An Esterel Virtual Machine

Stephen A. Edwards

Columbia University

Octopi Workshop
Chalmers University of Technology
Gothenburg, Sweden
December 2018
An Esterel Virtual Machine

Goal: Run big Esterel programs in memory-constrained settings.

Our target: the Hitachi H8-based RCX Microcontroller for Lego Mindstorms

[SLAP 2006]
module Example:
input I, S;
output O;
signal R, A in
every S do
    await I;
    weak abort
    sustain R
    when immediate A;
    emit 0
    loop
        pause; pause;
        present R then
            emit A
        end present
    end loop
    end every
end signal
end module
Esterel’s semantics require any implementation to deal with three issues:

- Concurrent execution of sequential threads of control within a cycle
- The scheduling constraints among these threads due to communication dependencies
- How control state is updated between cycles
How did we handle them?

- A virtual machine specifically designed to support Esterel features
- A sequentializing algorithm
- Conversion from GRC to BAL and then to a compact byte code
Phase 1: Schedule
Phase 2: Assign Threads

Assignment Diagram:

1. Assign Threads
2. Process 1
3. Process 2
4. Process 3
5. Process 4
Phase 3: Sequentialize

Sequentialize

Diagram showing sequentialization process with nodes labeled 0 to 4.
Phase 4: Add Labels

![Diagram with nodes and arrows indicating the flow of a program with labels and operations such as "jmp done" and "done"]
Phase 5: Convert to BAL

- t0
  - STHR 1 t1
  - EMT 1
  - SWC 1
  - STHR 1 NR1
  - END

- NR1
  - SWCU
t1
  - TWB 2 2 case_1
  - JMP done

- done
case_1
- jmp done

- done

Diagram:
- Node 0
  - Node 1
  - Node 2
  - Jump 0
    - Node 3
      - Case 1
        - Node 0
          - Node 1
            - Node 2
              - Node 0
                - Node 3
                  - Case 1
                    - Node 0
                      - Node 1
                        - Node 2
                          - Node 0
                            - Node 3
                              - Case 1
                                - Node 0
                                  - Node 1
                                    - Node 2
                                      - Node 0
                                        - Node 3
                                          - Case 1
                                            - Node 0
                                              - Node 1
                                                - Node 2
                                                  - Node 0
                                                    - Node 3
                                                      - Case 1
                                                        - Node 0
                                                          - Node 1
                                                            - Node 2
                                                              - Node 0
                                                                - Node 3
                                                                  - Case 1
                                                                    - Node 0
                                                                      - Node 1
                                                                        - Node 2
                                                                          - Node 0
                                                                            - Node 3
                                                                              - Case 1
                                                                                - Node 0
                                                                                  - Node 1
                                                                                    - Node 2
                                                                                      - Node 0
                                                                                       - Node 3
                                                                                         - Case 1
                                                                                           - Node 0
                                                                                             - Node 1
                                                                                               - Node 2
                                                                                                 - Node 0
                                                                                                  - Node 3
                                                                                                    - Case 1
                                                                                                      - Node 0
                                                                                                        - Node 1
                                                                                                         - Node 2
                                                                                                          - Node 0
                                                                                                            - Node 3
                                                                                                             - Case 1
                                                                                                               - Node 0
                                                                                                                - Node 1
                                                                                                                 - Node 2
                                                                                                                   - Node 0
                                                                                                                     - Node 3
                                                                                                                      - Case 1
                                                                                                                        - Node 0
                                                                                                                            - Node 1
                                                                                                                                - Node 2
                                                                                                                                   - Node 0
                                                                                                                                      - Node 3
                                                                                                                                          - Case 1
                                                                                                                                            - Node 0
                                                                                                                                                - Node 1
                                                                                                                                                    - Node 2
                                                                                                                                                      - Node 0
                                                                                                                                                        - Node 3
                                                                                                                                                            - Case 1
                                                                                                                                                               - Node 0
                                                                                                                - Node 1
                                                                                                                  - Node 2
                                                                                                                    - Node 0
                                                                                                                     - Node 3
                                                                                                                       - Case 1
                                                                                                                         - Node 0
                                                                                                                           - Node 1
                                                                                                                              - Node 2
                                                                                                                                - Node 0
                                                                                                                                   - Node 3
                                                                                                                                     - Case 1
                                                                                                                                         - Node 0
                                                                                                                                             - Node 1
                                                                                                                                                - Node 2
                                                                                                                                                    - Node 0
                                                                                                                                                        - Node 3
                                                                                                                                                            - Case 1
                                                                                                                                                               - Node 0
                                                                                                                - Node 1
                                                                                                                  - Node 2
                                                                                                                    - Node 0
                                                                                                                     - Node 3
                                                                                                                       - Case 1
                                                                                                                         - Node 0
                                                                                                                           - Node 1
                                                                                                                              - Node 2
                                                                                                                                - Node 0
                                                                                                                                   - Node 3
                                                                                                                                     - Case 1
                                                                                                                                         - Node 0
                                                                                                                                             - Node 1
                                                                                                                                                - Node 2
                                                                                                                                                    - Node 0
                                                                                                                                                        - Node 3
                                                                                                                                                            - Case 1
                                                                                                                                                               - Node 0
Phase 6: Convert to Byte Code

```
t0
    STHR 1 t1
    EMT 1
    SWC 1
    STHR 1 NR1
    END

  Convert to byte code
    07 01 00 0e
    04 01
    05 01
    07 01 00 0d
    03

NR1
    SWCU
    0c

t1
    TWB 2 2 case_1
    JMP done
    49 02 00 15
    06 00 15

case_1
done
    SWC 0
    05 00
```
Sequential Code Generation

