

---

# Fast Simulation and Equivalence Checking using OpenAccess

---

University of **M**ichigan

Kai-hui Chang, **David A. Papa**, Igor L. Markov and Valeria Bertacco  
{changkh, iamyou, imarkov, valeria}@umich.edu

---

# Outline

- IWLS 2006 Programming Challenge
  - OpenAccess Gear
  - Fast simulation
    - Oblivious vs. Event Driven
  - Equivalence checking with simulation signatures
  - Incremental Verification
  - Custom vs. Native implementations
  - Graphical User Interface extensions
  - Plug-In interface
-

---

# IWLS 2006 Programming Challenge

- Logic optimization student programming competition
  - Must be implemented on OpenAccess and should make use of OAGear infrastructure
  - Judged according to correctness, efficiency, importance, design, coding style, etc.
  - 1<sup>st</sup> place was given to two teams
    - Sat Sweeping package
    - Fast simulation and equivalence checking
  - Both entries are now part of OAGear
-

---

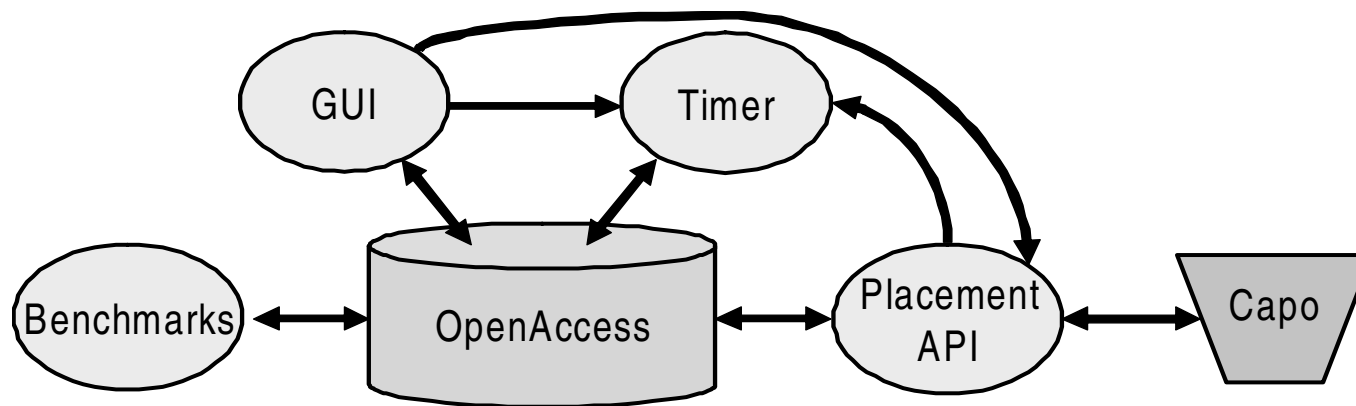
# OpenAccess Gear



- Release useful tools and libraries to enable research
  - Make OpenAccess a useful platform for academia
  - Provide common infrastructure for research and benchmarking
  - Adopt an open source development model
  - Initiated and supported by Cadence Design Systems
-

# Early OAGear Overview

- **Focus on four main components**
  - **GUI: Layout and Schematic Viewer**
  - **Static Timing Analysis**
  - **Generic Standard Cell Placement Interface: Capo API**
  - **Benchmarks in OpenAccess Format**



# Static Timing Analysis: OAGear Timer

- Built on OpenAccess for integration into other tools, e.g. placement
- Two modes: Full timing analysis and incremental timing analysis
- Different models for wires: No wire delay, bounding box model; can be extended easily to more accurate models
- Library formats: Cadence .tlf and Synopsys .lib
- Timing constraints: Subset of .sdc constraints
- Standardized timing reports
- Detailed documentation

```
CELL(DFFX1
...
TIMING_Model(7x7
(Spline
(Load_AXIS ...)
(INPUT_SLEW_AXIS ...)
data(...))
)
...
Path(CK => Q ... ..)
...
Setup(D => CK ... ..)
...
)
```

