Auckland Reactive Processors and Model Checking Embedded Systems using SystemJ

27 November 2007

Partha S Roop
Research Centre for Embedded Intelligence (CEMBI)
Department of Electrical and Computer Engineering
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>A review of Auckland Reactive Processors</td>
</tr>
<tr>
<td>The SystemJ language</td>
</tr>
<tr>
<td>A protocol stack in SystemJ</td>
</tr>
<tr>
<td>Some tools and implementation pathways</td>
</tr>
<tr>
<td>Model checking: An airport BHS model and its verification</td>
</tr>
<tr>
<td>Comparison to the Esterel Prover</td>
</tr>
<tr>
<td>Conclusions</td>
</tr>
<tr>
<td>Reactive Processors</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>▶ Processors that directly interface with their environment.</td>
</tr>
<tr>
<td>▶ No interrupts, use Esterel-like ISA and preemption.</td>
</tr>
<tr>
<td>▶ Two functions so far ..</td>
</tr>
<tr>
<td>▶ Implementation of embedded systems without using interrupts for environment interaction.</td>
</tr>
<tr>
<td>▶ Direct execution of reactive languages.</td>
</tr>
</tbody>
</table>
REFLIX datapath operation
[FPL02, VLSI 03]
HiDRA: Detailed Architecture [ERSA 04]
Overview of EMPEROR [EMSOFT04, SLAP05, SLAP06]

Inputs from the external environment

- Processor Core 1
  - Program Memory
  - Data Memory
- Processor Core 2
  - Program Memory
  - Data Memory
- Processor Core 3
  - Program Memory
  - Data Memory

Thread Control Unit (TCU)
- Shared Register Bank
  (32 Shared Registers)
- Sync Handling Block (SHB)
- Thread Status Registers (TSR)
- Tick Handling Hardware

Output to the external environment
<table>
<thead>
<tr>
<th>Latest Processor - STARPro</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Multithreaded architecture supporting up to 512 threads.</td>
</tr>
<tr>
<td>- Extends our REMIC processor with an ESU to preserve semantics of Esterel.</td>
</tr>
<tr>
<td>- Executes using the concept of variable tick.</td>
</tr>
<tr>
<td>- Supports both static and dynamic scheduling.</td>
</tr>
<tr>
<td>- With static scheduling in hardware we have about speedup compared to KEP3a while suffering some code size penalty compared to KEP. STARPro utilizes much less hardware resources in comparison.</td>
</tr>
<tr>
<td>- With dynamic scheduling, performance similar to circuit compilers for pure software and even better in STARPro.</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>View of synchrony</td>
</tr>
<tr>
<td>Preemption/Abort</td>
</tr>
<tr>
<td>Scheduling of threads</td>
</tr>
<tr>
<td>Intermediate format</td>
</tr>
<tr>
<td>Pipelining</td>
</tr>
</tbody>
</table>
From SR to GALS

- Esterel like synchronous models are good for hardware that operate using a single clock.
- Many applications such as control systems require both synchronous and asynchronous concurrency.
- We offer a new language and associated tools for GALS systems – a set of synchronous islands, called clock domains that communicate using rendezvous.
- The individual clock domains are Esterel like reactive programs in a Java environment.
- This new language, called SystemJ, compiles to pure Java using two different approaches.
Some hypothesis for our motivation

- An Esterel program is hierarchical – control modelled in Esterel and data computation through C functions called synchronously.
- Consequence – verification of datapath is difficult.
- SystemJ relies on OO encapsulation elegance of Java and by extending Java with reactive constructs, provides a single unifying language.
- Side effect: we may be able to verify control, data and (control + data) seamlessly.
A Protocol Stack in SystemJ
[MEMOCODE’06]
The Model in SystemJ

Listing 4. The top-level SystemJ reaction for the simple protocol stack.

```java
reaction TheStack(in channel reset,
    in channel Byte in_byte)
    { signal Packet packet;
    signal Boolean crc_ok;
    Assemble(reset,in_byte,packet) ||
    Checkcrc(reset,packet,crc_ok) ||
    Prochdr(reset,crc_ok,packet);
}
```

