

DEMOCRATIZING COMPUTER SCIENCE SIMULATION WITH A COMPONENTS LIBRARY

Presented by Bobby R.
Bruce

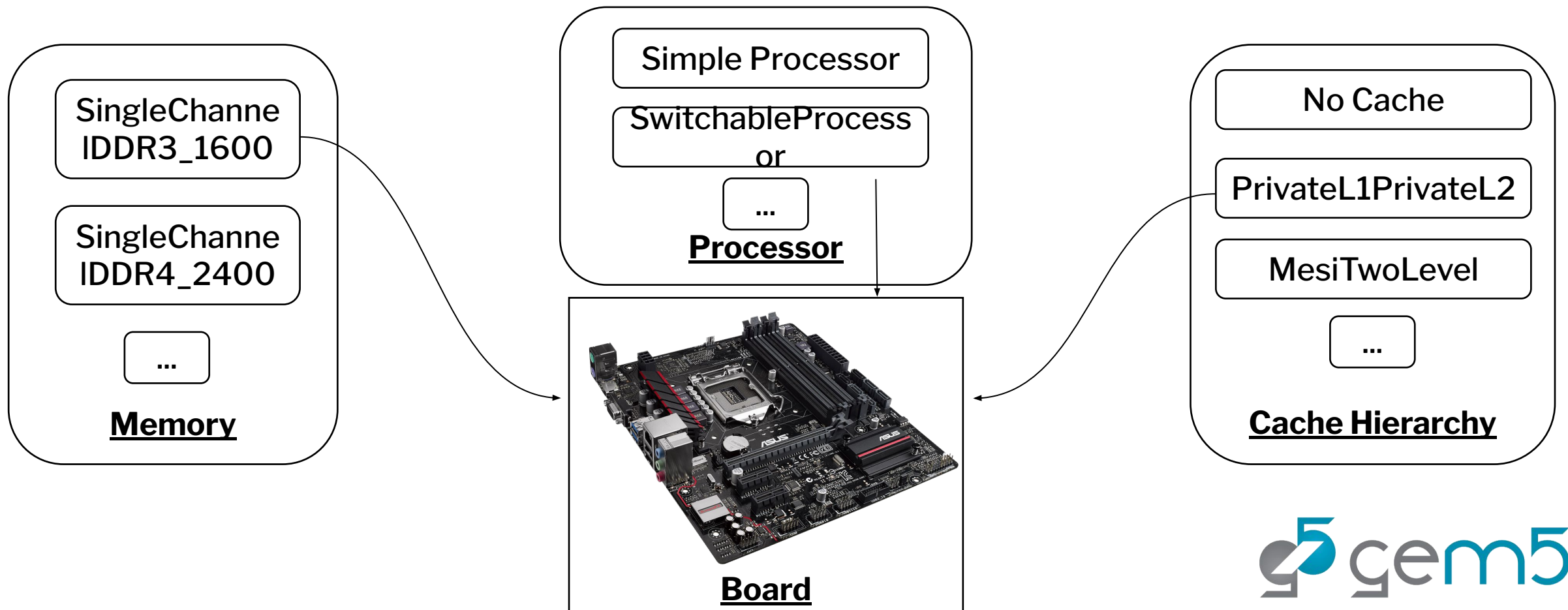
"HELLO WORLD" IN GEM5: PLENTY OF BOILERPLATE

```
simple.py
simple.py
simple.py No Selection
39 import m5
40 # import all of the SimObjects
41 from m5.objects import *
42
43 # create the system we are going to simulate
44 system = System()
45
46 # Set the clock frequency of the system (and all of its children)
47 system.clk_domain = SrcClockDomain()
48 system.clk_domain.clock = '1GHz'
49 system.clk_domain.voltage_domain = VoltageDomain()
50
51 # Set up the system
52 system.mem_mode = 'timing' # Use timing accesses
53 system.mem_ranges = [AddrRange('512MB')] # Create an address range
54
55 # Create a simple CPU
56 system.cpu = TimingSimpleCPU()
57
58 # Create a memory bus, a system crossbar, in this case
59 system.membus = SystemXBar()
60 |
61 # Hook the CPU ports up to the membus
62 system.cpu.icache_port = system.membus.cpu_side_ports
63 system.cpu.dcache_port = system.membus.cpu_side_ports
64
65 # create the interrupt controller for the CPU and connect to the membus
66 system.cpu.createInterruptController()
67
68 # For x86 only, make sure the interrupts are connected to the memory
69 # Note: these are directly connected to the memory bus and are not cached
70 if m5.defines.buildEnv['TARGET_ISA'] == "x86":
71     system.cpu.interrupts[0].pio = system.membus.mem_side_ports
72     system.cpu.interrupts[0].int_requistor = system.membus.cpu_side_ports
73     system.cpu.interrupts[0].int_responder = system.membus.mem_side_ports
74
75 # Create a DDR3 memory controller and connect it to the membus
```

