Generating Fast Circuits

Esterel's semantics match hardware. Tra
is straightforward.

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

Enables a hierarchical state
assignment/synthesis procedure.

An Overview of Esterel

Synchronous model of time: implicit glob

Communication through wire-like signals

Two flavors of statement:

Combinational	**Sequential**
Execute in one cycle	*Take multiple cyc*
emit	pause
present / if	await
loop	sustain

High Level Synthesis from Synchronous Language Es

Raising the level of abstraction above RT

Prof. Stephen A. Edwards

Students: Cristian Soviani, Jia Zeng (20

Mike Kishinevsky, Intel

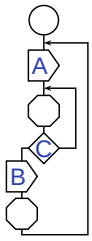
*We intend to make Esterel a viable hardw
description language for control-dominat
systems by developing a compiler that p
optimized circuits from it.*

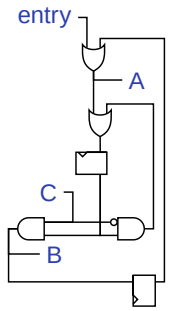
A State Assignment Examp

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


Basic Circuit Generation

```
loop
  emit A; await C;
  emit B; pause
end
```





Verilog More Verbose Than

```
case (cur_state) // synopsys parallel_case
  IDLE:    begin
     if (pcsu_powerdown & !jmp_e &
         !valid_diag_window) begin
    next_state = STANDBY_PWR_DN;
    end
    else if (valid_diag_window | ibuf_full |
             jmp_e) begin
    next_state = NC_REQ_STATE;
    end
    else if(icu_miss&!cacheable) begin
    next_state = NC_REQ_STATE ;
    end
    else if (icu_miss&cacheable) begin
    next_state = REQ_STATE;
    end
    else    next_state = cur_state ;
end

NC_REQ_STATE: begin
    if(normal_ack| error_ack) begin
    next_state = IDLE ;
    end
    else    next_state = cur_state ;
end

REQ_STATE:  begin
    if (normal_ack) begin
    next_state = FILL_2ND_WD;
    end
    else if (error_ack) begin
    next_state = IDLE ;
    end
    else next_state = cur_state ;
end

FILL_2ND_WD: begin
    if(normal_ack) begin
    next_state = REQ_STATE2;
    end
    else if (error_ack) begin
    next_state = IDLE ;
    end
    else next_state = cur_state ;
end

REQ_STATE2:  begin
    if(normal_ack) begin
    next_state = FILL_4TH_WD;
    end
    else if (error_ack) begin
    next_state = IDLE ;
    end
    else next_state = cur_state ;
end

FILL_4TH_WD: begin
    if(normal_ack| error_ack) begin
    next_state = IDLE;
    end
    else next_state = cur_state ;
end

STANDBY_PWR_DN: begin
    if(pcsu_powerdown | jmp_e ) begin
    next_state = IDLE;
    end
    else next_state = STANDBY_PWR_DN;
end

default:     next_state = 7'bx;
endcase
```

```
loop
  await
    case [icu_miss and
          not cacheable] do
      await [normal_ack or er
    end
    case [icu_miss and
          cacheable] do
      abort
        await 4 normal_ack;
      when error_ack
    end
    case [pcsu_powerdown and
          not jmp_e and
          not valid_diag_win
      await [pcsu_powerdown a
             not jmp_e]
    end
  end;
  pause
end
```


Hierarchical States

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


Basic Circuit Generation

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

- Many combinational redundancies. E
  `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 wast
  Touati, Toma, Sentovich, and Berry
  [1993–1997] proposed techniques to
  eliminate many, but requires reachab
  space and only works on circuit.

Why is Esterel More Succin

Verilog:

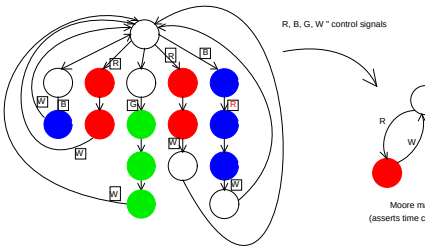
```
REQ_STATE2:  begin
   if(normal_ack) begin
     next_state = FILL_4TH_WD;
   end
   else if (error_ack) begin
     next_state = IDLE ;
   end
   else next_state = cur_state;
end
```

Este

```
abor
  aw
when
```

- Esterel provides cross-clock control-
- State machine logic represented imp
- Higher-level constructs like *await*

Machines running in lock-s



This generated many sequential don't-ca
were slowing the logic.

