: what if we were fast *and* won the race?

Executing tests on FPGA allow for designs to be tested on hardware ***sooner*** without sending the design to silicon.

Executing tests on FPGA through synthesis allow the same amount of testing to occur, but ***faster***.

System block diagram



- ⬭ Software
- ▭ Hardware
- 🟩 Off the shelf
- 🟨 In progress
- 🟥 Yet to be implemented

Requirements

Functionality:

- Support testing all instructions in ISA
- Test cases can have size ranging from 1 ~ 20k instructions

Performance:

- 3x speedup relative to simulation

Ease of use:

- User can customize test case size and instruction(s) to test
- Can randomize test case
- Provide exact cycle of failure

More on communication: JTAG vs USB vs Ethernet

Simulation approach

- Time increases drastically as design size increases
- For our small design, expect **< 2 MHz**

Our approach: send data to FPGA, process, send data from FPGA

- FPGA clock: **50 MHz**
- In worst case, need to send 16 + 16x16 = **272 bits/cycle**
- JTAG: **4 Mbit/s**, USB: **12 Mbit/s**, Ethernet: **1000 Mbit/s**

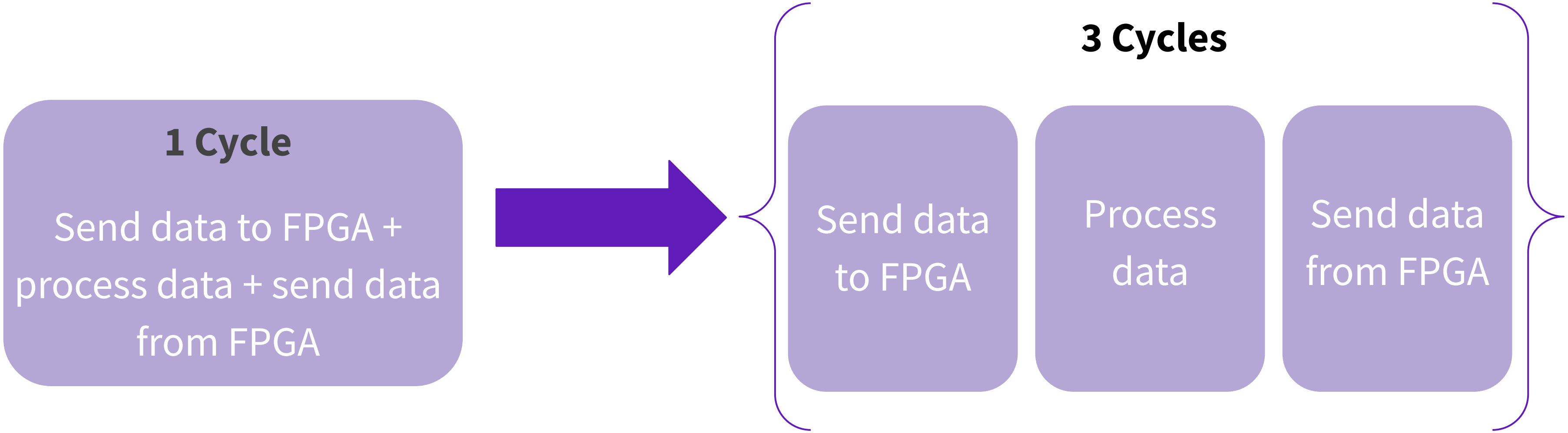
=> **Need to use Ethernet to meet 3x speedup**

More on communication: cutting down latency

Send entire register dump (16 bit instruction + 16 registers by 16 bits = **272 bits**)

→

Send only data for register that was updated (16 bit instruction + register number + 16 bits register data = **36 bits**)



1 Cycle

Send data to FPGA + process data + send data from FPGA

3 Cycles

Send data to FPGA

Process data

Send data from FPGA

More on the DUT: ISA

Support **16-bit ALU** instructions:

- Cannot support full RISC-V instruction set due to time and FPGA limitations
- ALU instructions make up >50% of typical instruction mix

