An Extensible Framework for At-Speed Evaluation of Arithmetic Hardware

Zifan Wang

Supervisor: Dr. James Davis

Imperial College London

June 26, 2019

Background

Motivation

- Started as a specialised evaluation system for high-radix online arithmetic units
  - At-speed (Overclockable)
  - Precision Checking

Motivation

- Started as a specialised evaluation system for high-radix online arithmetic units
  - At-speed (Overclockable)
  - Precision Checking
- Digital designers all use their own testbenches

Motivation

- Started as a specialised evaluation system for high-radix online arithmetic units
  - At-speed (Overclockable)
  - Precision Checking
- Digital designers all use their own testbenches
- Ad hoc, one-time use, inefficient

Motivation

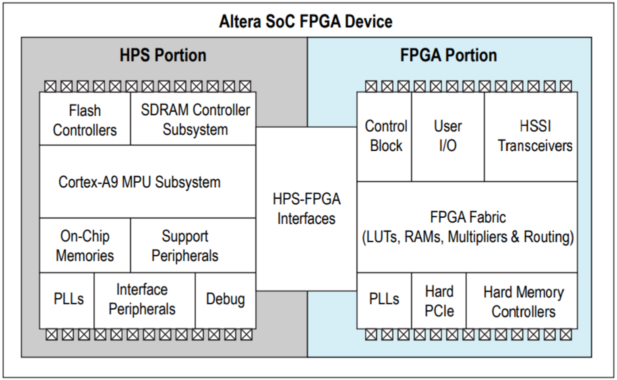
- Started as a specialised evaluation system for high-radix online arithmetic units
    - At-speed (Overclockable)
    - Precision Checking
- Digital designers all use their own testbenches
- Ad hoc, one-time use, inefficient

- Propose an extensible framework
    - With variable frequency with a high maximum (Assume DUT @300MHz)
    - Extensible
    - User-friendly

Design & Implementation

Hardware Choice

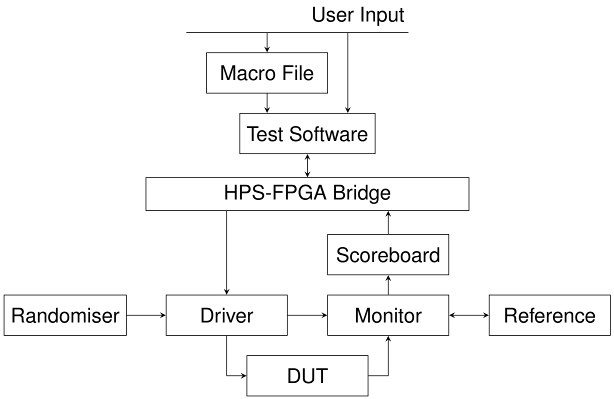
Cyclone V SX SoC Development Board



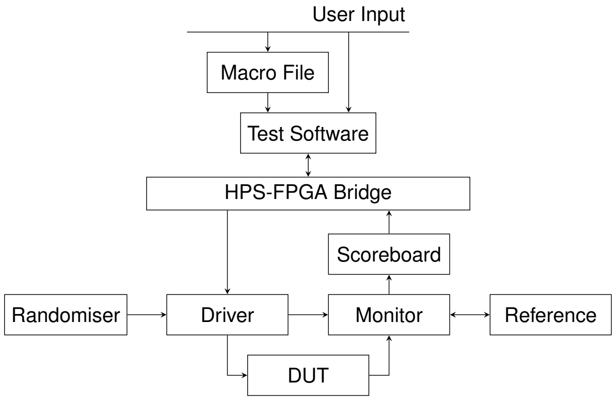
- HPS – user interaction and test control
- FPGA – actual hardware testing