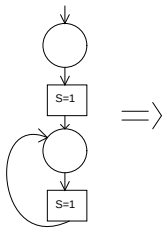
Local Redundancy

Locally redundant constructs:

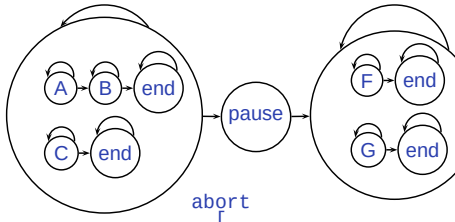
```
pause;
sustain S
```

equivalent to

```
loop
  pause;
  emit S
end loop
```



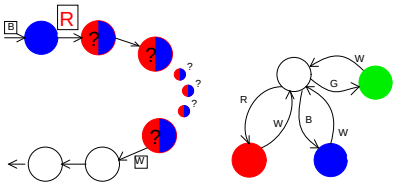
Five Simple FSMs




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


Sequential Redundancy

Signal emitted in a cycle where it is neve

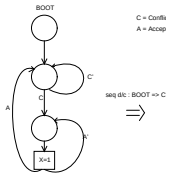



```
% during first cycle :
% * start sustaining pWREQ: on the next cycle, we
%   shall have WREQ and DMA address ready cycle a
% * prepare Lca drive for next cycle

emit pLcaDrives;
await tick;
% setup data path from pam to host
emit pPamDrives;
% ...
emit pHostDrives;
```


Simple Sequential Don't-Ca

```
loop
  await ConflictOnSEL;
  do
    every immediate SEL do
      emit RejectSEL
    end
  watching AcceptSEL
end loop
```



Needed to know that ConflictOnSEL was present in the first cycle.

Encoding Experiment

What does it take to select a good encod

Compared expensive automatic flow

V5 → SIS with sequential optimiza

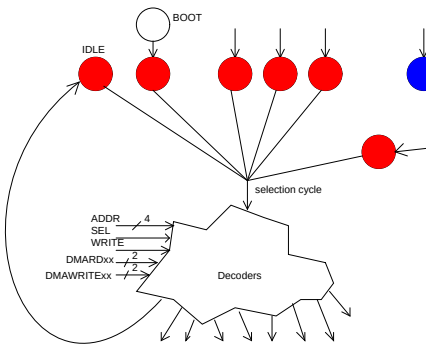
to human cleverness

CEC → manual encoding → SIS (combi

Results

example	size	method	levels	LUTs	I
graycounter	91	V5 + blifopt	5	66	
		manual	4	51	
abcdef	142	V5 + blifopt	5	114	
		manual	3	128	
mem-ctrl	80	V5 + blifopt	3	24	
		CEC + comb	3	52	
		CEC + blifopt	3	27	
		manual	2	31	
		Original VHDL	2	17	
mem-ctrl2	36	V5 + blifopt	2	17	
		CEC + comb	2	23	
		CEC + blifopt	2	18	
		manual	2	14	
		JEDI + comb	2	14	
tcint	689	V5 + blifopt	5	93	
		manual	3	118	

Equivalent States, Redunda



Discoveries

Many local optimizations possible

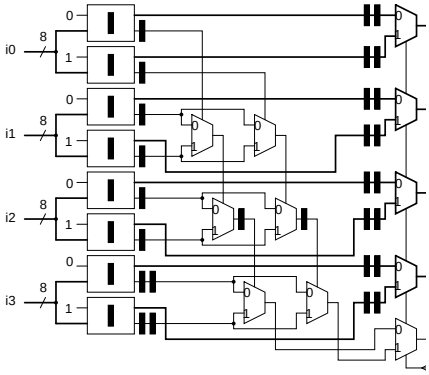
Matching SIS required knowing some sta
reachability

Really need a combination of both for qu
results

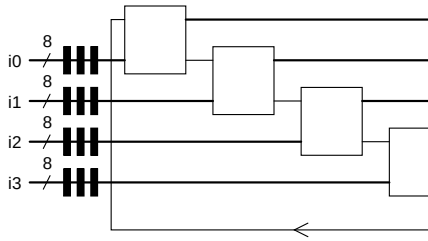
Overall Algorithm

- Registers become nodes with $-p$ de (negative clock period)
- Compute "complex" arrival times for at each node.
  - Bellman-Ford relaxation algorithm cyclic graph.
  - Number of variants pruned aggres
- Reconstruct the circuit: choose varia each node that satisfies these arrival
- Run normal retiming.