Instruction format

opcode	dest reg	source 1	source 2
4 bits	4 bits	4 bits	4 bits

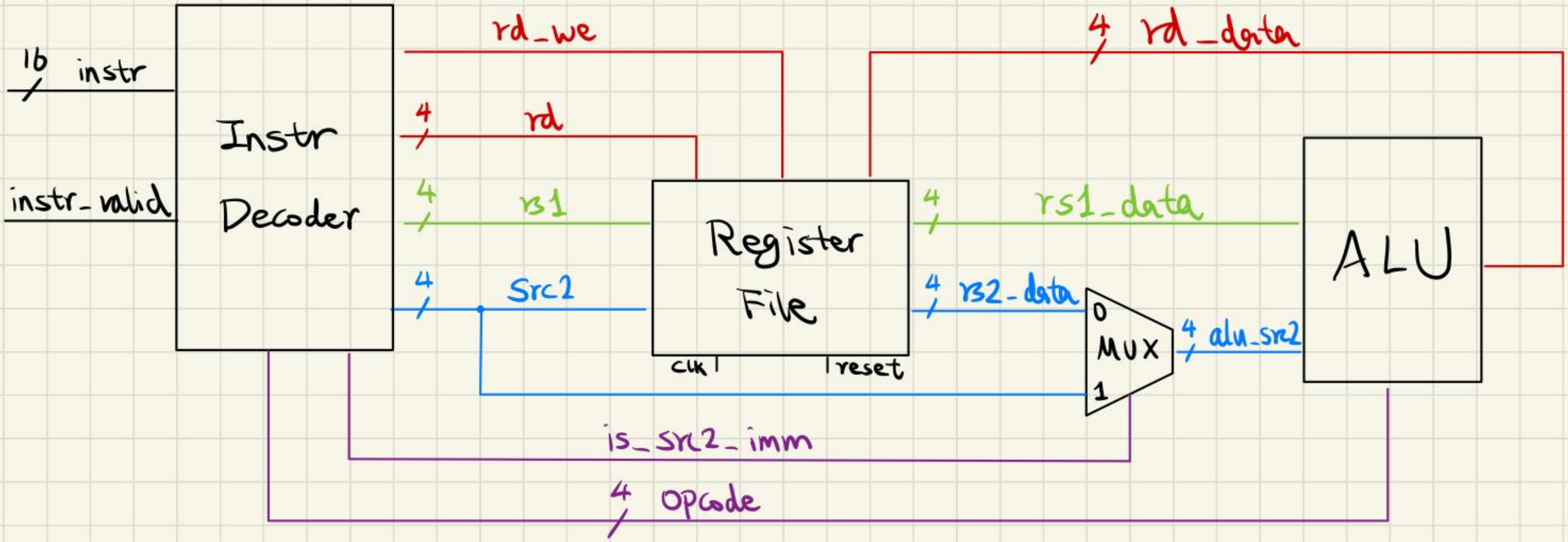
Register file

16 registers by 16 bits

Instructions

- Register movement: MOV rd, rs1
- ALU register-register: ADD/SUB/AND/OR/XOR/SLL/SRL/SRA rd, rs1, rs2
- ALU register-immediate: ADDI/ANDI/ORI/XORI/SLLI/SRLI/SRAI rd, rs1, imm

More on the DUT: hardware block diagram



Testing

Functionality (can test all instructions, test case can have 1 ~ 20k instructions):

- For all instructions, write test case containing that one instruction
- Write test cases of other sizes (e.g., 5, 100, 1k, 20k, etc.)
- **For all these cases, check communication is successful**

Performance (3x speedup):