1. Schedule the nodes in the graph
2. Assign thread numbers
3. Sequentialize the graph
4. Set the execution path by adding labels
5. Convert to BAL
6. Assemble to produce bytecode
Sequentialization
Sequentialization

The dotted line labeled F represents the frontier. The frontier starts at the top of the graph.
The frontier moves down a node at a time in scheduled order.
When a node is in the same thread as the most recently moved one, it is simply moved above the frontier.
Sequentialization

However, when the next node is from a different thread, a switch is added to the previous thread and an active point is added to the new thread just above the just-moved node.
Sequentialization

The algorithm is complete when the frontier has swept across all nodes in scheduled order.
Sequentializing Algorithm

1: for each thread \( t \) in \( G \) do
2:   create new active point \( p \)
3:   copy first node \( n \) of \( t \) in \( G \) to \( n' \) new node in \( G' \)
4:   connect \( p \) and \( n' \)
5:   add \( p \) to \( P[t] \) and add \( n' \) to \( A[t] \)
6:   \( t' = \) the first thread
7: for each node \( n \) in scheduled order do
8:   \( t \) is thread of \( n \)
9:   if \( t \neq t' \) then
10:     for each parent \( p \) in \( P[t'] \) do
11:       for each successor \( c \) of \( p \) in \( A[t'] \) do
12:         create switch node \( s \) from \( t' \) to \( t \) and connect \( s \) between \( p \) and \( c \)
13:     replace \( P[t'] \) with the set of new switch nodes
14:     move \( n \) to \( P[t] \) and remove it from \( A[t] \)
15: for each unreached successor \( c \) of \( n \) do
16:   copy \( c \) to \( c' \) new node in \( G' \)
17:   if \( n \) is a fork then
18:     add child to \( A[\text{thread of } c] \)
19:   else
20:     add child to \( A[t] \)
21:   \( t' = t \) \{remember the last thread\}
Why VM?

- Goal: constrained-memory environment
- Instruction set has direct support for Esterel constructs like concurrency, preemption, and signals
- E.g., a context switch can be specified in just two bytes
VM Details

- Signal status registers
- Completion code registers
- Per-thread program counters
- Inter-instant state-holding registers
## VM: Signal, State, and Thread

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMT</td>
<td>Emit a Signal</td>
<td>04 RR</td>
</tr>
<tr>
<td>SSIG</td>
<td>Clear Signal</td>
<td>0A RR</td>
</tr>
<tr>
<td>SSTT</td>
<td>Set State</td>
<td>0B RR VV</td>
</tr>
<tr>
<td>STHR</td>
<td>Set Thread</td>
<td>07 TT HH LL</td>
</tr>
</tbody>
</table>
## VM: Control Flow Instructions

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>Tick End</td>
<td>03</td>
</tr>
<tr>
<td>JMP</td>
<td>Jump</td>
<td>06 HH LL</td>
</tr>
<tr>
<td>NOP</td>
<td>No Operation</td>
<td>01</td>
</tr>
</tbody>
</table>
## VM: Branch, Switch, Terminate

