High Level Synthesis from the Synchronous Language Esterel 1068.00

Raising the level of abstraction above RTL

Prof. Stephen A. Edwards

Students: Cristian Soviani, Jia Zeng (2007?)

Mike Kishinevsky, Intel

Results (Papers)

- Shannon decomposition plus retiming
  Soviani, Tardieu, & Edwards, DATE 2006
- High-level synthesis for router pipelines
  Soviani, Hadžić, & Edwards, DAC 2006 (submitted)
- More efficient “decyclification” algorithm
  Neiroukh, Edwards, & Song, ISVLSI 2006
- Separate compilation for Esterel (software)
  Zeng & Edwards, ICESS 2005
- Approximate Esterel reachability (formal)
  Tardieu & Edwards, ATVA 2005
- An Esterel virtual machine for small memories
  Plummer, Khajanchi, & Edwards, SLAP 2006

Results (Software)

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

Verilog More Verbose Than Esterel

```
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 = cur_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 error_ack]
    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_window] do
      await [pcsu_powerdown and
             not jmp_e]
    end
  end;
  pause
end
```

Why is Esterel More Succinct?

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
```

Esterel:

```
abort
  await normal_ack
when error_ack
```

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

Shannon and Retiming

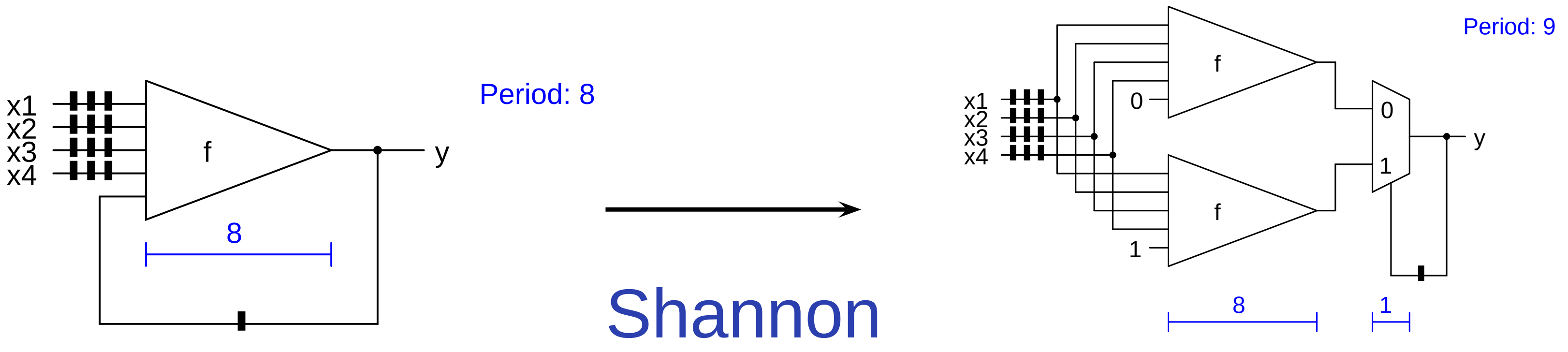
Cristian Soviani, Olivier Tardieu, and Stephen A. Edwards.

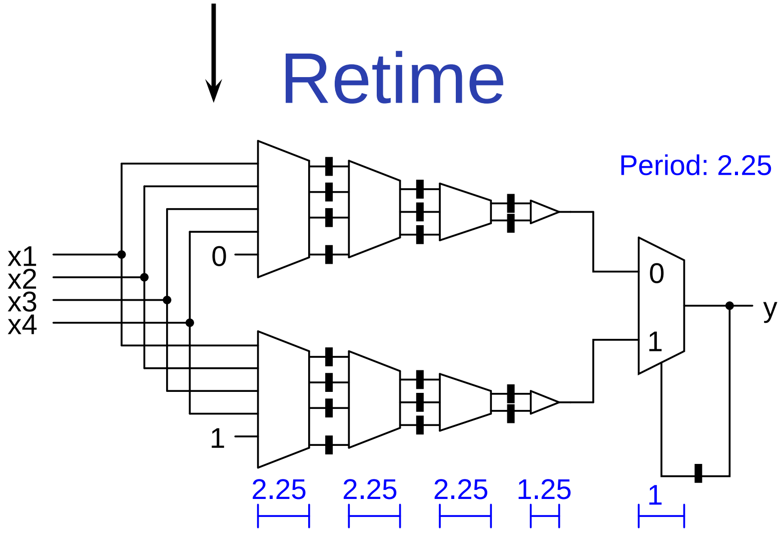
Optimizing Sequential Cycles through Shannon Decomposition and Retiming.

Proceedings of Design Automation and Test in Europe (DATE).

Munich, Germany, March 2006.

