A Library-Based Approach to Portable, Parallel, Object-Oriented Programming

Interface, Implementation, and Application

Steven Parkes
Sierra Vista Research
Santa Clara, California

John A. Chandy, Prithviraj Banerjee
University of Illinois
Urbana, Illinois

Overview

- Motivation
- Model and Interface
- Implementation
- Application study

Overview

- Challenges:
  - Large applications
  - Concurrent applications
- A significant number of problem domains do not fit the models of existing techniques
- Qualities of results of many EDA applications are strongly dependent on magnitude of applied resources

Motivation

- Challenges:
  - Large applications
  - Concurrent applications
- A significant number of problem domains do not fit the models of existing techniques

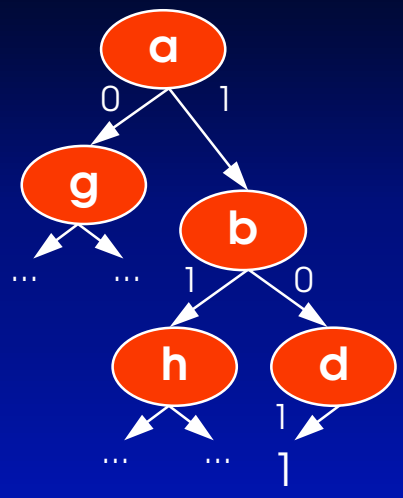
Electronic Design Automation

- Used to design next generation products (including HPC platforms)
- $2 billion/year market
- Many EDA applications run for hours, days, and sometimes weeks
- Qualities of results of many EDA applications are strongly dependent on magnitude of applied resources

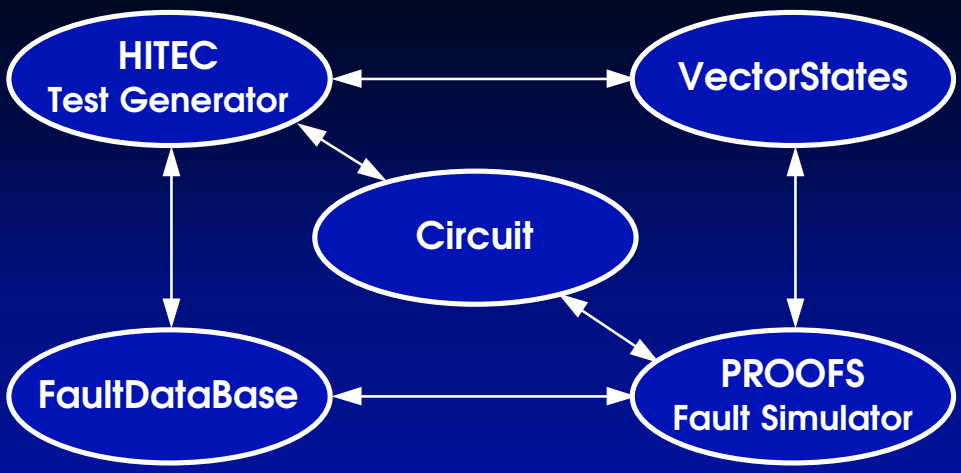
An Example: Automatic Test Pattern Generation

- Used to test chips for manufacturing errors
- Conceptually a simple depth-first search problem
- Challenges:
  - Parallel search
  - Issues unique to ATPG

f = a(bh + d) + g
g = ...
h = ...



Serial ATPG: HITEC/PROOFS



Serial ATPG to Parallel ATPG

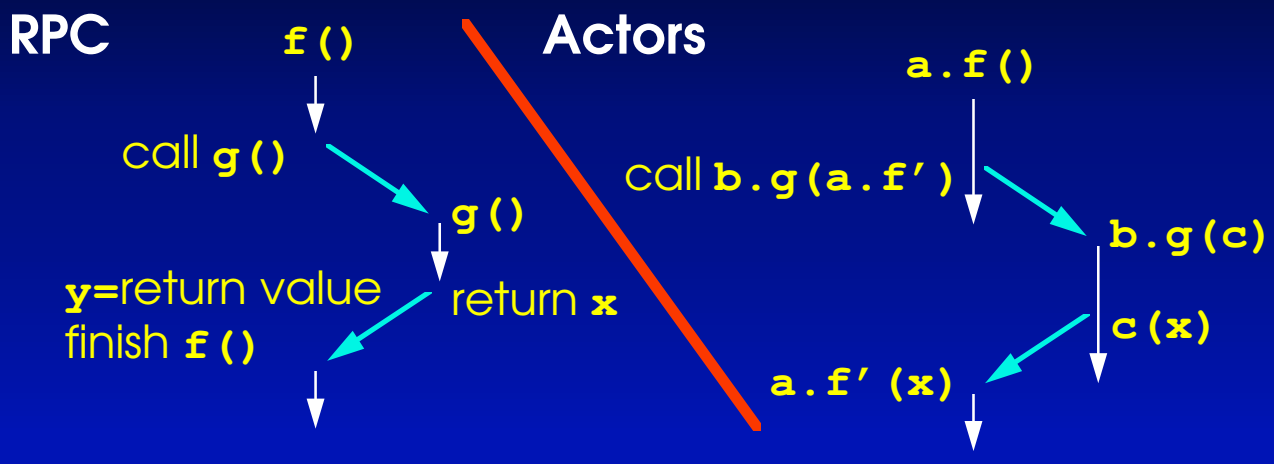
- Existing language: *C++*
- Existing tools: *run-time library*
- Flexibility: *multiple levels of abstraction*
- Incremental parallelization and co-development with serial application: *object-oriented techniques*