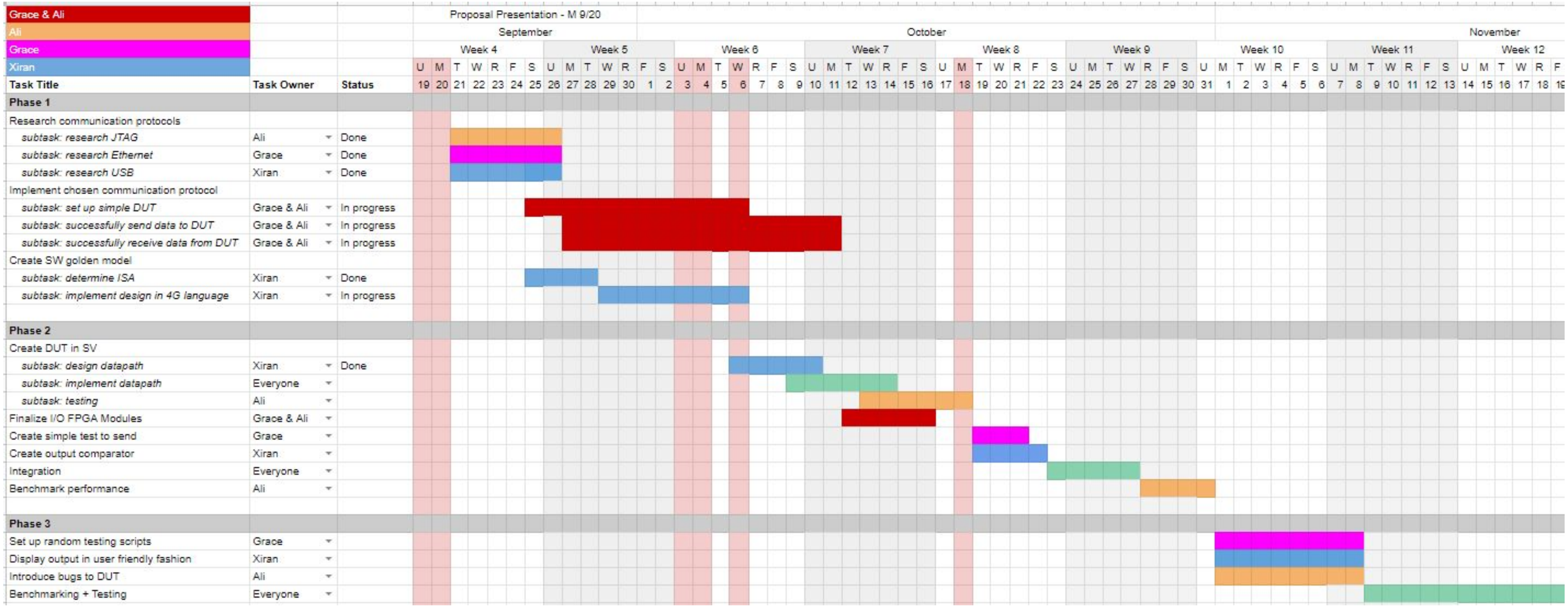
- Measure simulation time and solution time for tests of different sizes
  - Simulation time: use vcs end of simulation CPU time
  - Solution time: use time() calls at beginning and end of flow
- **Speedup = simulation time / solution time**

Testing continued

Ease of use (test case user customization, randomization, exact failure cycle):

- **Try different customizations and manually inspect test case** to see if customizations applied successfully
  - e.g., if user chose to only test ADD, test case should only contain ADD instructions
- **For randomization, manually check for different generated values**
- To see if failure cycle is reported correctly, **introduce bugs to DUT**
  - Realistic bugs to introduce: register stuck at 0, opcode decoding error

Schedule



Task Title	Task Owner	Status
Grace & Ali		
Ali		
Grace		
Xiran		
Phase 1		
Research communication protocols		
subtask: research JTAG	Ali	Done
subtask: research Ethernet	Grace	Done
subtask: research USB	Xiran	Done
Implement chosen communication protocol		
subtask: set up simple DUT	Grace & Ali	In progress
subtask: successfully send data to DUT	Grace & Ali	In progress
subtask: successfully receive data from DUT	Grace & Ali	In progress
Create SW golden model		
subtask: determine ISA	Xiran	Done
subtask: implement design in 4G language	Xiran	In progress
Phase 2		
Create DUT in SV		
subtask: design datapath	Xiran	Done
subtask: implement datapath	Everyone	
subtask: testing	Ali	
Finalize I/O FPGA Modules	Grace & Ali	
Create simple test to send	Grace	
Create output comparator	Xiran	
Integration	Everyone	
Benchmark performance	Ali	
Phase 3		
Set up random testing scripts	Grace	
Display output in user friendly fashion	Xiran	
Introduce bugs to DUT	Ali	
Benchmarking + Testing	Everyone	