Architecture for Massively Parallel HDL Simulations

Rich Porter

Art of Silicon
Art of Silicon

- Founded in 2005
- Bristol based
- Multimedia centric Silicon IP
- Bespoke IP creation
- Consultancy
It's all about MONEY

- Engineer productivity is key to success
  - Employment cost is $$$s
  - Tooling cost
  - Computers
- Makes good business sense to improve productivity
  - Less time per chip
  - More chips per man year
- Productive engineers are happy engineers!
Coding Time

- Engineers do **NOT** spend all their time writing code
  - 5% writing code
  - 95% debugging code
  - Thinking & waiting for waves, regressions
- Highly desirable to quantify the effect of any design delta
  - As quickly & as easily as possible
  - For all engineers & management
Minimizing Idle Time

• Scalable test engine
  – 40+ instances per engineer
  – Up to as many as they can use
• Large compute farm of dense elements
  – 8, 12, 16, 24 core boxes
• Reduce iterations during the day
• Provide full regression bandwidth at night, weekends
• So how do I do this?
Verilator

- Used to generate standalone license-free test executable
- Execute as many as your compute infrastructure can support
- Deliver to other teams
  - Architecture
  - Systems
  - Toolchain
  - Customer
Signoff

• Event driven simulators are gold standard for signoff
  – Gate level simulations

• HAVE to run in this environment too

• Spending engineer time to create 2 environments costs $s
  – Gratuitous creation of extra work
Architecture

• Single testbench architecture
  – Ease of maintainence
  – Issue replication
  – Portable to silicon

• It is essential that test stimulus and cycle level behaviour is identical across platforms
  – Even though one platform is event driven and the other is cycle based
What AoS did

- **Stimulus & checker**
  - in C++
  - Identical code

- 'Gaskets' used
  - Provides uniform I/F
    - `interface->signal = value;`
    - `value = interface->signal->signed();`

- All design code in regular verilog
Example – verilog

```verilog
initial begin
    `ifdef verilator
        $c("aos_simctrl_init(&", rst_cnt,
            ", &", cycles,
            ", &", std_simctrl_finish_r, ");"
    `else
        $aos_simctrl_init(
            rst_cnt,
            cycles,
            std_simctrl_finish_r
        );
    `endif
end

always @(posedge clk1)
    `ifdef verilator
        $c("aos_simctrl_clk();");
    `else
        $aos_simctrl_clk;
    `endif
```
Autogeneration

- Single sourced from SPIRIT XML description
  - Verilog
    - module declarations
    - grey box instantiation
    - verilog side gasket instantiation
  - C++
    - headers
    - port descriptions (name, size, direction)
    - PLI wrappers
Test Configuration

• Tests were configured at execution using `plusargs`:
  - `+argument=value`

• Scanning code was identical for both:
  - To ensure consistency

• Regression script
  - Created these configurations
  - Passed them to batch scheduler
  - Collated results for web presentation
Logging

• Unified logging interface
  – Consistent output
  – Captured in markup to preserve semantics
  – No cludgy parsing of text post mortem that varies from platform to platform

• Test fail produces identical logs across platforms
  – No problems with spaces, split lines, mixed streams, output ordering
Triage

• Group messages by content, code, file, line
  – Placing most numerous first

• Order by cycles within
  – Increasing cycle count to failure

• Failure at the top is debug candidate

• Web I/F can provide rich set of filters/collation options
Performance

- Performance has steadily increased

- Memory footprint remains low
  - No simulation kernel
Key Features

- Single testbench providing cycle accurate stimulus
- Autogeneration from single source
- Single test configuration
- Unified logging
Conclusion / AoS Experience

- Little spent waiting for simulations
  - 60 wide queue gave 6MHz effective
  - Engineers spent time really debugging
- Small amount of time spend bringing up each simulator
  - Not much, but was very early on
- Verilator proved robust
- See no reason why cannot scale to 1000s of simulations
Recommendations

- Get a compute farm
- Use a queue
- Replace CPUs regularly
- Prioritize high value jobs
  - $ licenses
  - interactive jobs
- Evaluate what verilator can do for you
- Profile, record and report
Questions

- Questions

Rich Porter
rich.porter@artofsilicon.com