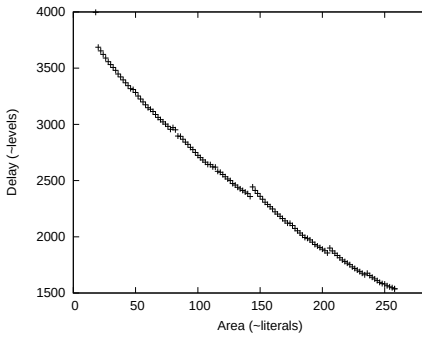
More aggressive decompos



Shannon Decomp. for Reti

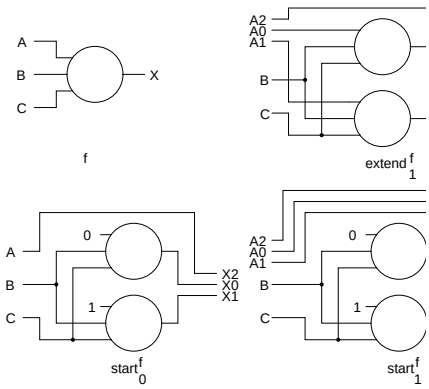


Delay/area tradeoff: 128-bit

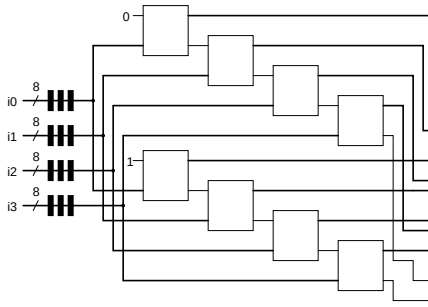


Fast: 24s to compute 120 points (88s inc

"Tech mapping" Shannon



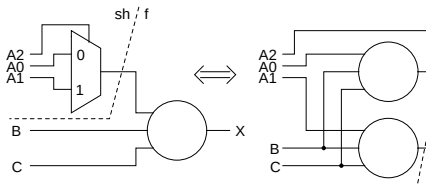
Shannon Decomposition



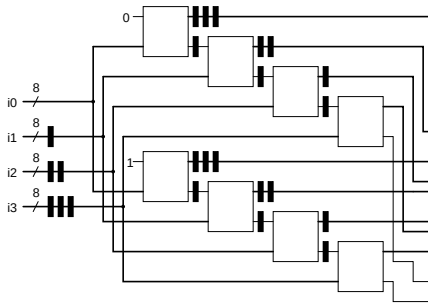
ISCAS benchmark results

	reference		retimed		Sh. + ret.	
	period	area	period	area	period	area
s510	8	184	8	203	8	203
s641	11	115	10	147	8	210
s713	11	118	10	150	9	212
s820	7	206	7	258	7	258
s832	7	217	6	235	6	234
s838	10	154	11	235	8	373
s1196	9	265	9	443	9	444
s1423	24	498	19	559	13	846
s1488	6	453	6	485	6	484
s1494	6	456	6	488	6	487
s9234	11	662	9	851	7	1037

"Tech mapping" Shannon



After Retiming



Publications 1

Stephen A. Edwards.
SHIM: A Language for Hardware/Software Integratio
In *Proceedings of Synchronous Languages, Applica*
Programming (SLAP), Edinburgh, Scotland, April 20

Stephen A. Edwards.
The challenges of hardware synthesis from C-like la
In *Proceedings of Design Automation and Test in Eu*
(DATE), Munich, Germany, March 2005.

Jia Zeng, Cristian Soviani, and Stephen A. Edwards.
Generating Fast Code from Concurrent Program De
Graphs.
In *Proceedings of the ACM SIGPLAN/SIGBED Conf*
Languages, Compilers, and Tools for Embedded Sy
(LCTES), Washington, DC, June 2004.

Deliverables

The Columbia Esterel Compiler

`http://www1.cs.columbia.edu/~s`

V5-compliant open-source Esterel comp

Backends for C, Verilog, BLIF, and VHDL

Written in C++

Source and Linux binaries available

Publications 2

Stephen A. Edwards, Vimal Kapadia, and Michael H
Compiling Esterel into Static Discrete-Event Code.
In *Proceedings of Synchronous Languages, Applica*
Programming (SLAP 2004). Barcelona, Spain, Marc

Stephen A. Edwards.
Making Cyclic Circuits Acyclic.
In *Proceedings of the 40th Design Automation Conf*
2003). Anaheim, California, June 2-6, 2003. pp. 159

Stephen A. Edwards.
Compiling Concurrent Languages for Sequential Pro
ACM Transactions on Design Automation of Electron
(TODAES) 8(2):141-187, April 2003.

Last Year's Accomplishmen

- LCTES paper on software backend
- IWLS paper on state-encoding exper (submitted)
- IWLS paper on Shannon for Retimin (submitted)
- SLAP paper on SHIM language for hardware/software codesign
- IWLS paper on hardware synthesis f
- LCTES paper on language for device

Next Year's Goals

- Shannon/Retiming flow on higher-lev models
- Improved Shannon area synthesis
- Peephole state optimization algorithr
- Global, approximate reachability algo