Concurrency Models

- Requirements
  - Mechanism for expressing parallelism
  - Mechanism for limiting state interference
- Data parallelism
  - Concurrent operations on distributed data structures (arrays)
- Task parallelism
  - Concurrent evaluation of loop iterations
  - Concurrent expression evaluation

Actor Model

- Actors: concurrent objects (Hewitt and Agha)
  - Actor objects express concurrency implicitly
  - Each actor has single light-weight thread
  - Inter-thread communication via continuations



Actor Model Implementations

- Pure Actor Languages: *Lisp-based*

  Act and ABCL languages, Sal, HAL, CA/Concert
- C/C++-based Languages

  Charm, Charm++
  - Metaprogrammability

  Mentat

  ICC++
- Libraries

  ACT++, CLAP

Actor Interface

- Actor classes created via derivation from library-supplied **`Actor`** class

Serial:

```
class Window { ... };
```

Parallel:

```
class Window : public Actor { ... };
```

Serial/Parallel:

```
class TestGenerator: public Actor,
                     public Window { ... };
```

- Incremental parallelism/co-development

Communication via Continuations

- Concurrency expressed via continuation calls on actor objects

Serial:

```
vectorDB.test(list);
```

Parallel:

1: Declare continuation **`c`** which will call **`test`** member on actor **`vectorDB`**:

```
test::Continuation c ( vectorDB );
```

2: Call **`c`** with value **`list`**:

```
c( list );
```

Actor Methods

Serial:

```
void test( struct VectorList );
```

Parallel:

```
void test( struct VectorList );
class test : public ActorMethod<VectorList>
             { ACTORMETHODOF(VectorList); };
```

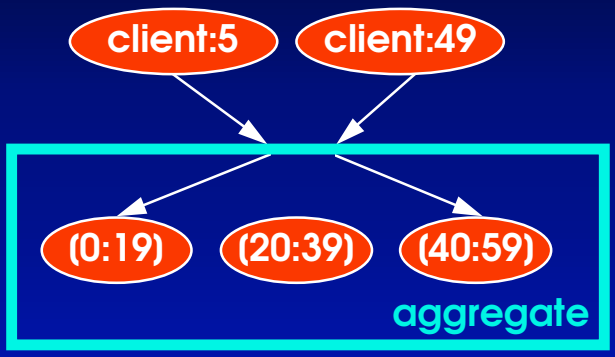
- Incremental parallelism: members may be called normally or via continuations
- All calls statically type checked
- Support for both static and dynamic binding

Collection Abstraction

- Aggregates: concurrent collections (Chien)
  - Collections of actors which share a common name
  - Unserialized, multi-access interface

Example: a distributed array:

Actor implementation



Aggregate implementation

Aggregate Interface

- **`Aggregate`** base class, derived from **`Actor`** base class
- Support for application-specific customization:
  - **`Distribution`** types to describe number and placement of constituent actors
  - Programmable representative selection

Metaprogrammability

- Application-specific customization of run-time library operations
- Local metaprogrammability: *per-class*
  - Prioritized continuation scheduling
  - Aggregate distribution/representative selection
- Global metaprogrammability: *per-application*
  - Debugging
  - Fault tolerance

Actor Interface Implementation: Abstract Parallel Architecture

- Actor Interface requires low-level services:

  Thread — Memory

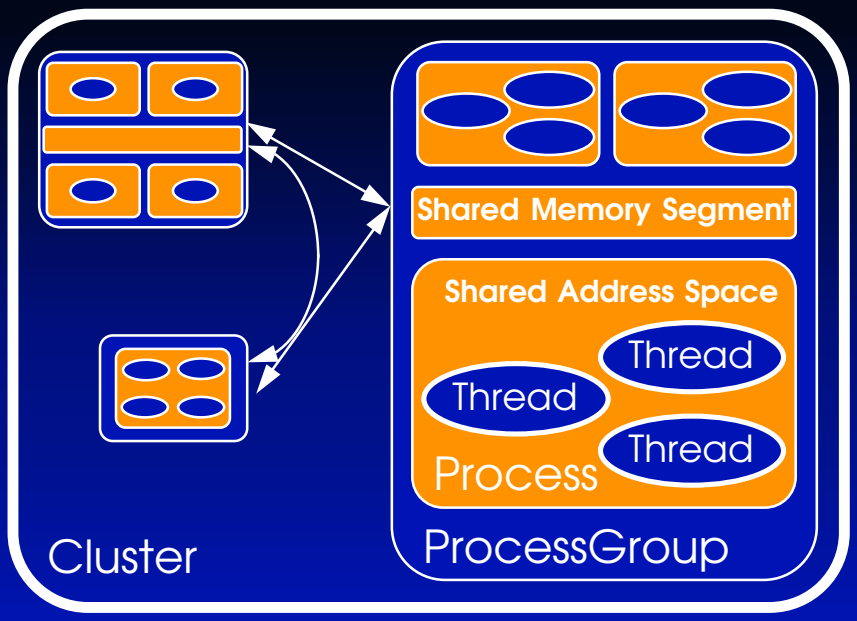
  Communication — Configuration

- APA provides integrated set of management classes
- Open implementation
- AIF supports most common architecture-specific customizations