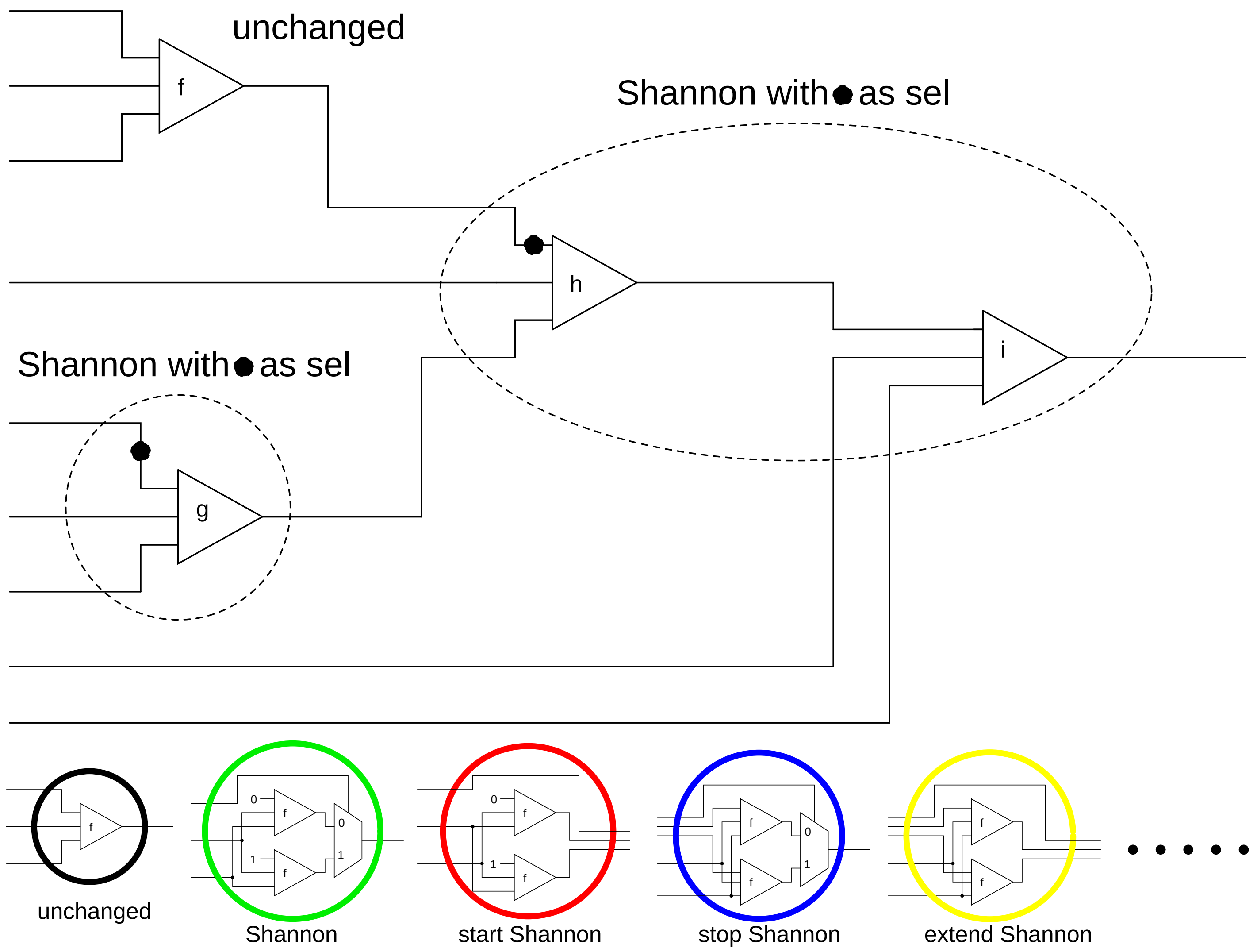
Motivating Example



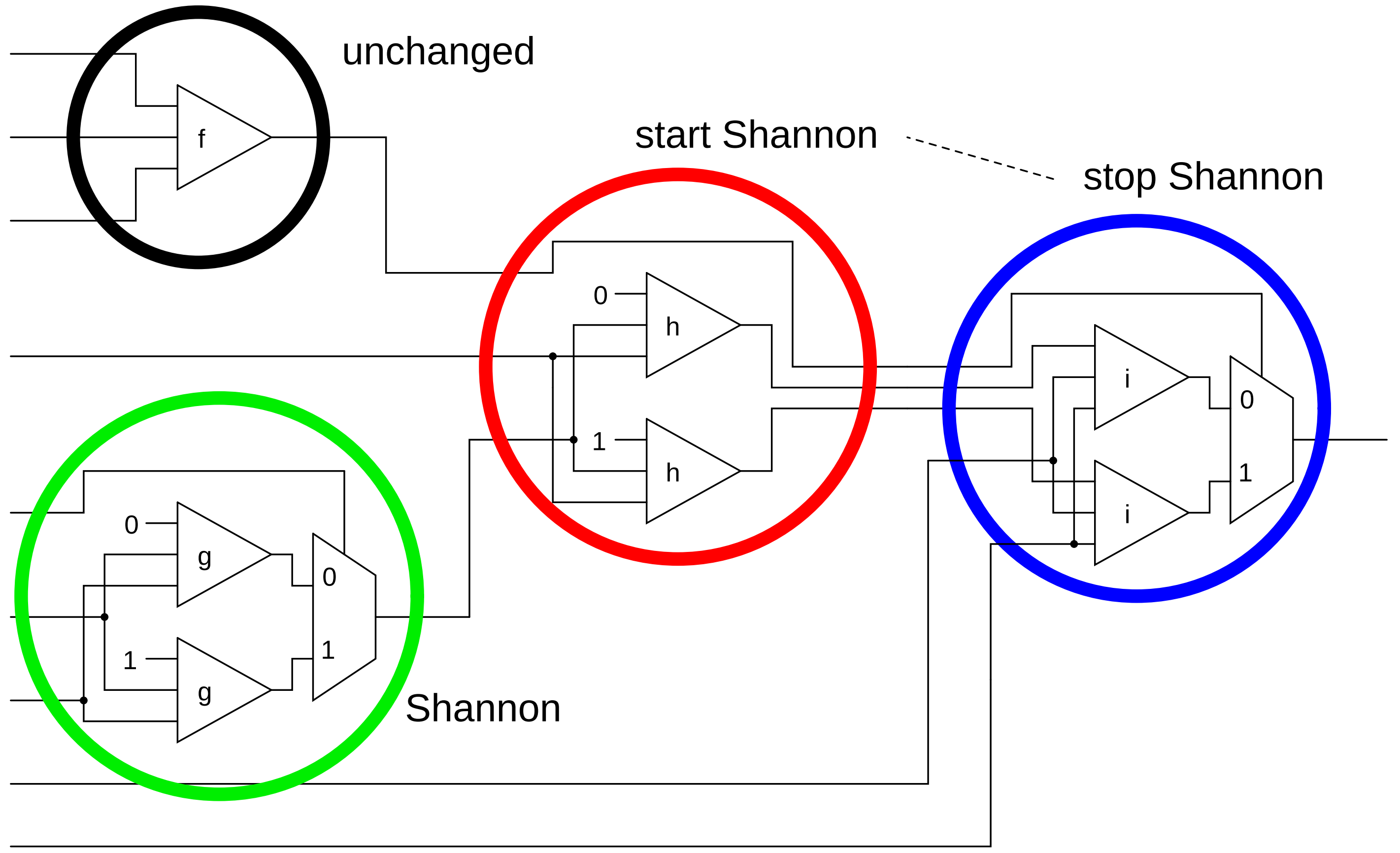


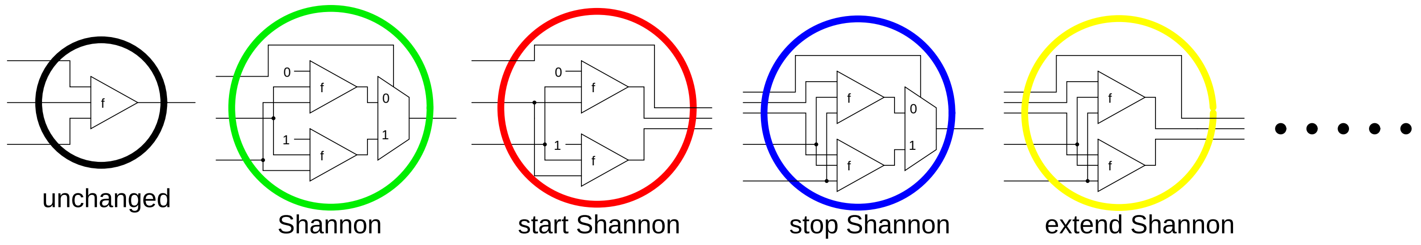
Tight feedback loop improved by combining the two techniques

Treat Shannon as a Covering Probler

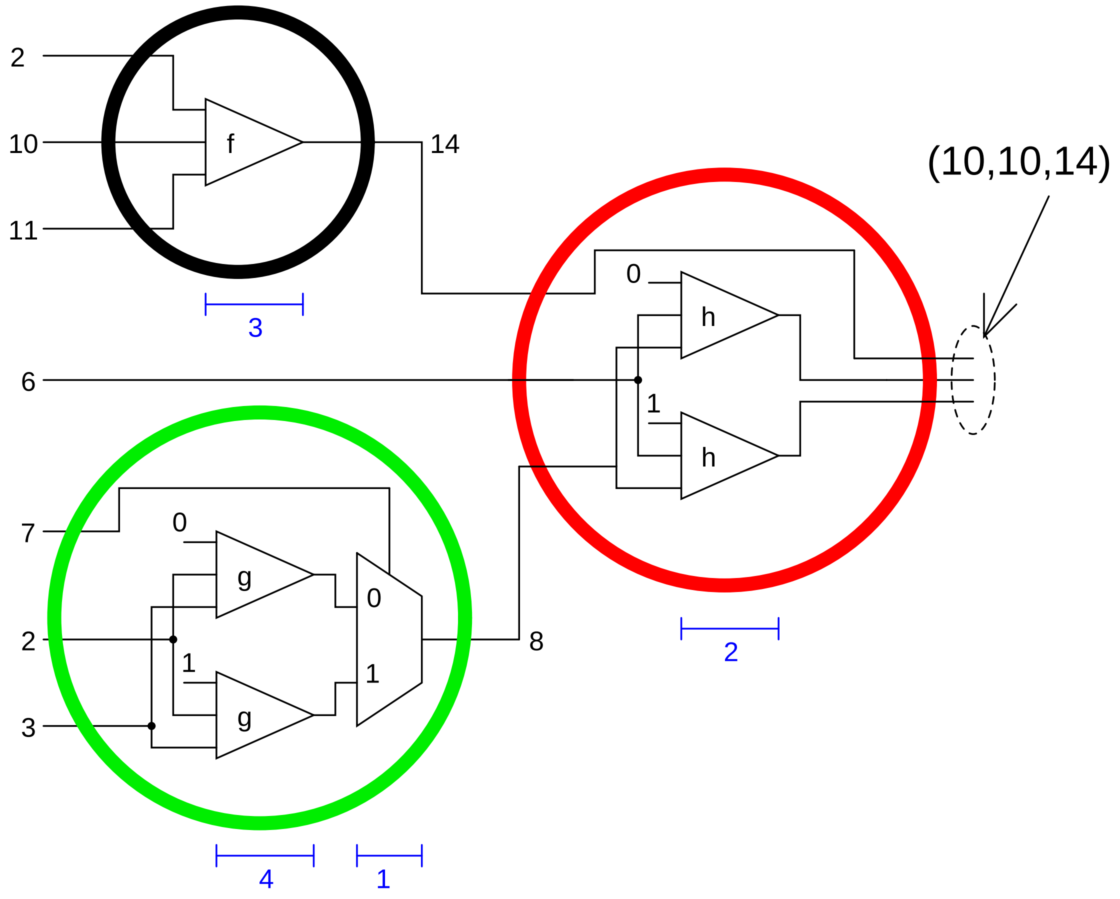