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWB</td>
<td>Multiway Branch (State)</td>
<td>2D NL RR HH2 LL2 ...</td>
</tr>
<tr>
<td></td>
<td>Multiway Branch (Comp.)</td>
<td>4D NL RR HH2 LL2 ...</td>
</tr>
<tr>
<td></td>
<td>Two Way Branch (State)</td>
<td>29 RR HH LL</td>
</tr>
<tr>
<td>TWB</td>
<td>Two Way Branch (Signal)</td>
<td>49 RR HH LL</td>
</tr>
<tr>
<td></td>
<td>Two Way Branch (Comp.)</td>
<td>69 RR HH LL</td>
</tr>
<tr>
<td>SWC</td>
<td>Switch Thread</td>
<td>05 TT</td>
</tr>
<tr>
<td>SWCU</td>
<td>Switch Unknown</td>
<td>0C</td>
</tr>
<tr>
<td>TRM</td>
<td>Set Completion Code for Join</td>
<td>08 RR VV</td>
</tr>
</tbody>
</table>
switch(opcode & 0x1F){
  ...
  case SWC:
    // Increment the program counter
    ++pc;
    // Store the current thread as the last thread
    last_thread = current_thread;
    // Get the next thread
    current_thread = *pc;
    // Increment the program counter
    ++pc;
    // Store old pc associated with the old thread
    threads[last_thread] = pc;
    // Load the pc associated with the new thread
    pc = threads[current_thread];
    break;
    ...
  ...
}
case SWCU:
    // Make the thread stored in last_thread, the current thread
    temp = current_thread;
    current_thread = last_thread;
    last_thread = temp;
    // Store old pc
    threads[last_thread] = pc;
    // Load new pc
    pc = threads[current_thread];
    break;

...
VM in action
VM in action

```
t0
00:  STHR 1 t1
04:  EMT 1
06:  SWC 1
08:  STHR 1 NR1
12:  SWC 1
14:  END
    NR1
15:  SWCU

          pc = 0
last_thread = 0

Threads  Signals
0 0
0 0

States  Joins
.. ..
```

```
t1
16:  TWB 2 1 case_1
19:  JMP done
    case_1
done
22:  SWC 0
```
VM in action

t0
00: STHR 1 t1
04: EMT 1
06: SWC 1
08: STHR 1 NR1
12: SWC 1
14: END

last_thread = 0

pc = 4

Threads    Signals
0          0
0          0
16         0

States    Joins
..          ..

15: SWCU
t1
16: TWB 2 1 case_1
19: JMP done
    case_1
done
22: SWC 0
VM in action

t0
00:  STHR 1 t1
04:  EMT 1
06:  SWC 1
08:  STHR 1 NR1
12:  SWC 1
14:  END
    NR1
15:  SWCU

pc = 6
last_thread = 0

Threads  | Signals
---------|--------
    0   |   0    
    16  |   1    

States  | Joins
--------|--------
    ..   |    ..  

16:  TWB 2 1 case_1
19:  JMP done
case_1
done
22:  SWC 0
VM in action

t0
00: STHR 1 t1
04: EMT 1
06: SWC 1
08: STHR 1 NR1
12: SWC 1
14: END
NR1
15: SWCU
t1
16: TWB 2 1 case_1
19: JMP done
   case_1 done
22: SWC 0

pc = 16
last_thread = 0

<table>
<thead>
<tr>
<th>Threads</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>
VM in action

```
  pc = 19
  last_thread = 0

+----------------+----------------+   Threads   |   Signals   |
|                |                |            |             |
|                |                |   8        |    0        |
|                |                |   16       |    1        |
```

States | Joins
```
.. | ..
```

t0
00:   STHR 1 t1
04:   EMT 1
06:   SWC 1
08:   STHR 1 NR1
12:   SWC 1
14:   END
   NR1
15:   SWCU

t1
16:   TWB 2 1 case_1
19:   JMP done
       case_1
done
22:   SWC 0
VM in action

t0
00:  STHR 1 t1
04:  EMT 1
06:  SWC 1
08:  STHR 1 NR1
12:  SWC 1
14:  END
    NR1
15:  SWCU
    t1
16:  TWB 2 1 case_1
19:  JMP done
    case_1
done
22:  SWC 0

pc = 22
last_thread = 0

<table>
<thead>
<tr>
<th>Threads</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>
VM in action

t0
00: STHR 1 t1
04: EMT 1
06: SWC 1
08: STHR 1 NR1
12: SWC 1
14: END

NR1
15: SWCU

t1
16: TWB 2 1 case_1
19: JMP done

States

pc = 8

last_thread = 1

Signals

Threads

8

24

States

Joins

..
VM in action

t0
00:   STHR 1 t1
04:   EMT 1
06:   SWC 1
08:   **STHR 1 NR1**
12:   SWC 1
14:   END
   NR1
15:   SWCU
   t1
16:   TWB 2 1 case_1
19:   JMP done
   case_1
done
22:   SWC 0