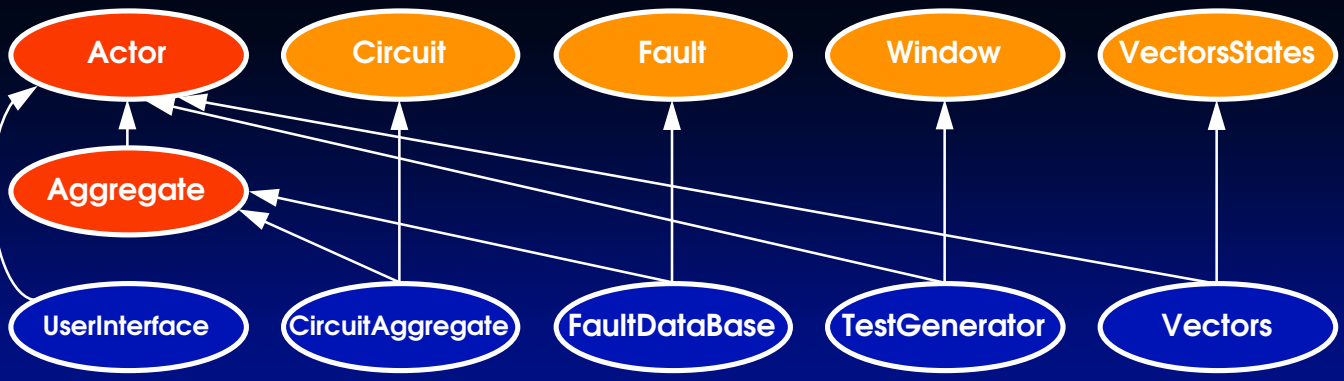
APA Thread Management

Threads
Processes
ProcessGroups

Multiprocessors:	
Mach	1/1/N
Unix	1/N/1
Multicomputers:	
Paragon	N/1/1
Hybrids:	
`Fat' Paragon	N/1/4



Parallel Test Pattern Generation: ProperHITEC



- Serial and parallel applications share a library
- Selection of HITEC/ProperHITEC code via dynamic binding

Parallelization Effort

Classes

HITEC Only	ProperHITEC Only	Shared
0	6	17

Member Functions

	HITEC Member Functions	ProperHITEC Member Functions	Actor Methods
Window/ TestGenerator	78	8	3
Fault/ FaultDatabase	35	21	8

ProperHITEC Results

Circuit	HITEC			ProperHITEC 1				ProperHITEC 2				
	T	E	V	T	SU	E	V	T	SU	PU	E	V
s344	481.4	94.2	89	485.8	0.99	94.2	89	215.8	2.23	1.12	96.8	105
s820	438.3	99.3	959	440.8	0.99	99.3	959	270.4	1.62	0.81	99.2	958
s953	140.2	100	14	147.7	0.95	100	14	89.13	1.57	0.79	100	14
s1238	14.15	100	374	23.29	0.61	100	374	14.69	0.96	0.48	100	383
s1494	819.8	99	1079	821.3	1.00	98.7	1079	503.3	1.63	0.82	99.1	1168

Intel iPSC/860
T: Elapsed time
E: Fault Efficiency
V: # test vectors
SU: Speed up
PU: Processor utilization

ProperHITEC 4					ProperHITEC 8				
T	SU	PU	E	V	T	SU	PU	E	V
194.5	2.48	0.62	96.8	112	142.1	3.39	0.42	96.5	102
158.0	2.77	0.69	99.3	951	108.0	4.06	0.51	98.9	1034
49.00	2.86	0.72	100	24	28.66	4.89	0.61	100	14
12.16	1.16	0.29	100	369	11.12	1.27	0.16	100	402
310.1	2.64	0.66	98.6	1113	192.2	4.27	0.53	98.8	1151

Status

- Platforms
  - Sun MP, Intel iPSCs, Paragon, TMC CM-5, Encore Multimax
  - IP-connected clusters
- Applications
  - Test generation/fault simulation
  - Cell placement
  - Logic synthesis
  - Gauss-Jacobi

Summary

- Class library approach to concurrency
  - Statically type checked
  - Incremental parallelization
  - Parallel/serial co-development
  - Local and global metaprogrammability
  - Independent, parameterizable low-level interface
  - Open implementations
- Future issues
  - Actor-level debugging and visualization
  - Continuation-based I/O