System Architecture

- Inspired by UVM agent
- Modular, thus extensible

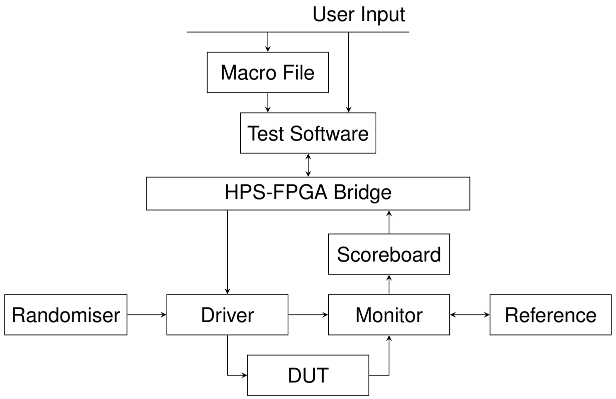


System Architecture



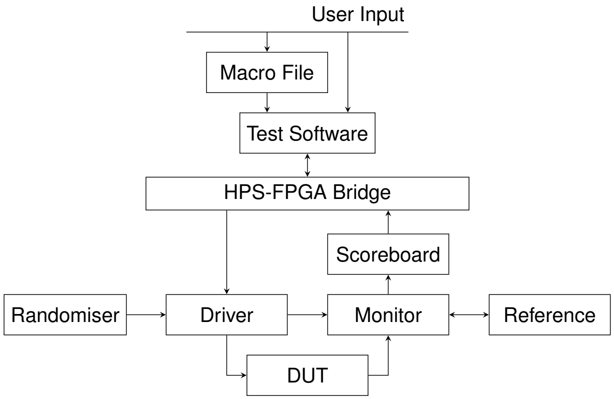
- Software accepts user commands from files or command line

System Architecture



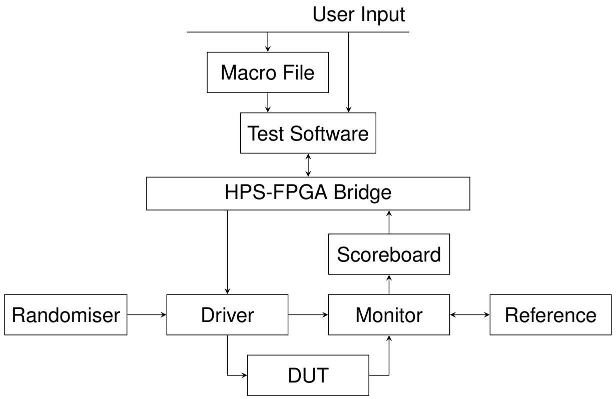
- Randomiser generate random data with LFSRs

System Architecture



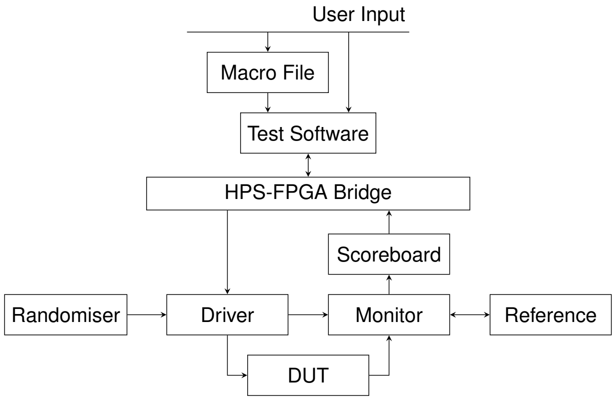
- Driver filters inputs and drives them to the DUT and the monitor

System Architecture



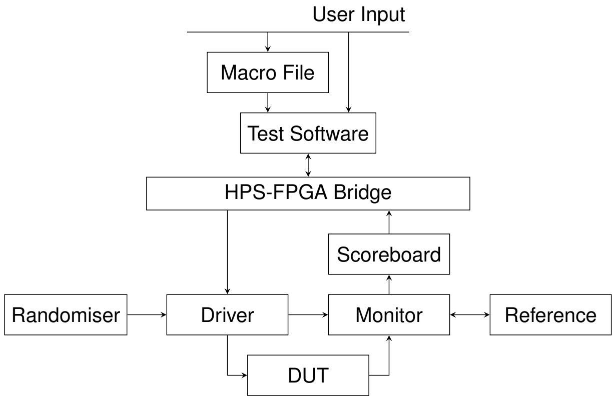
- Monitor watches for the differences between DUT and reference outputs