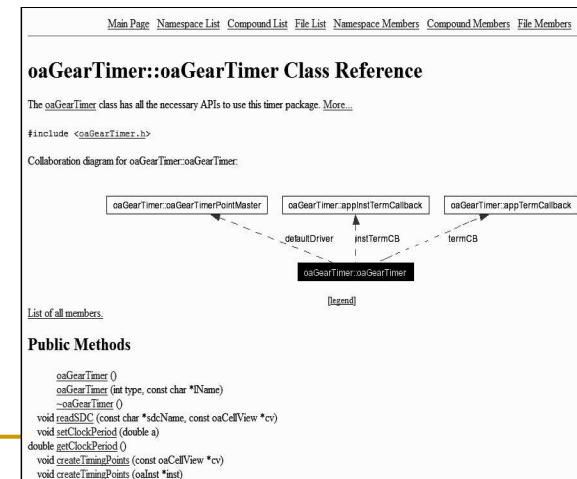
.tlf

```
Cell (DFFX1) {
...
pin(Q){
...
timing() {
related_pin : "CK";
timing_sense : non_unate;
timing_type : rising_edge;
cell_rise(7x7) {
index_1 ("... ..");
index_2 ("... ..");
values { ... ..};
}
}
}
```

.lib

```
#SDC constraint file
create_clock
-period 1 [get_ports {CK}]
set_input_delay 0.04
-clock CK [all_inputs]
set_output_delay 0.02
-clock CK [all_outputs]
set_driving_cell
-lib cell INVX2 [get_ports {G*5}]
set_load 0.01 [get_ports {G2*}]
```

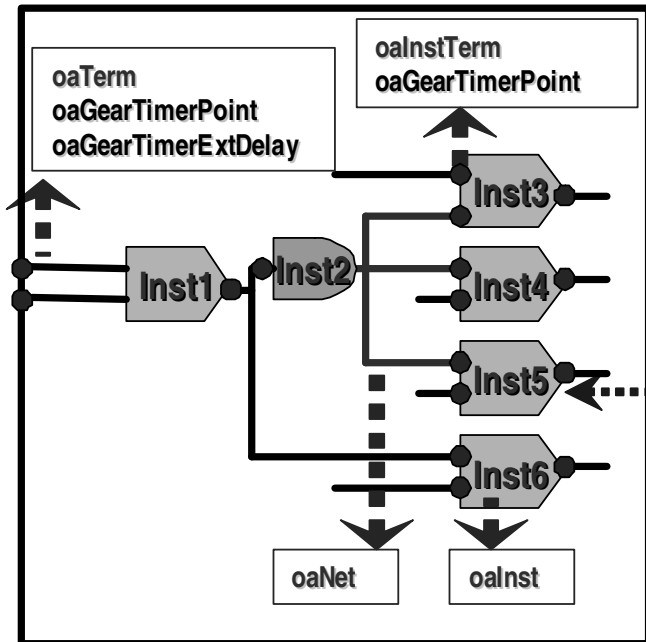
.sdc



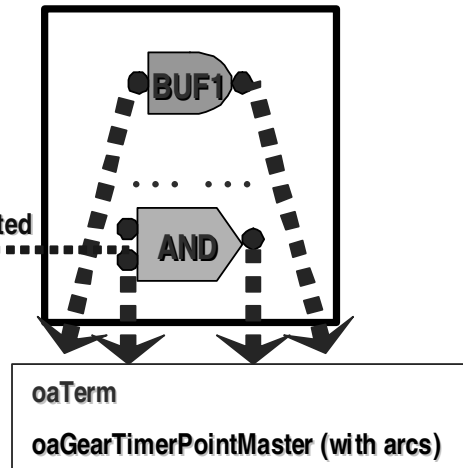
Documentation

# Static Timing Analysis: OAGear Timer

## Design



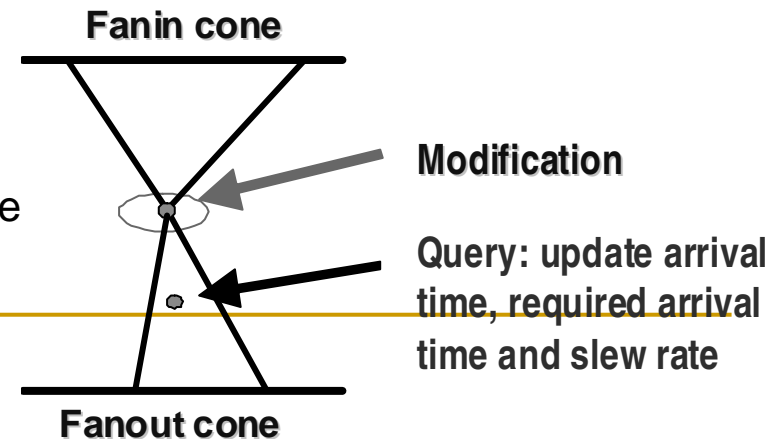
## Standard Cell Library



The timing information is stored using the OpenAccess extension mechanism (oaGearTimerPoint, ...)