- A single core setup connected directly to main memory, with no cache, requires 36 lines of Python!
- Many hundreds are required for a system capable of booting a modern OS.
- Unsupported scripts and examples are circulated in the community as many configurations do not vary between simulations.



SOLUTION: GEM5 COMPONENTS

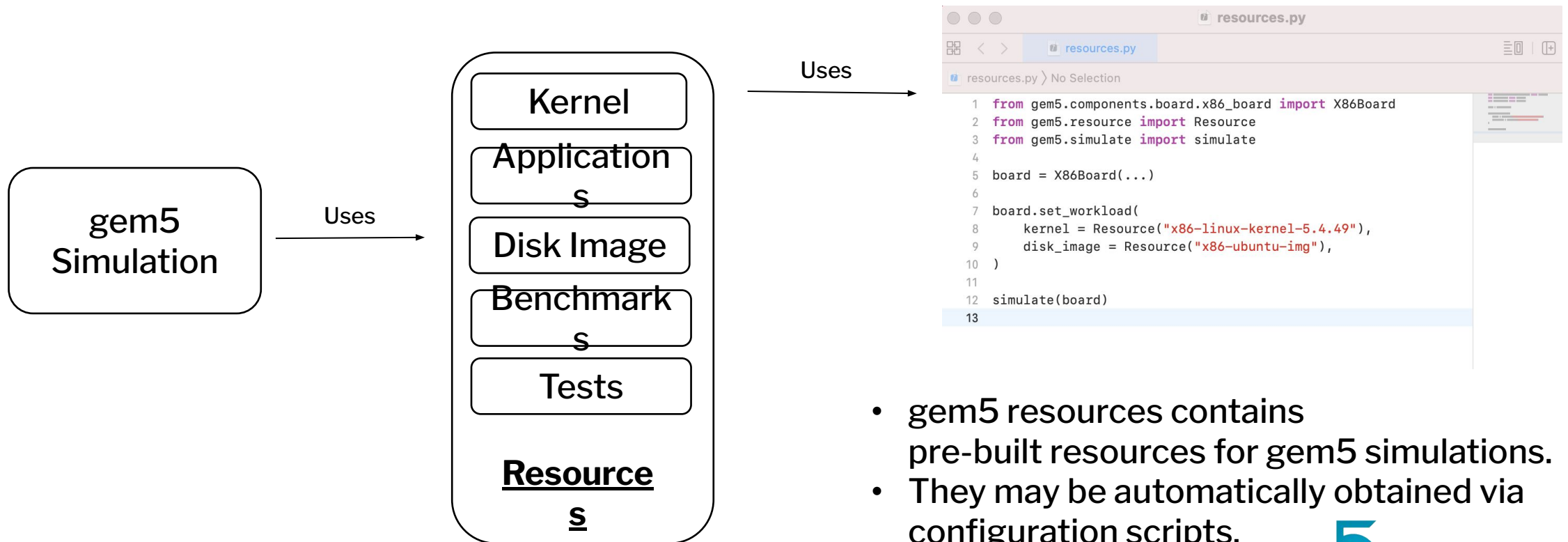


GEM5 COMPONENTS

```
gem5components.py
gem5components.py
gem5components.py > No Selection
1 from gem5.components.boards.x86_board import X86Board
2 from gem5.components.memory.single_channel import SingleChannelDDR3_1600
3 from gem5.components.processors.simple_processor import SimpleProcessor
4 from gem5.components.cachehierarchies.classic.private_l1_cache_hierarchy import (
5     PrivateL1CacheHierarchy,
6 )
7 from gem5.processors.cpu_types import CPUTypes
8
9 memory = SingleChannelDDR3_1600(size="3GB")
10 cache_hierarchy = PrivateL1CacheHierarchy(l1d_size="16kB", l1i_size="16kB")