threads   signals
\[\begin{array}{cc}
8 & 0 \\
15 & 1 \\
\end{array}\]

states   joins
\[\begin{array}{cc}
. . \\
. . \\
\end{array}\]

pc = 12
last_thread = 1
VM in action

t0
00: STHR 1 t1
04: EMT 1
06: SWC 1
08: STHR 1 NR1
12: SWC 1
14: END

NR1
15: SWCU
t1
16: TWB 2 1 case_1
19: JMP done
case_1
done
22: SWC 0

pc = 15
last_thread = 0

<table>
<thead>
<tr>
<th>Threads</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>
VM in action

<table>
<thead>
<tr>
<th>t0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00:</td>
<td>STHR 1 t1</td>
</tr>
<tr>
<td>04:</td>
<td>EMT 1</td>
</tr>
<tr>
<td>06:</td>
<td>SWC 1</td>
</tr>
<tr>
<td>08:</td>
<td>STHR 1 NR1</td>
</tr>
<tr>
<td>12:</td>
<td>SWC 1</td>
</tr>
<tr>
<td>14:</td>
<td>END</td>
</tr>
<tr>
<td></td>
<td>NR1</td>
</tr>
<tr>
<td>15:</td>
<td>SWCU</td>
</tr>
<tr>
<td>t1</td>
<td></td>
</tr>
<tr>
<td>16:</td>
<td>TWB 2 1 case_1</td>
</tr>
<tr>
<td>19:</td>
<td>JMP done</td>
</tr>
<tr>
<td></td>
<td>case_1</td>
</tr>
<tr>
<td></td>
<td>done</td>
</tr>
<tr>
<td>22:</td>
<td>SWC 0</td>
</tr>
</tbody>
</table>
VM in action

```
t0
00:   STHR 1 t1
04:   EMT 1
06:   SWC 1
08:   STHR 1 NR1
12:   SWC 1
14:   END
   NR1
15:   SWCU
   t1
16:   TWB 2 1 case_1
19:   JMP done
   case_1
   done
22:   SWC 0
```

```
pc = 15
last_thread = 0

<table>
<thead>
<tr>
<th>Threads</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
```
The engineering details

- brickOS 2.6.10 on Redhat Linux
- gcc cross compiler 4.0.2. for H8300
- Download lx files to the lego RCX via USB IR tower
## Code Sizes

<table>
<thead>
<tr>
<th>Example</th>
<th>BAL</th>
<th>x86</th>
<th>BAL%</th>
<th>x86%</th>
</tr>
</thead>
<tbody>
<tr>
<td>dacexample</td>
<td>369</td>
<td>917</td>
<td>60%</td>
<td>842</td>
</tr>
<tr>
<td>abcd</td>
<td>870</td>
<td>2988</td>
<td>71%</td>
<td>2648</td>
</tr>
<tr>
<td>greycounter</td>
<td>1289</td>
<td>3571</td>
<td>64%</td>
<td>2836</td>
</tr>
<tr>
<td>tcint</td>
<td>5667</td>
<td>11486</td>
<td>51%</td>
<td>10074</td>
</tr>
<tr>
<td>atds-100</td>
<td>10481</td>
<td>38165</td>
<td>73%</td>
<td>26334</td>
</tr>
</tbody>
</table>

BAL: the size of our bytecode (in bytes)
x86: the size of optimized C code for an x86
H8: the size of optimized C code for an Hitachi H8
Percentages represent the size savings of using bytecode.
## Execution Times

<table>
<thead>
<tr>
<th>Example</th>
<th>x86</th>
<th>BAL</th>
<th>18×</th>
<th>4×</th>
<th>7×</th>
</tr>
</thead>
<tbody>
<tr>
<td>dacexample</td>
<td>0.06μs</td>
<td>1.1μs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tcint</td>
<td>0.28μs</td>
<td>1.1μs</td>
<td></td>
<td></td>
<td>4×</td>
</tr>
<tr>
<td>atds-100</td>
<td>0.20μs</td>
<td>1.4μs</td>
<td></td>
<td></td>
<td>7×</td>
</tr>
</tbody>
</table>
Conclusions

- Simple Virtual Machine
- Compilation scheme statically schedules the concurrent behavior and generates straight-line code for each thread
- VM supports context-switching well
- Bytecode for our virtual machine is roughly half the size of optimized native assembly code generated from C
- Speed tradeoff not that bad! Between 4 and 7 times slower than optimized C code