Listing 5. SystemJ reaction generating the test vector.

```java
reaction TestBench(out channel reset,
    out channel Byte out_byte)
    { send reset;
    pause;
    byte tosend[] = {13, 73, 127, 100,
    // ... more values
    55, 77};
    for(int i=0;i<tosend.length;i++)
    send out_byte(new Byte(tosend[i]));
    System.err.println("Test\_completed.");
}
```

Listing 6. A SystemJ GALS, composed of the protocol stack.

```java
system {
    channel reset; channel Byte data;
    TestBench(reset, data) << TheStack(reset, data);
}
```
One Reaction in Detail

Listing 3. SystemJ reaction performing a computation on the packet header.

reaction Prochdr(in channel reset,  
in signal boolean crc_ok,  
in signal Packet inpkt)
{ signal kill_check;  
boolean match_ok;  
while(true)  
  abort(reset) {  
    await inpkt;  
    { abort(kill_check) {  
      // some lengthy computation,  
      // determining the value of match_ok  
    }  
    } || {  
    } || {  
    await crc_ok;  
    if(!crc_ok) emit kill_check;  
  } // parallel  
  if(crc_ok && match_ok)  
    System.err.println("Address_match!");  
  } // abort  
} // reaction Prochdr

Thread performing lengthy address computation

Thread performing Coordination with the CRC checker
Embedded implementations of SystemJ

- SystemJ
  - Hardware compilation
    - AGRC compilation
    - Modified AGRC compilation

- Java code
  - Custom reactive instruction
  - Java data-path
  - ARM, MIPS, NIOS running j2me on OS
  - ARM, MIPS, NIOS running j2me
  - ARM, MIPS, NIOS running j2me
  - Custom built reactive processor

- VHDL
  - FPGA/Soc
Baggage Handling System

- Specialized material handling deployed in Airports.
- Main functions are tracking of bags and sorting of bags.
- **Sortation**: Bags need to be either sent automatically to the destination plane during boarding or to the appropriate arriving baggage carousel upon arrival.
- **Tracking**: All bags must be tracked by the control system at all times using their IDs.
- **Key requirements:**
  - Safety-critical control system.
  - Reliability – system is operational all the time and any disruption causes huge financial loss.
  - Fault tolerance and automated reconfiguration – an active research area now in manufacturing and automation sectors.
Baggage Handling System

Merge Subsystem
Operation of the System

- INCOMING BAGS
- Photoeye 4
- Photoeye 5
- OUTGOING BAGS
- Photoeye 3
- Photoeye 2
- Photoeye 1
Project Motivation

A Typical Problem

INCOMING BAGS

Photoeye 4
Photoeye 5

OUTGOING BAGS

Photoeye 3

Photoeye 1

Photoeye 2
<table>
<thead>
<tr>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Develop a model checker for SystemJ</td>
</tr>
<tr>
<td>- Create a SystemJ model of the merge subsystem of BHS.</td>
</tr>
<tr>
<td>- Validate the model using both simulation and model checking.</td>
</tr>
<tr>
<td>- Compare our approach with other formal provers for embedded systems such as the Esterel prover.</td>
</tr>
</tbody>
</table>
Modelling the Merge in SystemJ

A More Detailed View

- Type S
- Type M
- Photoeye 1
- Photoeye 2
- Photoeye 3
- Photoeye 4
- Photoeye 5
- Recv / NextRecv
- Send / PrevSend
- Gap / Merge
- Bag readings
- Module 1
- Module 2
- INCOMING BAGS
- OUTGOING BAGS
- BAGID
Model Checking using observers

VERIFICATION REPORT

Output
---------------------------------------------
error path
Tick 0: input: -
output: -

Tick 1: input: A
output: -

Tick 2: input: B
output: X

============================
Property Violation : p1Violation
============================

SYSTEM

OBSERVER

Check for Property

Results

Run FSM

Check Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?