11 processor = SimpleProcessor(cpu_type=CPUTypes.TIMING, num_cores=4)
12
13 board = X86Board(
14     clk_freq="3GHz",
15     processor=processor,
16     memory=memory,
17     cache_hierarchy=cache_hierarchy,
18 )
19
20 board.connect_things()
21 |
```

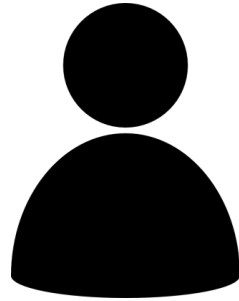
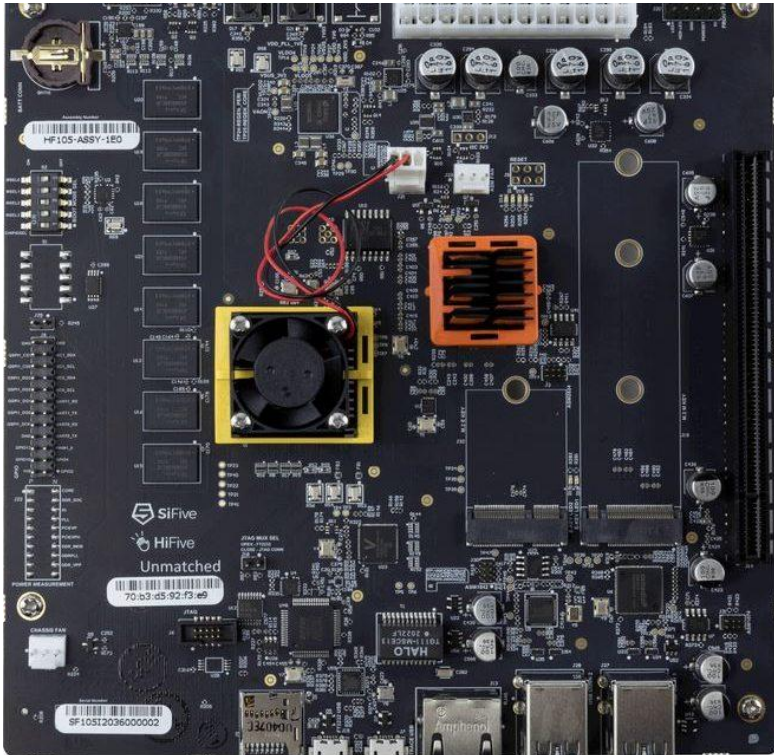


AND, GEM5 RESOURCES!



- gem5 resources contains pre-built resources for gem5 simulations.
- They may be automatically obtained via configuration scripts.

COMING SOON! "KNOWN GOOD CONFIGURATIONS"



Known Good
Configurations

```
sifive_board.py  
sifive_board.py > No Selection  
1 from gem5.known_good.unmatched_sifive import SiFive  
2 from gem5.simulate import simulate  
3 from gem5.resource import Resource  
4  
5 board = SiFive()  
6  
7 board.set_workload(  
8     Resource("riscv-ubuntu-boot")  
9 )  
10  
11 simulate(board)  
12 |
```