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

Home Motor.

This motor, as shown above, will operate a sewing machine. Easily attached; makes sewing a pleasure. The many attachments shown on this page may be operated by this motor and help to lighten the burden of the home. Operates on usual city current of 105 to 115 volts. Shipping weight, about 5 pounds.

No. 5777564 Price, complete, as shown.................. $8.75

Easter Attachment.

Wipe eggs and house rugs and many other things will be found for these attachments which are used in connection with the Home Motor. Parts include the stand, handle and the basket. Shipping weight, about 14 ounces.

No. 5777565 Price.................. $1.30

Churn and Mixer Attachment.

Used in connection with the Home Motor, makes a small churn and mixer for which you will find many uses. The attachments include the base, support, mixer, handle and sheet cover for jar. Shipping weight, about 14 pounds.

No. 5777582 Price.................. $1.30

Fan Attachment.

Includes fan and guard which can be quickly attached to Home Motor, and will be a great comfort in hot weather. Shipping weight, about 14 ounces.

No. 5776515 Price............... $1.50

About $100 in today’s dollars.

What happened to Home Motors?

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

How many motors do you own?
How many computers do you own?
What will the SoCs of the future be?

Hint:
Hidden Computers

- Casio Camera Watch
- Nokia 7110 Browser Phone
- Sony Playstation 2
- Philips DVD Player
- Philips TiVo Recorder
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>

“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

Photo by Thomas Danoghue
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!
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.
Timing

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) {} 
        }
    }
}

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

$ java Clock
Sat Sep 14 13:04:27 EDT 2002
Sat Sep 14 13:04:29 EDT 2002
Sat Sep 14 13:04:30 EDT 2002
Sat Sep 14 13:04:31 EDT 2002

Esterel

every 1000 MS do
emit SECOND
end

Just works

A Leap Second?
An Example

emit B;
present C then emit D end;

Force signal present in this cycle
Make D present if C is
An Example

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

Wait for next cycle where A is present
Wait for next cycle
An Example

loop

Infinite Loop

await A;
emit B;
present C then
emit D end;
pause
end
An Example

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

Run Concurrently

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

end

loop
  present B then
  emit C end;
  pause
end

end
An Example

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

Same-cycle bidirectional communication
```

```plaintext
loop
  present B then
    emit C end;
  pause
end
end
```
An Example

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

Good for hierarchical FSMs
Bad at manipulating data

Hardware Esterel variant proposed to address this
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
end

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

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

loop
  present B then
    emit C end;
  pause
end

s=2
s=1
every R do
  loop
    await A;
    emit B;
    present C then
      emit D end;
    pause
  end
end

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

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

loop
  present B then
  emit C end;
  pause
end
Run First Node
Run First Part of Left Thread

R

1

s=2

A

B

C

D

s=1

R

1

S

A

B

2

C
Context Switch

1. **R**
2. **A**
3. **B**
4. **C**
5. **D**

- **s=2**
- **s=1**

- **t=0**
- **t=1**

1. **R**
2. **S**
3. **A**
4. **B**
5. **C**

- **1**
- **2**
Run Right Thread
Context Switch
Finish Left Thread
Completed Example
if (!R) {
    if (s == 1 && A) {
        B = 1;
        t = 0;
    } else {
        t = 1;
    }
    if (B) C = 1;
    if (t == 0) {
        if (C) D = 1;
        s = 2;
    } else {
        s = 1
    }
}
Summary

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
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