## Incremental timing analysis

- **When a modification occurs:**
  - **Mark the required arrival time of nodes in the fan-in cone invalid**
  - **Mark the arrival time of nodes in the fan-out cone invalid**
- **Later if there is a query, update the timing information**



---

# Capo Placement API

- Open source placement tool
    - Maintained at U. of Michigan
    - <http://vlsicad.eecs.umich.edu/BK/PDtools/>
  - Extended to use OpenAccess by the OAGear CapoWrapper package
    - Reads design data directly from OpenAccess
    - Builds appropriate Capo data structures in memory
    - Reads back Capo data structures and writes results to OA
  - Example of integrating large mature programs
    - Porting to a native codebase would require extraordinary effort
-

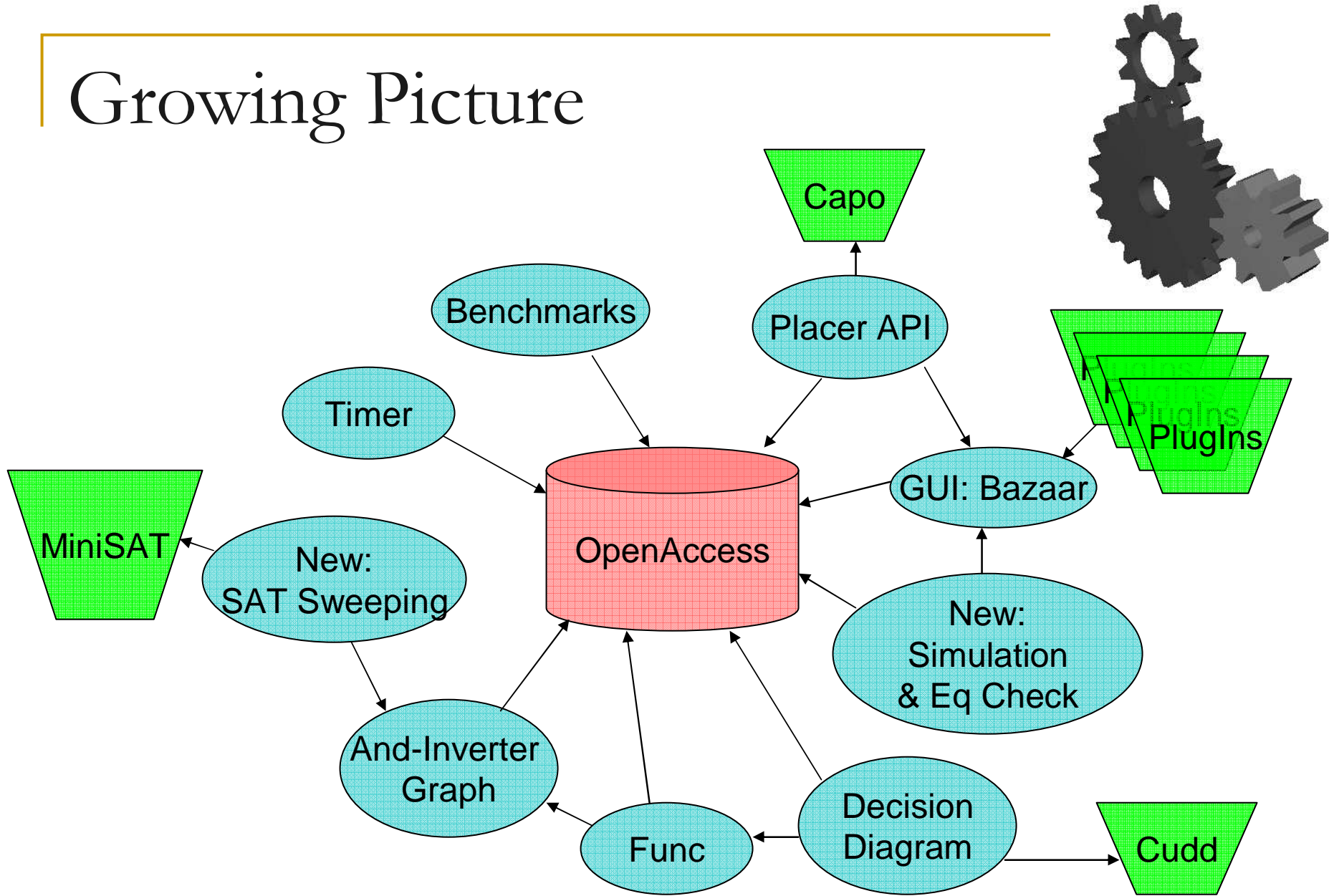
---

# GUI: Bazaar

## **Technical Capabilities**

- Easy to read and extend, built on Qt and OpenGL
  - In the style and spirit of the OpenAccess standard
  - Layout Editor displays block domain design data directly from database
  - Schematic Editor displays module domain design's logical connectivity
  - Controller operates Capo API for on-demand placement
  - Fast OpenGL rendering scales to very large designs
  - “oaRegionQuery” accesses only relevant portions of the design
-

# Growing Picture



---

# Fast Simulation

- Bit-parallel simulation
    - 32 or 64 patterns simulated simultaneously
  - Special cases for common gate types
    - Compiler can use CPU instructions to implement AND, OR, etc.
  - Levelize the circuit for faster topological traversals
-

---

# Simulation Algorithms

- Event-driven algorithm
    - Evaluates only gates with events
    - Suitable when the number of events is small
  - Oblivious algorithm
    - Evaluates all the gates
    - Avoids overhead of event scheduling
    - Suitable for random simulation
-

---

# Equivalence Checking With Simulation Signatures

- Easy to disprove equivalence with counter-examples from simulation
  - Signatures mismatch => not equivalent
  - Signatures match => need more testing
- SAT-based equivalence checking is performed when signatures match
- Counter-examples are returned
  - To help understand the discrepancy

[J. Zhang et al. "Simulation and Satisfiability in LogicSynthesis", *IWLS* 2005]