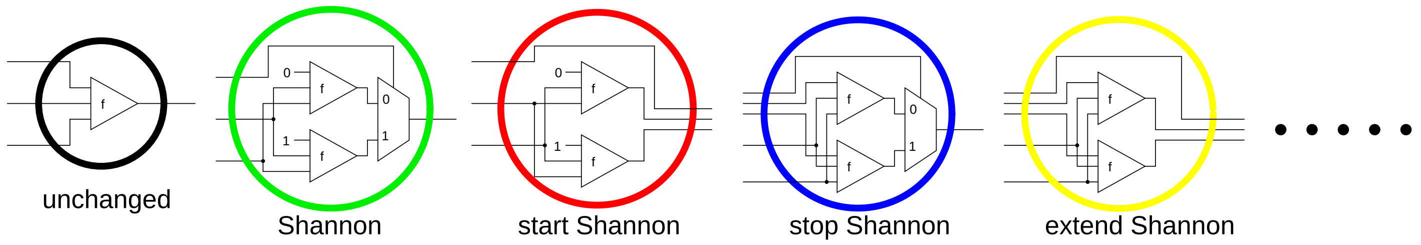
Treat Shannon as a Covering Probler



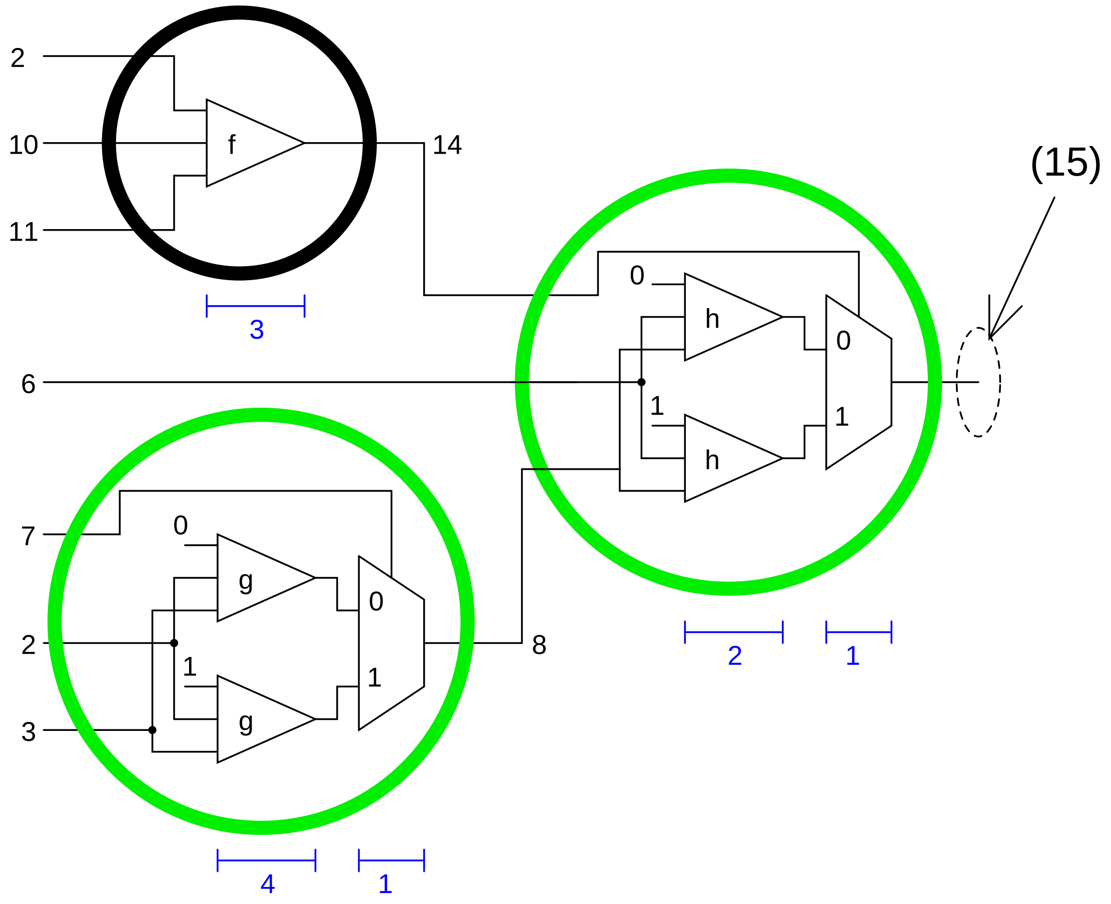


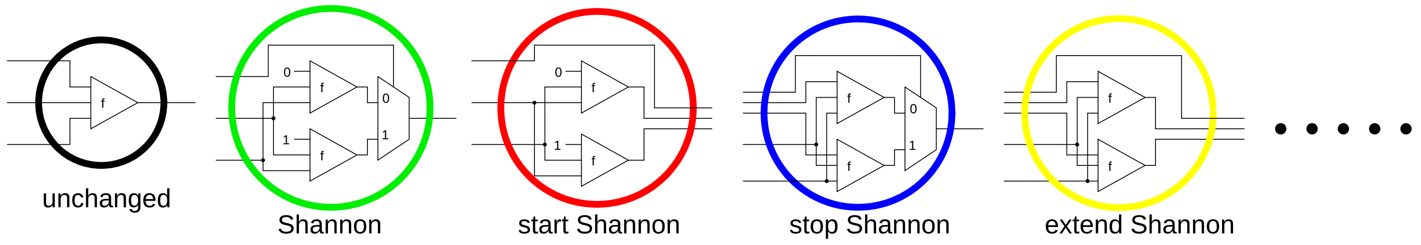
Considering Node Variants



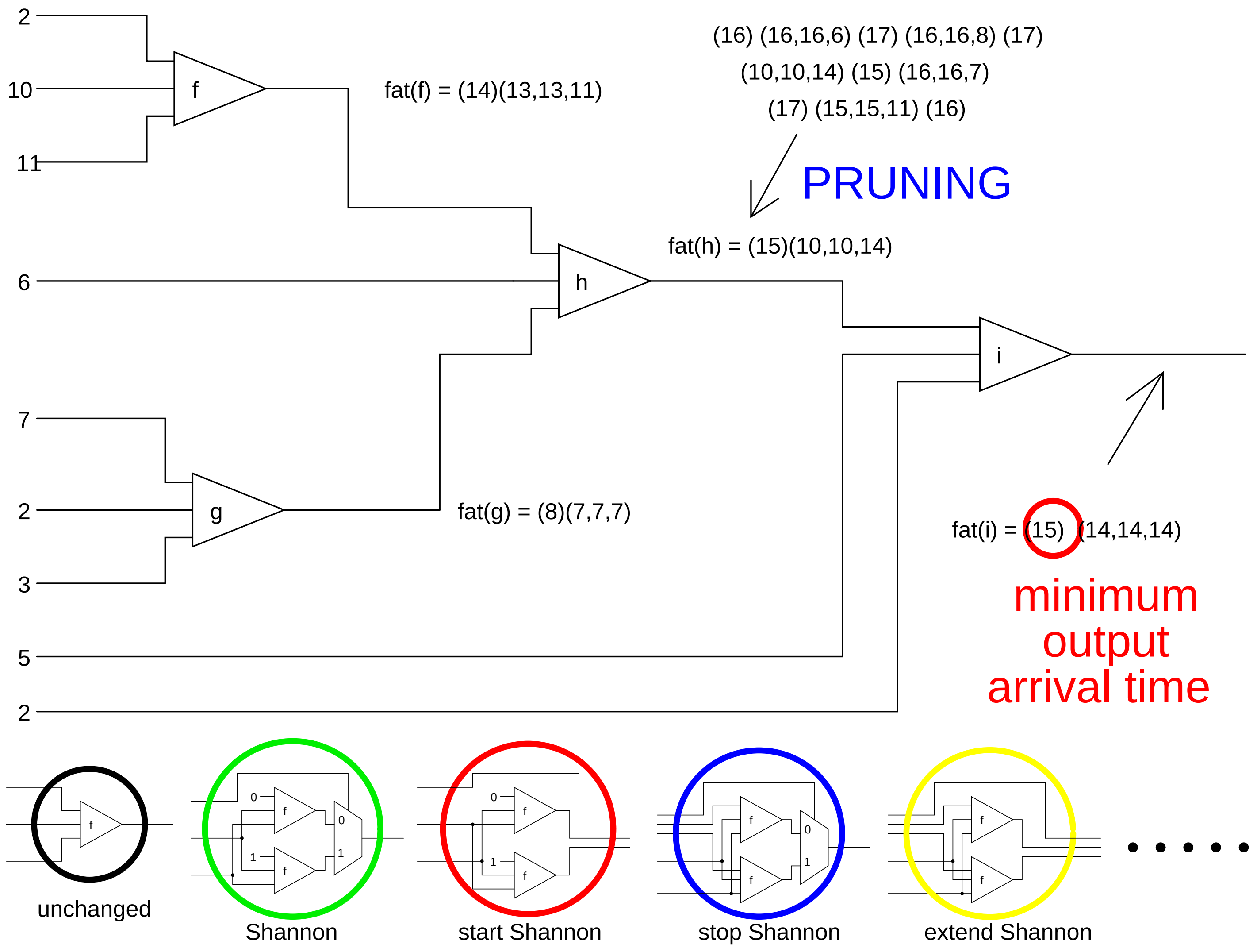


Considering Node Variants