System Architecture



- Scoreboard tallies them and provides results to software

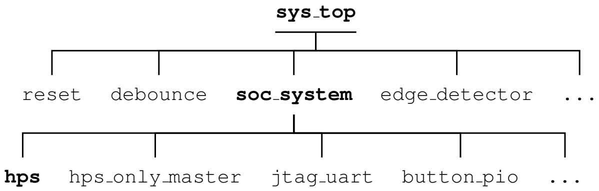
System Architecture



- Software reads results and present them to user

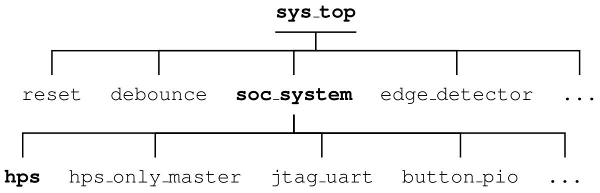
Hardware Project Hierarchy

- Adapted from a golden system reference design



Hardware Project Hierarchy

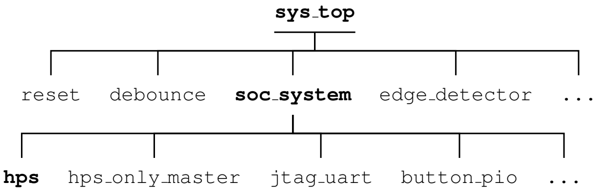
- Adapted from a golden system reference design



- Already uses *Qsys* for `soc_system`

Hardware Project Hierarchy

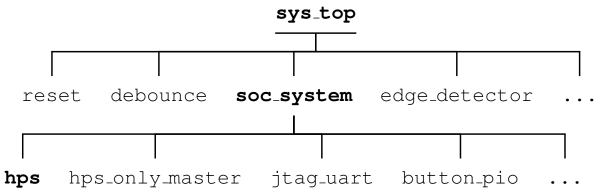
- Adapted from a golden system reference design



- Already uses *Qsys* for `soc_system`
- Frequency control uses `pll` (Phase-locked loop) and `pll_reconfig` IP, easy integration with *Qsys*.

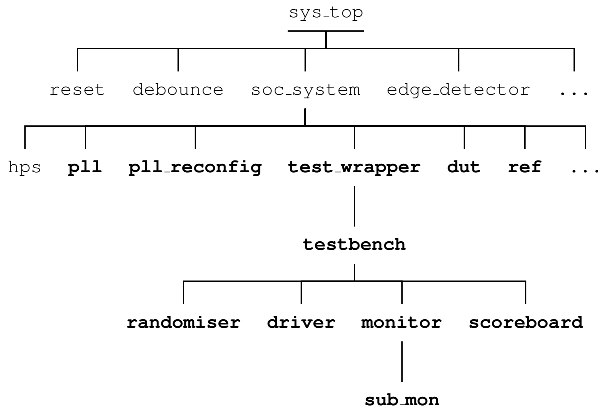
Hardware Project Hierarchy

- Adapted from a golden system reference design



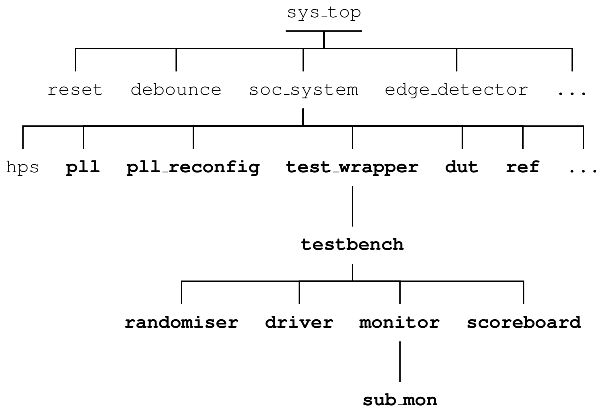
- Already uses *Qsys* for `soc_system`
- Frequency control uses `pll` (Phase-locked loop) and `pll_reconfig` IP, easy integration with *Qsys*.
- Great interface for user to connect their design and reference modules

Hardware Project Hierarchy



- Access to software through `hps`

Hardware Project Hierarchy



- Access to software through `hps`
- Access to hardware design through `dut` and `ref`