---

---

# Incremental Verification

- User specified set of gates to EQ check
    - Still need to add GUI support for this feature
  - Use fast simulation to define Similarity Factor between two netlists
    - $(\text{Matching signals}) / (\text{number of signals})$
    - Signals are matching if simulation signature appears in both circuits
    - Small Similarity Factor means potential discrepancy
    - Reported in the GUI equivalence checker
-

---

# Custom Vs. Native Implementation

- Simulator originally used custom data structures and file I/O
  - Ported it to run natively on OpenAccess
  - Observed significant slowdown in native impl.
    - Primarily due to oaAppDef lookup time
  - Spent significant effort optimizing native impl.
    - Converted oaAppDef uses to `std::hash_map<oaNet*>`
  - Unable to match custom impl. performance
-

# Custom vs. Native Experimental Data

- Asymptotic improvement over original simulator
- Both new simulation algorithms scale linearly
- Both scale to realistic circuit sizes
- Custom data structures perform better for large inputs

Benchmark	Gate count	Simulator runtime (sec)			EQcheck runtime (sec)
		OAGear orig.	Our simulators custom	native	
s27	19	9.8e0	0.4	0.4	0.3
s344	132	4.0e1	0.4	0.7	0.6
s1196	483	9.5e1	0.5	1.7	1.6
s15850	685	5.0e3	0.6	2.0	1.8
s9234_1	974	2.4e3	0.7	2.9	2.8
s13207	1218	1.7e4	0.8	3.3	3.4
s38417	8278	3.0e5	2.0	21.0	1.4e2
vga_lcd	124031	time-out	20.2	3.3e2	2.5e4

# Graphical User Interface Elements

The screenshot shows the OAGear Bazaar GUI. The main window has a menu bar (File, View, Place, Windows, Help, MSim) and a toolbar. A 'Simulation Data' window is open, displaying a table for the 'Next Cycle'.

Next Cycle		
	1	2
1	G17	0xffffffff9ffffffd

Below the table is a command prompt window with the following text:

```
> run_cycle_simulator  
> simulate -lib simEquiTests -cell s27 -liberty gsclib.lib -useOpt -patternFile /z/h  
Reading Liberty file gsclib.lib
```

A 'Send Command' button is located at the bottom right of the command prompt area. The status bar at the bottom left shows 'Ready'.

The screenshot shows the OAGear Bazaar GUI with the 'Equivalence Check Results' window open. The window displays configuration parameters and a counter example table.

