Behavioral Primitives Analysis
An Approach for Selecting Models of Computation

Ivan Jeukens
ijeukens@ime.usp.br

Marius Strum
strum@ime.usp.br

Microelectronics Lab
University of São Paulo

Introduction

- The problem: Choosing Models of Computation
  - Motivation
- Our approach
- Methodology
- Preliminary results
- Ongoing and future work
Motivation

- Previous work at our group
  - ATM Switch
  - E1 Signalization
- ATM
  - simulation: SDF + DDF
  - synthesis: VHDL + CGC
- E1
  - simulation: SDF + DE
  - synthesis: SDF

Motivation (2)

- Both designs contained much control flow
- Heavy use of SDF, why?
  - Some data flow
  - Too many models
  - Designers with little background
- Correct specifications where obtained, but ...
Concept

"Behavioral Primitive" (BP)
- basic behavior employed by a designer when capturing a subset of the system
- analogy to a basis from linear algebra
- BP actually a “pseudo-basis”

What we Need

- Tool 1
  - BP Lib
  - Model
  - Guidelines & Scores

- Tool 2
  - BP Lib
  - Synthesis
  - New Executable Spec
  - Report of Guidelines & Scores

- Manual Extraction
  - Executable Spec
  - Non Executable Spec
Our Work

- Methodology
  - Capture a BP using one or more toy benchmarks under different MoCs
  - Analyze the results of the execution
- Analysis result: a matrix where each position is the rank for the respective pair (BP, MoC)
  - Need for good ranking metrics

Our Work (2)

- We developed a prototype of a tool to aid the analysis of a specification execution
  - Record token exchange and actor’s state
  - Debugging and profiling capabilities
- It can be used for
  - Providing insight into the use of a MoC
  - Evaluating different execution policies of a MoC
  - Finding errors in a specification
  - Optimizing specifications
Analysis Tool

Trace collector frame

Analysis Tool (2)

Topology

Trace table

Receiver state
Analysis Tool (3)

Behavioral Primitives

- Basic Block - butterfly curve
- Conditional Execution - line slope
- Fixed Length Iteration - matrix multiplication
- Synchronism (data) - DCT
- Resource Sharing - dinning philosophers
- Concurrency (data) - trapezoidal rule
- Preemption - ABRO
- Recursion - Fibonacci
- others
**Basic Block**

- Specification
  - domain polymorphic actors
- Execution
  - trivial schedule for all MoCs
  - deadlock situation with CSP

**Conditional Execution**

- Specification
  - SDF: problem with actors always being fired

```java
double coef(int x1, int y1, int x2, int y2) {
    if(x1 == x2) { // Cond1
        return 1.0; // Cte1
    } else
        if(y1 == y2) { // Cond2
            return 0.0; // Cte2
        } else {
            return (double) (y1 - y2)/(x1 - x2);   // Coef
        }
}
```
Conditional Execution (2)

**Specification**
- **DDF**

![](image1)

DDF Specification 1

Conditional Execution (3)

![](image2)

DDF Specification 2
### Conditional Execution (4)

- **Specification**
  - CSP and PN: same as DDF
  - DE

```java
if(x1.hasToken(0)) {
    _x1 = ((IntToken)x1.get(0)).intValue();
}
...
if(trigger.hasToken(0)) {
    trigger.get(0);
    output.broadcast(
        new DoubleToken(_y1 - _y2) / (_x1 - _x2));
}
```

### Conditional Execution (5)

- **Specification**
  - SR: employed a non-monotonic actor

```java
public void fire() throws IllegalActionException {
    if(a.preset(0) && !output.known(0)) {
        output.broadcast(a.get(0));
    }
    if(b.preset(0) && !output.known(0)) {
        output.broadcast(b.get(0));
    }
    if(c.preset(0) && !output.known(0)) {
        output.broadcast(c.get(0));
    }
}
```

- **Execution**
  - DDF: second specification more efficient in terms of firings but had longer running time
Conditional Execution (6)

- Execution
  - DDF

DDF Specification 1

DDF Specification 2

Execution
CSP: again deadlock due to actor D

Fixed Length Iteration

- Specification
  - Polymorphic

```java
for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++) {
        for(int k = 0; k < N; k++) {
            res[i][j] += data[i][k] * cT[k][j];
        }
    }
}
```
Fixed length Iteration (2)

- Specification
  - SDF: non unitary sample rates

```
Sequencer2 → R → CT → Mult → AccumAdd → C → Res
Sequencer1 → R → C → Data
Sequencer3
```

Fixed length Iteration (3)

- Specification
  - PN, CSP and DDF: blocking reads
  - DE: event driven
  - SR: makeAbsent() method

- Execution
  - SDF: FIFO Capacity x Unnecessary token production
  - DDF: FIFO Capacity x Actor Scheduling
Final Remarks

- Our approach: based on the idea of decomposing a specification in a number of primitive behaviors
- Status: we are building the BP libraries
- Tasks: tool, examples with a few TB
- Identification of a set of BP
- Too early to draw relevant conclusions

Future Work

- Near
  - extend the study to other primitives and MoCs
  - give a theoretical basis to the analysis
  - specifications with multiple BP/MoCs
- Far
  - add intelligence to a GUI in order to compose / decompose actors
  - include synthesis metrics
- Hopefully
  - find a strategy to decompose a specification into BP
  - develop a tool that will recognize when to use a MoC