We intend to make Esterel a viable hardware description language for control-dominated systems by developing a compiler that produces optimized circuits from it.

### A State Assignment Example

```est
abort
[    
  await A; await B ||
  await C
]
when D;
emit E;
pause;
[
  await F ||
  await G
]
```

### Basic Circuit Generation

```verilog
module A;
    input [1:0] A;
    output [1:0] B;
    wire C;
    assign C = A;
endmodule
```

### Why Is Esterel More Succinct?

Esterel's semantics match hardware. Translation is straightforward.

Nice feature: state space is well-defined and hierarchical (e.g., due to `abort` and concurrency).

Enables a hierarchical state assignment/synthesis procedure.

### Basic Circuit Generation

```verilog
module A;
    input [1:0] A;
    output [1:0] B;
    wire C;
    assign C = A;
endmodule
```

### Berry's Technique

Berry's technique [1992] works, but is fairly inefficient:

- Many combinational redundancies. E.g., present A then emit B end; present C then emit D end produces two redundant OR gates.
- Many sequential redundancies. One flop per pause can be very wasteful.

### Generating Fast Circuits

Esterel's semantics match hardware. Translation is straightforward.

Nice feature: state space is well-defined and hierarchical (e.g., due to `abort` and concurrency).

Enables a hierarchical state assignment/synthesis procedure.
This generated many sequential don’t-cares that were slowing the logic.

Sequential Redundancy

Signal emitted in a cycle where it is never needed

% during first cycle: % start sustaining pREQ; on the next cycle, we % shall have pREQ and DMA address ready cycle a % prepare lots drive for next cycle
emit pLocalDrives;
await tick;
% setup data path from pam to host emit pPamDrives;
% ... emit pHostDrives;

What does it take to select a good encoding?

Compared expensive automatic flow

V5 → SIS with sequential optimization

to human cleverness

CEC → manual encoding → SIS (combined)

Many local optimizations possible

Matching SIS required knowing some state reachability

Really need a combination of both for quality results
**Overall Algorithm**
- Registers become nodes with \(-p\) delay (negative clock period)
- Compute “complex” arrival times for each node.
- Bellman-Ford relaxation algorithm on cyclic graph.
- Number of variants pruned aggressively.
- Reconstruct the circuit: choose varia

**More aggressive decomposition**

**Shannon Decomposition** for Retiming

**Delay/area tradeoff: 128-bit**

**“Tech mapping” Shannon**

**Shannon Decomposition**

**ISCAS benchmark results**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Retimed area</th>
<th>Sh. + ref. period area</th>
</tr>
</thead>
<tbody>
<tr>
<td>s510</td>
<td>8 184</td>
<td>8 203</td>
</tr>
<tr>
<td>s641</td>
<td>11 115</td>
<td>10 147</td>
</tr>
<tr>
<td>s713</td>
<td>11 118</td>
<td>10 150</td>
</tr>
<tr>
<td>s820</td>
<td>7 206</td>
<td>7 258</td>
</tr>
<tr>
<td>s832</td>
<td>7 217</td>
<td>6 235</td>
</tr>
<tr>
<td>s838</td>
<td>10 154</td>
<td>11 235</td>
</tr>
<tr>
<td>s1196</td>
<td>9 265</td>
<td>9 443</td>
</tr>
<tr>
<td>s1423</td>
<td>24 498</td>
<td>19 559</td>
</tr>
<tr>
<td>s1488</td>
<td>6 453</td>
<td>6 485</td>
</tr>
<tr>
<td>s1494</td>
<td>6 456</td>
<td>6 488</td>
</tr>
<tr>
<td>s9234</td>
<td>11 662</td>
<td>9 851</td>
</tr>
</tbody>
</table>

**“Tech mapping” Shannon**

**After Retiming**
### Deliverables

**The Columbia Esterel Compiler**

http://www1.cs.columbia.edu/~sedwards/cec/

V5-compliant open-source Esterel compiler

Backends for C, Verilog, BLIF, and VHDL

Written in C++

Source and Linux binaries available

### Last Year’s Accomplishments

- LCTES paper on software backend
- IWLS paper on state-encoding experiments (submitted)
- IWLS paper on Shannon for Retiming (submitted)
- SLAP paper on SHIM language for hardware/software codesign
- IWLS paper on hardware synthesis for devices
- LCTES paper on language for device drivers

### Next Year’s Goals

- Shannon/Retiming flow on higher-level models
- Improved Shannon area synthesis
- Peephole state optimization algorithm
- Global, approximate reachability algorithm

### Publications 1

Stephen A. Edwards.

SHIM: A Language for Hardware/Software Integration.


Stephen A. Edwards.

The challenges of hardware synthesis from C-like languages.

In *Proceedings of Design Automation and Test in Europe (DATE)*, Munich, Germany, March 2005.


Generating Fast Code from Concurrent Program Dependence Graphs.


### Publications 2


Compiling Esterel into Static Discrete-Event Code.


Stephen A. Edwards.

Making Cyclic Circuits Acyclic.

In *Proceedings of the 40th Design Automation Conference (DATE)*, Anaheim, California, June 2-6, 2003. pp. 159-162.

Stephen A. Edwards.

Compiling Concurrent Languages for Sequential Processors.