A

B

X

Check for Property

Does Property Violate?
Observers

- Writing temporal logic formulae not interesting for engineers.
- Observer is a synchronous parallel thread that is equivalent to an LTL property written in SystemJ syntax.
- Safety: merging conveyor will not send if merge is not present

```java
while(true){
    present(~Merge){
        present(Send)
        emit ob_safe1violated;
    }
    pause;
}
```
Bounded Liveness

//Send should happen within some bounded delay after Merge
while(true){
    present( Merge )
    {
        abort(Send)
        {
            for(int i = 0; i < 50; i++){
                pause;
            }
            emit ob_live2violated;
        }
    }
    pause;
}
A data observer

//Gap is calculated correctly
int GAP = 0;
while(true){
    abort (PE2){
        while(true){
            present(RUN2 & RUN3){
                if(GAP< bound)
                    GAP++;
                if(counter != GAP)
                    emit ob_data1violated;
            }
            pause;
        }
    }
    GAP = 0;
    if(counter != GAP)
        emit ob_data1violated;
}
From JPF to Reactive Prover

- SystemJ programs converted to standard Java using the TReK library or the AGRC intermediate format.
- JPF can verify standard Java. However, we need to verify a reactive program.
- The state of a reactive program is based on tick boundaries which are obtained using pause instructions (state boundaries).
- Reactive prover supplies JPF with this additional state information so that JPF checks for data / control points being observed only at these state boundaries.
Verifier System Architecture

- System Model & Observers
- Input Selection
- Custom Virtual Machine
- Search Strategy
- State Space Reduction
- Reaction Scheduler
- Observer
- Tick Choice Generator
  - Selects inputs from external interface to inject
- Declarations
  - External Valued input bounding

Detect TICK
Verifier System Architecture

**TickChoiceGenerator**
- Watches the program’s executing bytecode and detects:
  - New Signal Object creations
  - Signal Emissions
- Using the collected data
  - Iterates through all Permutations of Input Signals
  - Includes Bounded Valued input signals from Declarations
  - Modifies the Stack to ‘emit’ a signal
  - Logs Inputs for later traces

**Observer**
- Monitors signal emissions
- Wait for a configured violation signal emission
  - Halt execution and print stack trace including I/O
<table>
<thead>
<tr>
<th></th>
<th>List of properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>[P1] Control (bounded liveness): After a conveyor stops it should restart within 100 ticks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[P2] Control (safety): Merge conveyor will not Send if Merge signal is low.</td>
</tr>
<tr>
<td></td>
<td>[P3] Control (bounded liveness): Send becomes high within some bounded delay after Merge.</td>
</tr>
<tr>
<td></td>
<td>[P5] Data and control (safety): Merge will be issued only when there is a sufficient Gap.</td>
</tr>
<tr>
<td></td>
<td>[P6] Data and control (bounded liveness): Data in is equal to data out.</td>
</tr>
</tbody>
</table>
## Verification of a single conveyer

- SystemJ verified P1 to P3 within 20 minutes.
- Esterel verification of the same properties on an equivalent model took less than a minute.
- In both Esterel and SystemJ P2 and P3 failed initially. With additional constraints, they passed.
- A single conveyer itself is pure control and Esterel excels on such systems.
- Complete systems: Esterel hadn’t produced a result for P1 after 17 hours. P1-P4 checked in JPF within 1.2 hours.
- P6 couldn’t be proved in JPF (timed out)
Conclusions

- SystemJ offers both concurrency and reactivity to facilitate the modelling, verification and implementation of a large class of embedded applications.
- Because of pure synchrony within a clock domain, we can model the behaviour of a given embedded device with many concurrent synchronous threads.
- We can model asynchronous components (such as a multiplayer game or a distributed system) using multiple clock domains that synchronize using rendezvous.
- SystemJ has been used in industrial applications (such as airport BHS) and we are now applying SystemJ as the backend code base for an industrial automation standard called IEC61499.
- Esterel is better suited to pure control (reactive) applications while SystemJ is ideal for applications that have both control and data.