Providing test data

- Assume DUT is 32-bit @300MHz, need sustained data input for stress testing

Providing test data

- Assume DUT is 32-bit @300MHz, need sustained data input for stress testing

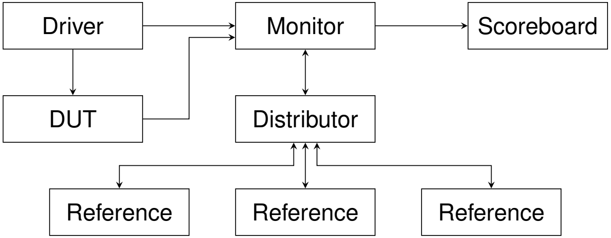
- Real time, on board generation of random test data
    - Low effort, significant coverage, can discover subtle errors
    - Include filters to give user control
    - Accept manual inputs from users for critical path

Relaxed Reference

- Monitors needs reference module to check correctness
- Has to be functionally correct
- Sensible to have reduced timing requirements
- Harder for users to make mistakes

Monitor Structure

- Parallel Monitor
  - Checks all data points in parallel



Software Design

- Requirement
  - Manual and auto input controls
  - Test duration
  - PLL frequency

Software Design

- Requirement
  - Manual and auto input controls
  - Test duration
  - PLL frequency
- Considered solutions
  - Test configuration files
    - Easy to build and scale
    - Slightly harder to use and manage
  - Interactive REPL system
    - Intuitive to control
    - Easy to automate and scale

Results

Results

- Maximum frequency
  - TimeQuest: 394.01MHz
  - Hardware test: Stable at 400MHz; breaks at 425MHz
  - DUT initially assume to work at 300MHz

Results

- Maximum frequency
  - TimeQuest: 394.01MHz
  - Hardware test: Stable at 400MHz; breaks at 425MHz
  - DUT initially assume to work at 300MHz
- User-friendliness
  - Performed an OOTB Testing
  - Knowledge of digital designs and testbenches
  - No previous knowledge on Qsys or the framework
  - Obtained results in 2 hours
  - No major interruptions, positive feedback

Results

- Maximum frequency
  - TimeQuest: 394.01MHz
  - Hardware test: Stable at 400MHz; breaks at 425MHz
  - DUT initially assume to work at 300MHz
- User-friendliness
  - Performed an OOTB Testing
  - Knowledge of digital designs and testbenches
  - No previous knowledge on Qsys or the framework
  - Obtained results in 2 hours
  - No major interruptions, positive feedback
- Flexibility

Flexibility

Item	Reconfigurability
`WIDTH`	$\leq$32 bits
`NUM_SUB_MON`	$\geq$2
f_{dut}	$\leq$400MHz
DUT I/O	2 in 1 out
DUT delay	All values
bitset/bitclr	All values
manual	All values
time	All values

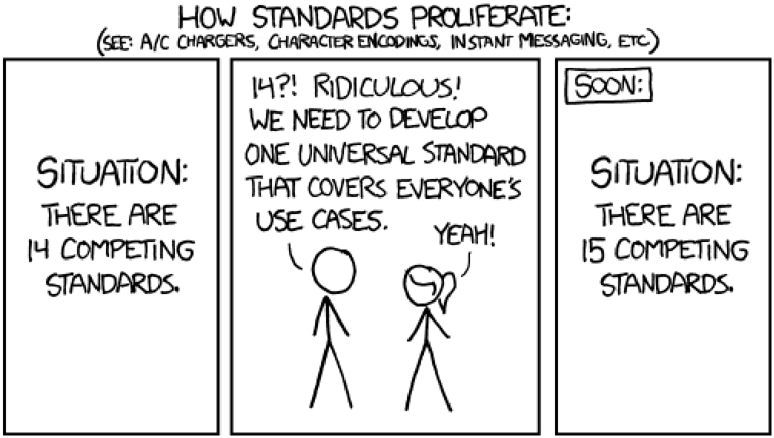
Demo

- Shows the configuration process
- Shows software interaction

Command	Explanation
`reset`	Resets the system and test results.
`version`	Prints the system version.
`freq <speed>`	Sets the clock speed to the specified value in MHz. Prints the actual frequency configured.
`mode <m|a>`	Choose between manual and auto test mode.
`manual <a|b> <hex>`	Give input in manual mode.
`bitset <a|b> <hex>`	Force bits to be 1 in auto mode.
`bitclr <a|b> <hex>`	Force bits to be 0 in auto mode.
`run <time>`	Runs the test for the duration specified in seconds. Prints the results at the end of the test.
`exit`	Exits the REPL.