Configuration parameters:

- Lib1:
- Lib2:
- Cell1:
- Cell2:
- Net1:
- Net2:
- Equivalent:
- Similarity Factor:

Counter Example table:

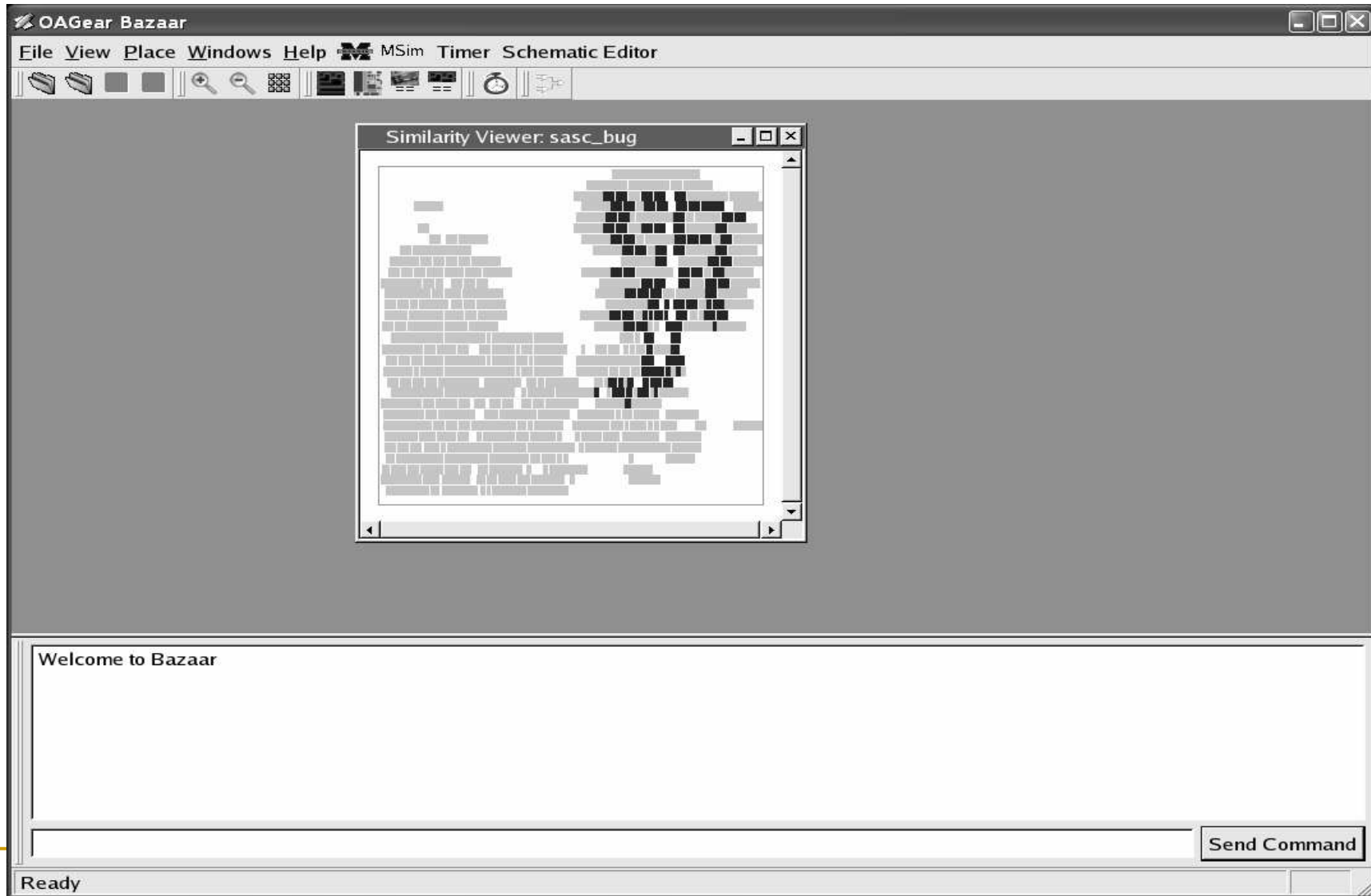
	1	2	3	4	5	6	7	
1								
2	1	blif_clk_net	blif_clk_net	0x2afbc22	G17	0xdf37fed7	G17	0xfda3f272
3	2	blif_reset_net	blif_reset_net	0xbda068e7	n_16	0x1d10f2d5	n_16	0x3d00f270
4	3	G0	G0	0x3d10f2f5	n_12	0x20c80128	n_12	0x25c0d8d
5	4	G1	G1	0x8f75fddd	n_11	0x8d75f95d	n_11	0x708a0222
6	5	G2	G2	0x2000480				
7	6	G3	G3	0x76de1ebf				
8	7	G5	G5	0x54033252				
9	8	G6	G6	0x744961ca				
	9	G7	G7	0x0				