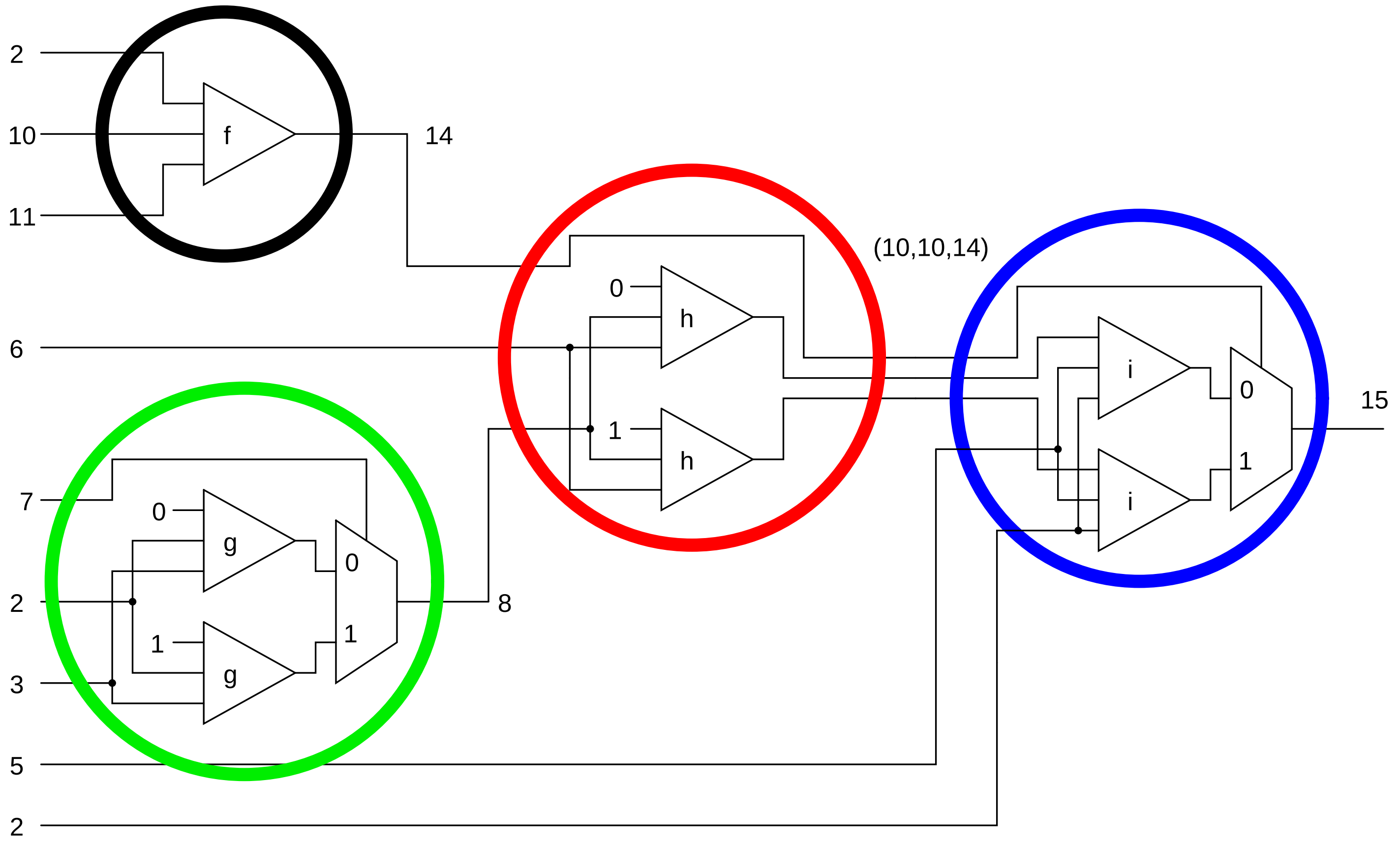


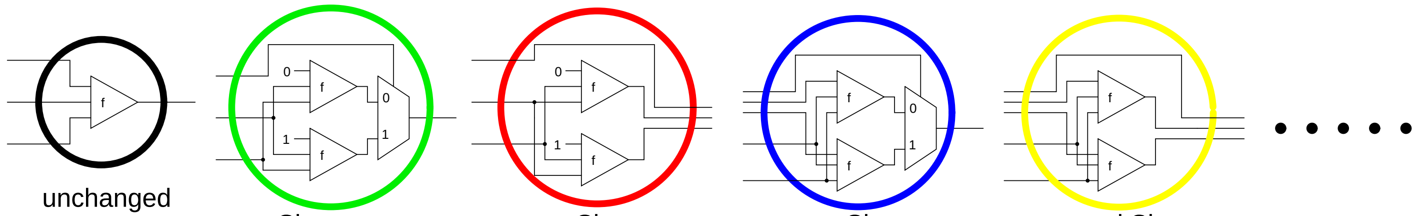


Pruning Node Variants



Best Solution





Results on ISCAS89 Benchmarks

	reference		retimed		Sh. + ret.		time	speed	area
	period	area	period	area	period	area	(s)	up	penalty
s510	8	184	8	184	8	184	0.5		