Evaluation

What have we done?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

xkcd.com/927

Evaluation

- Limitations
    - Customisability of current implementation

Evaluation

- Limitations
  - Customisability of current implementation
- Improvements
  - User experience can be made better with a unified software system + Verilog preprocessor
  - Set up a more powerful HPS-FPGA communication system to allow more insightful results

Evaluation

- Limitations
  - Customisability of current implementation
- Improvements
  - User experience can be made better with a unified software system + Verilog preprocessor
  - Set up a more powerful HPS-FPGA communication system to allow more insightful results

- Not limits to the extensibility of the framework

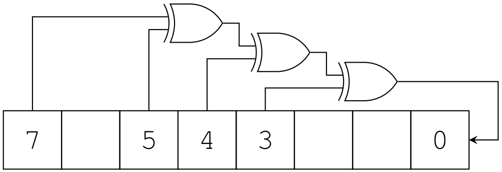
Thank you

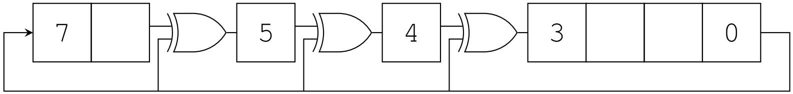
Questions?

Providing test data

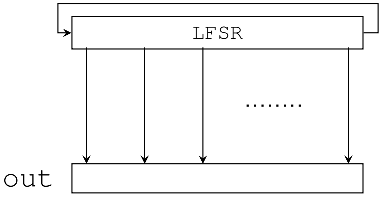
- Assume 32-bit @600MHz, need sustained 19.2Gbps for stress testing
- HPS-FPGA bridge
  - Assume perfect packing and always burst transfer
  - 128-bit @133MHz, 17.0Gbps
- Off-chip SDRAM
  - Assume always burst, access pipelined
  - 32-bit DDR3 @400MHz, 25.6Gbps
  - Sufficient, but needs time to fill up
  - SDRAM controller can be complex to use and manage
- On-chip memory
  - Assume widest possible arrangement
  - 768.9kB @315MHz, 242Gbps
  - Very fast, but extremely small capacity for sustained load
- Real time generation

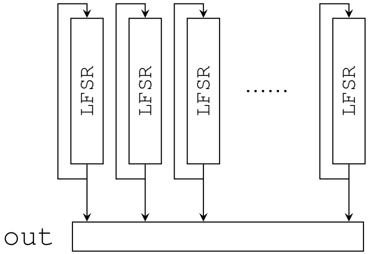
LFSRs



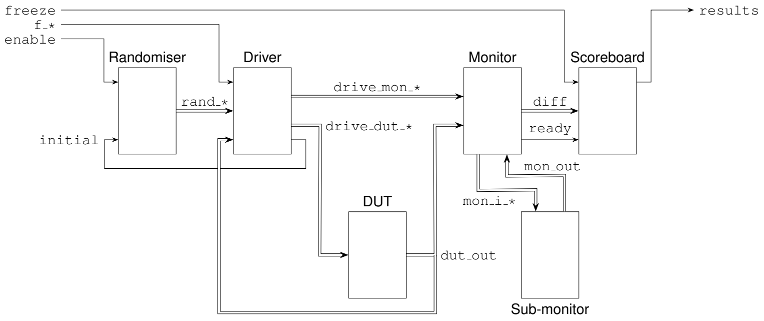


Randomiser Structure



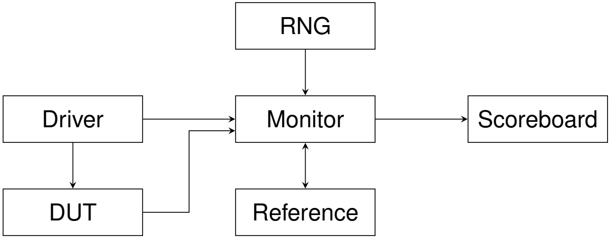


Hardware


freeze
f_*
enable
initial
Randomiser
rand_*
Driver
drive_mon_*
drive_dut_*
DUT
dut_out
Monitor
diff
ready
mon_out
mon_i_*
Sub-monitor
Scoreboard
results