Below the table is a command prompt window with the following text:

```
> check_design_equivalence  
> check_equivalence -lib simEquiTests -lib2 simEquiTests -cell1 s27 -cell2 s27_bug -randomPattern 1024 -liberty gsclib.lib
```

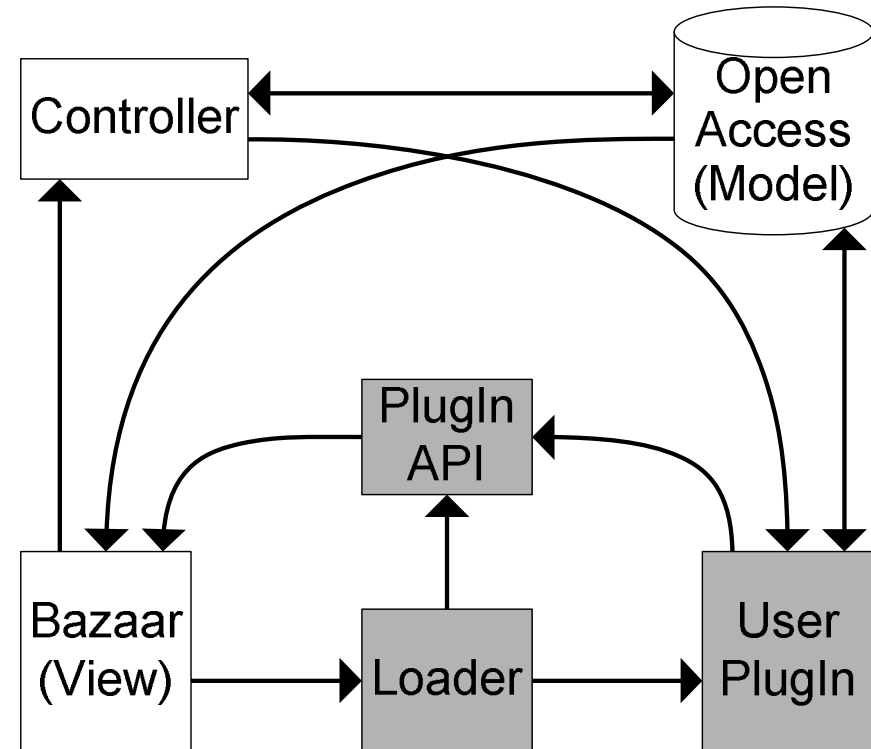
A 'Send Command' button is located at the bottom right of the command prompt area. The status bar at the bottom left shows 'Ready'.

# Similarity Layout View



# Plug-In Interface for OAGear Bazaar

- Encapsulates Bazaar's interface
- Greatly simplifies extending Bazaar
- Facilitates dynamic loading of user code
- Decouples user code from Bazaar



---

# Conclusions

- Identified a major component of OAGear with poor scalability
  - Wrote new logic simulation engine using different algorithms
    - Reduces runtime by up to 100 times in our experiments
    - Asymptotic improvement
  - Leveraged simulator to speed up equivalence checking
  - Defined new metric of circuit similarity useful in incremental verification and debugging
  - Extended OAGear's graphical user interface, Bazaar
    - Implemented and evaluated several use-cases
    - Designed new infrastructure for creating user plug-ins
-

---

# Future Work in OAGear

- Convert existing GUI tools to Plugins
    - Layout Editor
    - Schematic Editor
  - More Plugins!
    - Waveform viewer for our logic simulator
    - User contributions... (hint, hint!)
  - Develop a communication mechanism between Plugins
  - Further integration of OAGear Tools into the GUI
    - Buffer Insertion, Timer, Sat Sweeping, etc.
  - More complete Tcl API for OAGear utilities
  - Possibly an OAGear Router
  - Ease of use improvements
-

---

Thank You!

Questions?

More screen shots...

---

