System-on-a-chip and the Coming Design Revolution

Stephen A. Edwards
Department of Computer Science, Columbia University
www.cs.columbia.edu/~sedwards
sedwards@cs.columbia.edu

1918 Sears Roebuck Catalog

Motors became cheap enough to embed in any appliance that needed them.

How many motors do you own?


2000 MacMall Catalog

How many computers do you own?

What will the SoCs of the future be?

Hint:

Transistor Cost Continues Plummeting

Each Pentium sold for about $600 initially.
Source: Intel

Computers’ Changing Role

Environment and humans subservient to computer
Simple peripherals

Computers subservient to humans and the environment
Complex peripherals

Embedded System Challenges

Real-time Deadlines
Embedded System Challenges

Complexity

Software complexity growing

Size of Typical Embedded System

<table>
<thead>
<tr>
<th>Year</th>
<th>LOC</th>
<th>Growth Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>13 kLOC</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>21 kLOC</td>
<td>44% per year</td>
</tr>
<tr>
<td>1998</td>
<td>1 MLOC</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2 MLOC</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>16 MLOC</td>
<td>≈ Windows NT 4.0</td>
</tr>
<tr>
<td>2010</td>
<td>32 MLOC</td>
<td>≈ Windows 2000</td>
</tr>
</tbody>
</table>


Written in stone-age languages

“Which of the following programming languages have you used for embedded systems in the last 12 months?”

- C 81%
- Assembly 70%
- C++ 39%
- Visual Basic 16%
- Java 7%


Embedded System Challenges

Concurrency

Existing Techniques

...aren’t up to the task.

- Existing multi-threaded concurrency models
  ...are completely unstructured
  The “goto” of control
- Most real-time scheduling
  ...ignores communication aspects

We need some alternatives!

Timing

Java

```java
class PClock implements Runnable {
    public void run() {
        for (;;) {
            java.util.Date now =
                new java.util.Date();
            System.out.
                println(now.toString());
            try {
                Thread.currentThread().
                    sleep(1000);
            } catch (IntExcept e) {}  
        }
    }
}
```

```java
public class Clock {
    public static void
        main(String args[]) {
        Thread t =
            new Thread(new PClock());
        t.start();
    }
}
```

Java class PClock

- Implement Runnable
- For loop
- Print current date
- Sleep for 1000 ms

An Example

```java
public class PClock {
    public void run() {
        for (;;) {
            java.util.Date now =
                new java.util.Date();
            System.out.
                println(now.toString());
            try {
                Thread.currentThread().
                    sleep(1000);
            } catch (IntExcept e) {}  
        }
    }
}
```

Esterel

```esterel
every 1000 MS do
    emit SECOND
end

Just works
```

An Alternative: Esterel

Domain-specific language for safety-critical, real-time systems.

Uses a synchronous model of time that is deterministic and provides precise control over time.

Timing verification becomes checking a single worst-case-execution-time bound.

An Example

```esterel
await A;
emit B;
present C then
    emit D end;

Force signal present in this cycle
```

An Example

```esterel
await A;
emit B;
present C then
    emit D end;
pause
```

A Leap Second?
An Example

Infinite Loop

loop
  await A;
  emit B;
  present C then
    emit D end;
  pause
end

Run Concurrently

loop
  present B then
    emit C end;
  pause
end

Same-cycle bidirectional communication

every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
end

Translate every

Good for hierarchical FSMs
Bad at manipulating data
Hardware Esterel variant proposed to address this

Add Threads

Split at Pauses

Add Code Between Pauses
Translate Second Thread

```
every R do
  loop
    await A;
    emit B;
  present C then
    emit D end;
  pause
end
```

Finished Translating

```
every R do
  loop
    await A;
    emit B;
  present C then
    emit D end;
  pause
end
```

Add Dependencies and Schedule

```
every R do
  loop
    await A;
    emit B;
  present C then
    emit D end;
  pause
end
```

Run First Node

```
R

1  s  2
A
B
C
D
s=2 s=1
```

Run First Part of Left Thread

```
R

1  s  2
A
B
C
D
s=2 s=1
```

Context Switch

```
1  s  2
A
B
C
D
s=2 s=1
```

Run Right Thread

```
R

1  s  2
A
B
C
D
s=2 s=1
```

Context Switch

```
1  s  2
A
B
C
D
s=2 s=1
```

Finish Left Thread

```
R

1  s  2
A
B
C
D
s=2 s=1
```
Plummeting transistor cost is making it practical to put more, smaller computer systems everywhere. Implemented with SoC technology, these embedded systems will be dominated by software.

Embedded system challenges:

- Real-time issues
- Concurrency
- Software complexity and reliability

Summary

Esterel and the synchronous paradigm solve some problems

- Synchronous model provides deterministic concurrency
- Finite state permits automatic model checking
- Execution time verification provides timing assurance
- Efficient compilation scheme eliminates OS overhead