s641	11	115	11	115	9	122	1.1	22%	6%
s713	11	118	11	118	10	121	0.9	10%	3%
s820	7	206	7	206	7	206	0.5		
s832	7	217	7	217	7	217	0.4		
s838	10	154	10	154	8	162	2.6	25%	5%
s1196	9	365	9	365	9	365	0.6		
s1423	24	408	21	408	13	460	3.8	61%	12%
s1488	6	453	6	453	6	453	0.7		
s1494	6	456	6	456	6	456	0.8		
s9234	11	662	8	656	8	684	6.7		
s13207	14	1382	11	1356	9	1416	18.0	22%	4%
s38417	14	7706	14	7652	13	7871	113	7%	3%

Synthesizing Pipelines

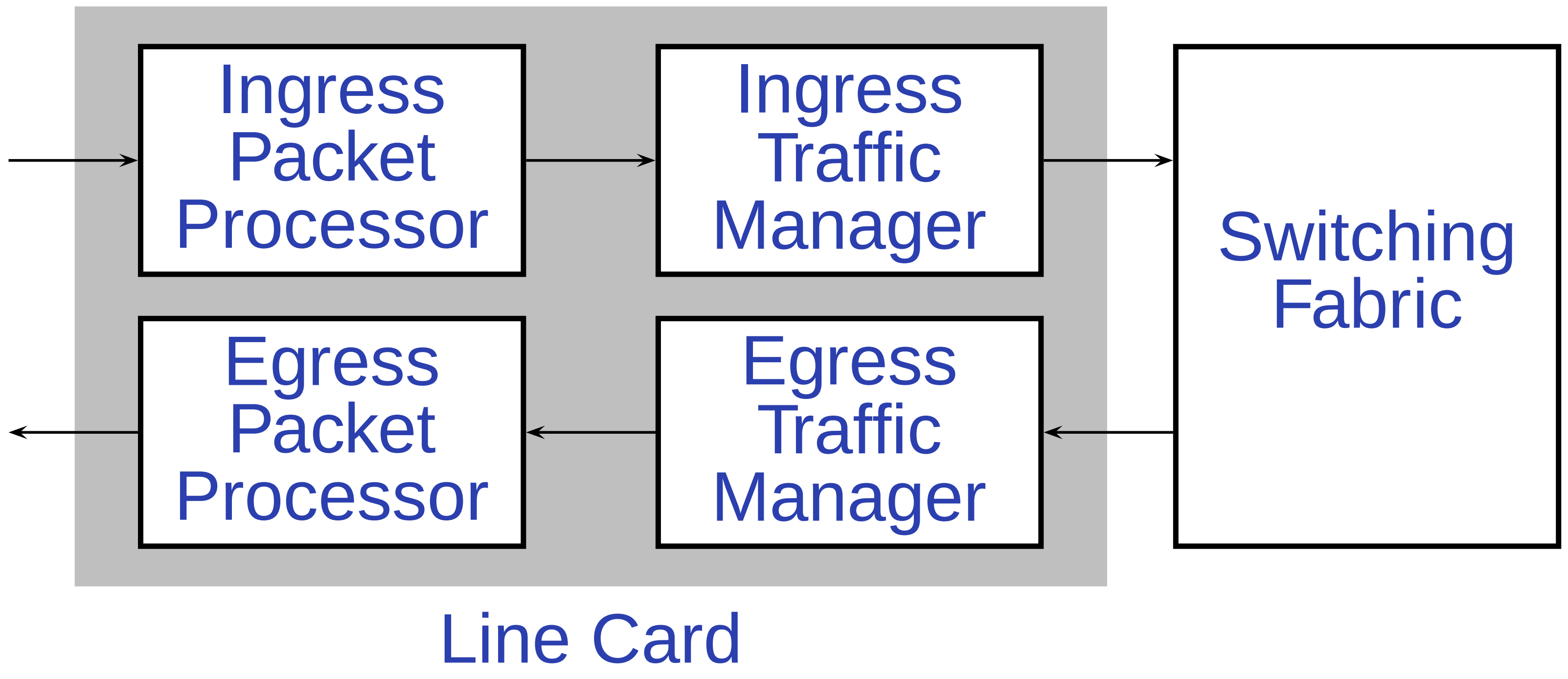
Cristian Soviani, Ilija Hadžić, and Stephen A. Edwards.

Synthesis of High-Performance Packet Processing Pipelines.