Monitor Structure

- Lazy Monitor
  - Randomly selects data points to check



Software Design

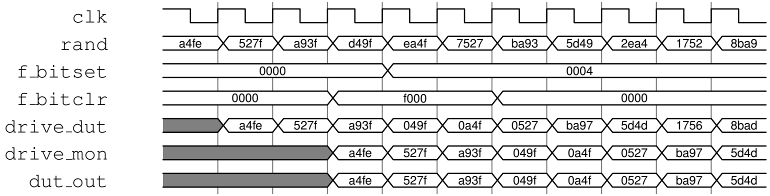
```
mode: auto
bitset:
  a: 00000001
  b: 00000000
bitclr:
  a: 00000000
  b: 00000001
input:
  - a: 00000000
    b: 00000000
freq: 200
runtime: 60
```

```
> mode auto
> bitset a 00000001
> bitclr b 00000001
> freq 200
PLL Configured to 200.00MHz
> run 60
Results: ...
```

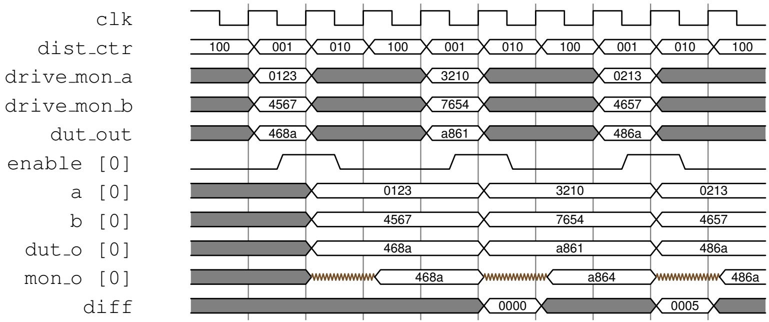
REPL Commands

Command	Explanation
`reset`	Resets the system and test results.
`version`	Prints the system version.
`freq <speed>`	Sets the clock speed to the specified value in MHz. Prints the actual frequency configured.
`mode <m|a>`	Choose between manual and auto test mode.
`manual <a|b> <hex>`	Give input in manual mode.
`bitset <a|b> <hex>`	Force bits to be 1 in auto mode.
`bitclr <a|b> <hex>`	Force bits to be 0 in auto mode.
`run <time>`	Runs the test for the duration specified in seconds. Prints the results at the end of the test.
`exit`	Exits the REPL.

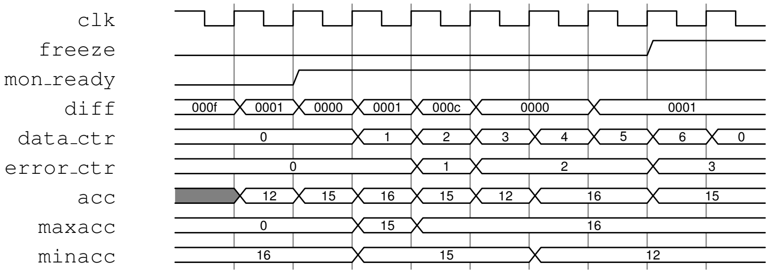
Driver


clk
rand
f_bitset
f_bitclr
drive_dut
drive_mon
dut_out
a4fe
527f
a93f
d49f
ea4f
7527
ba93
5d49
2ea4
1752
8ba9
0000
0004
0000
f000
0000
a4fe
527f
a93f
049f
0a4f
0527
ba97
5d4d
1756
8bad
a4fe
527f
a93f
049f
0a4f
0527
ba97
5d4d
a4fe
527f
a93f
049f
0a4f
0527
ba97
5d4d

Monitor


clk
dist_ctr
100
001
010
100
001
010
100
001
010
100
drive_mon_a
0123
3210
0213
drive_mon_b
4567
7654
4657
dut_out
468a
a861
486a
enable [0]
a [0]
0123
3210
0213
b [0]
4567
7654
4657
dut_o [0]
468a
a861
486a
mon_o [0]
468a
a864
486a
diff
0000
0005

Scoreboard


clk
freeze
mon_ready
diff
000f
0001
0000
0001
000c
0000
0001
data_ctr
0
1
2
3
4
5
6
0
error_ctr
0
1
2
3
acc
12
15
16
15
12
16
15
maxacc
0
15
16
minacc
16
15
12