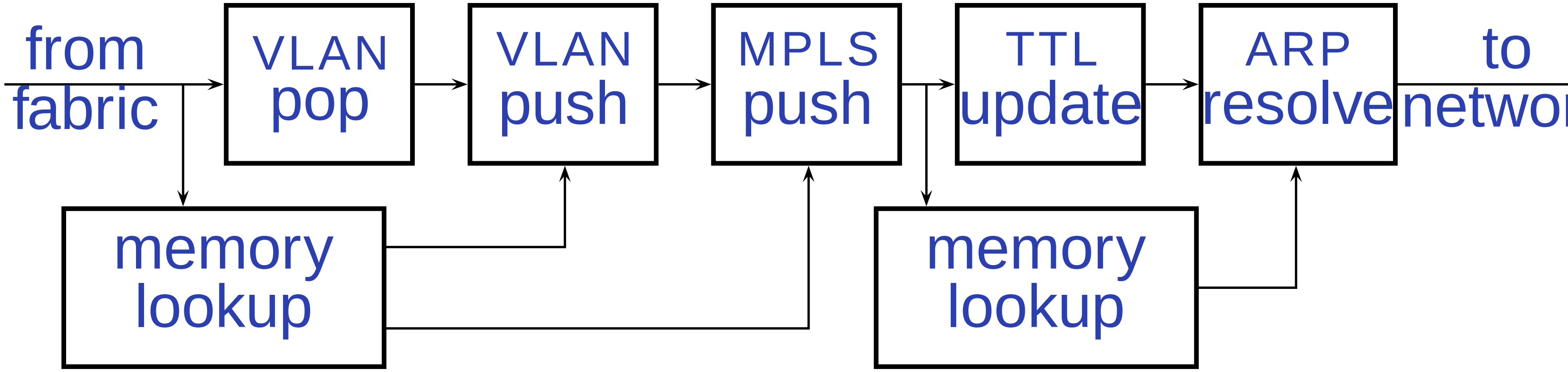
Submitted to the *Design Automation Conference (DAC)*.

San Francisco, California, July 2006.

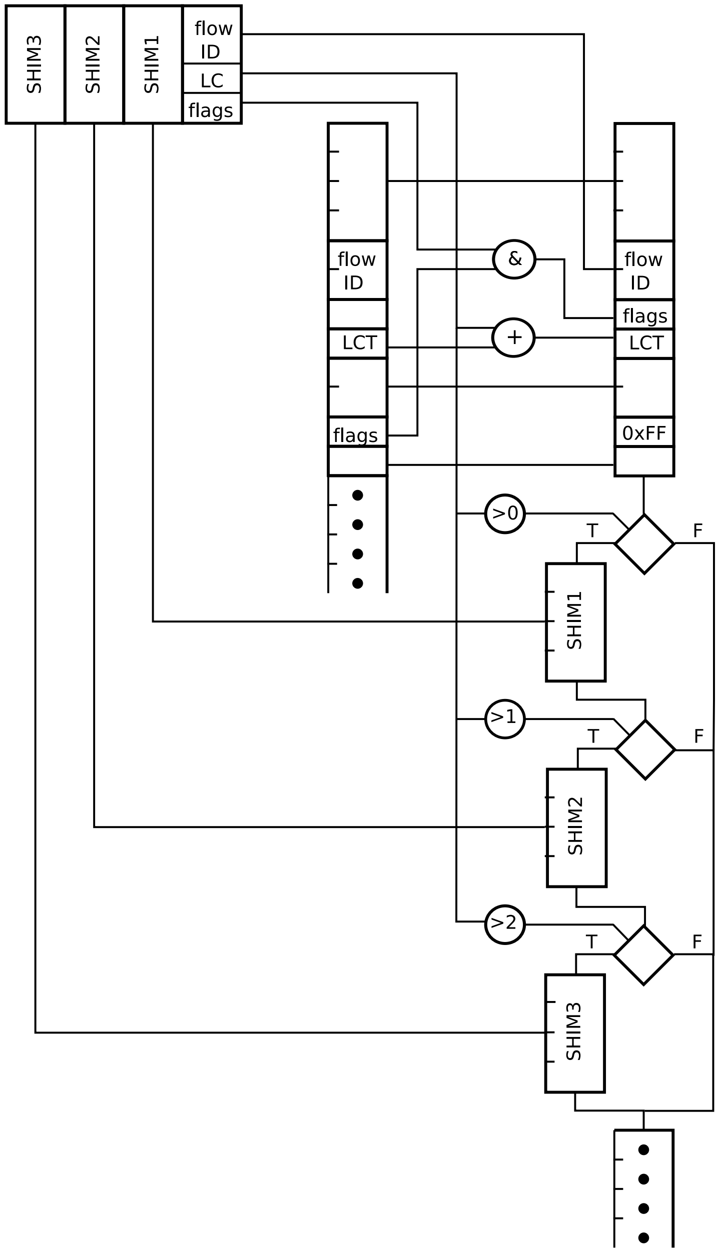
Packet Switch Architecture



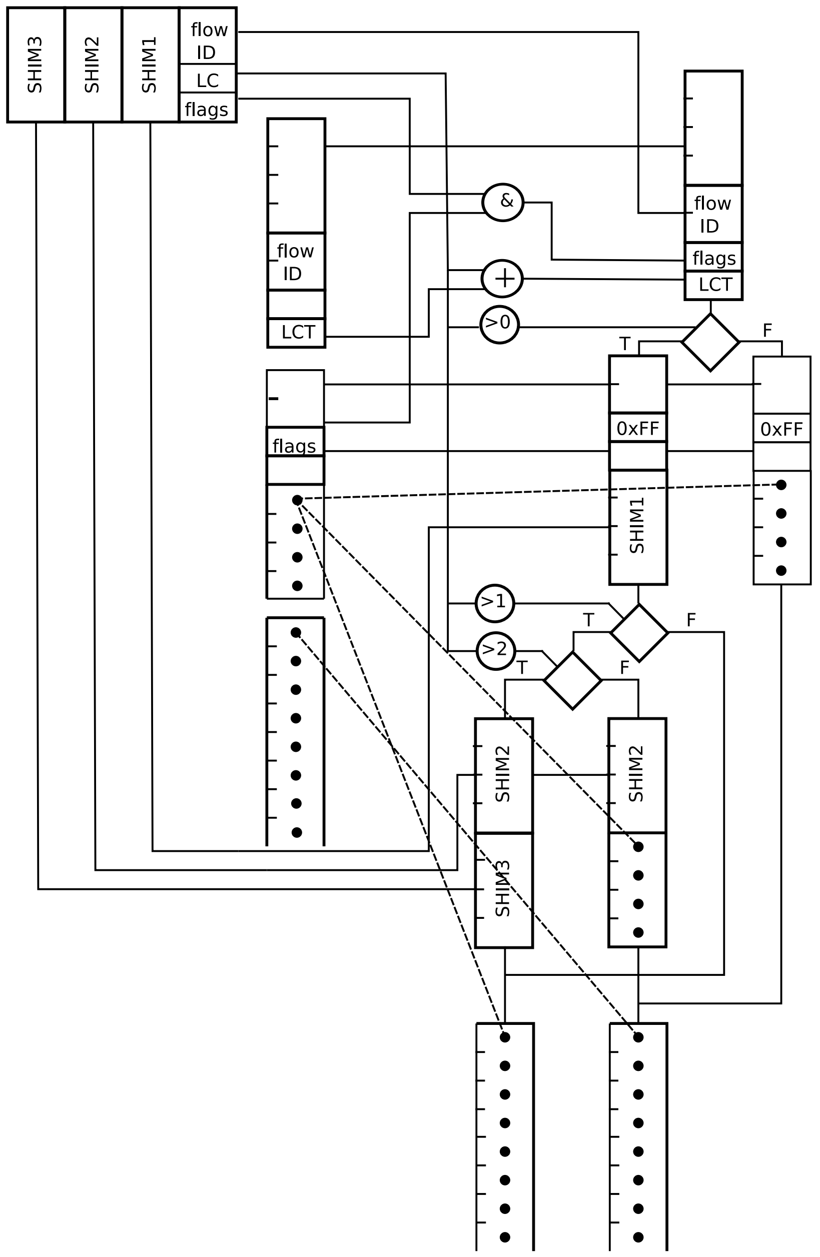
Typical Packet Pipeline



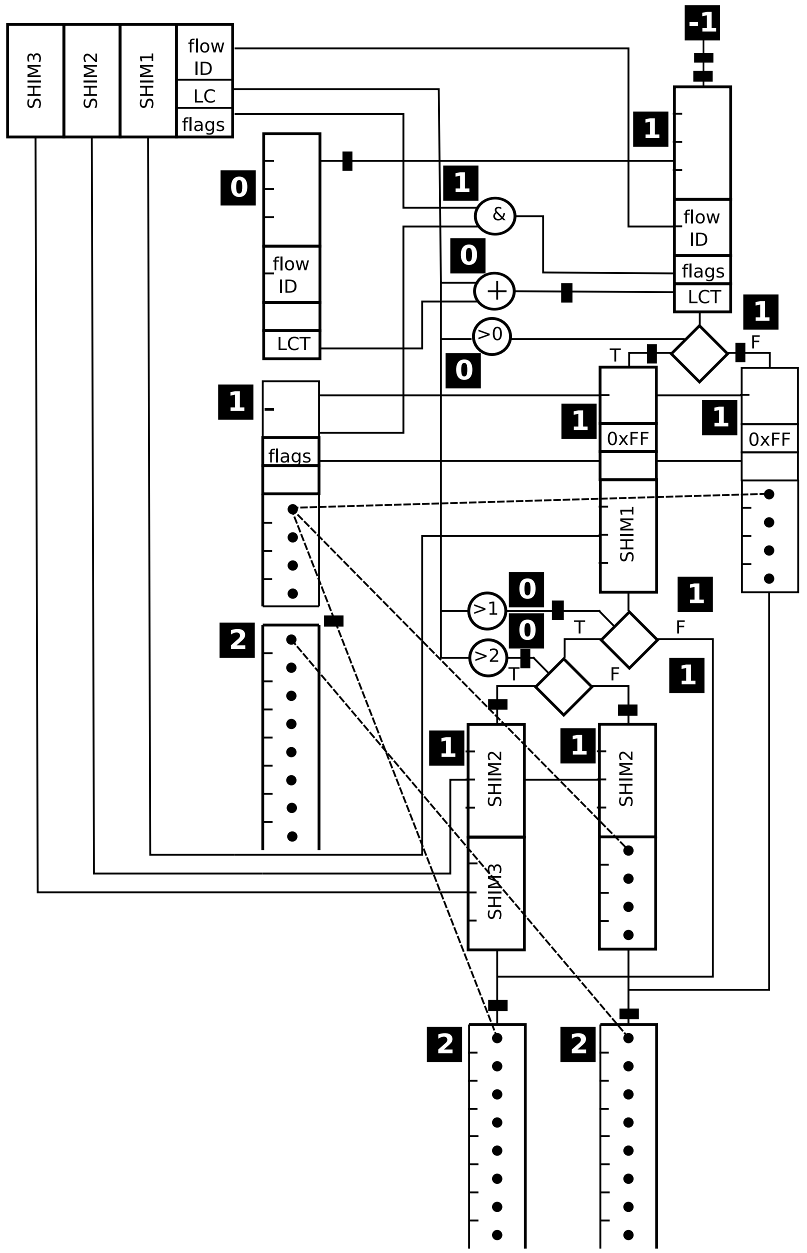
Packet Editing Graph



Block Up into 64-bit Words



Add Cycle Boundaries and Delays



RTL synthesis straightforward from here

Able to achieve 40 GB/s on an FPGA: as good as by hand

Much easier than hand-coding RTL

Tool handles tedious bookkeeping, FSM synthesis

Dycyclifying Circuits

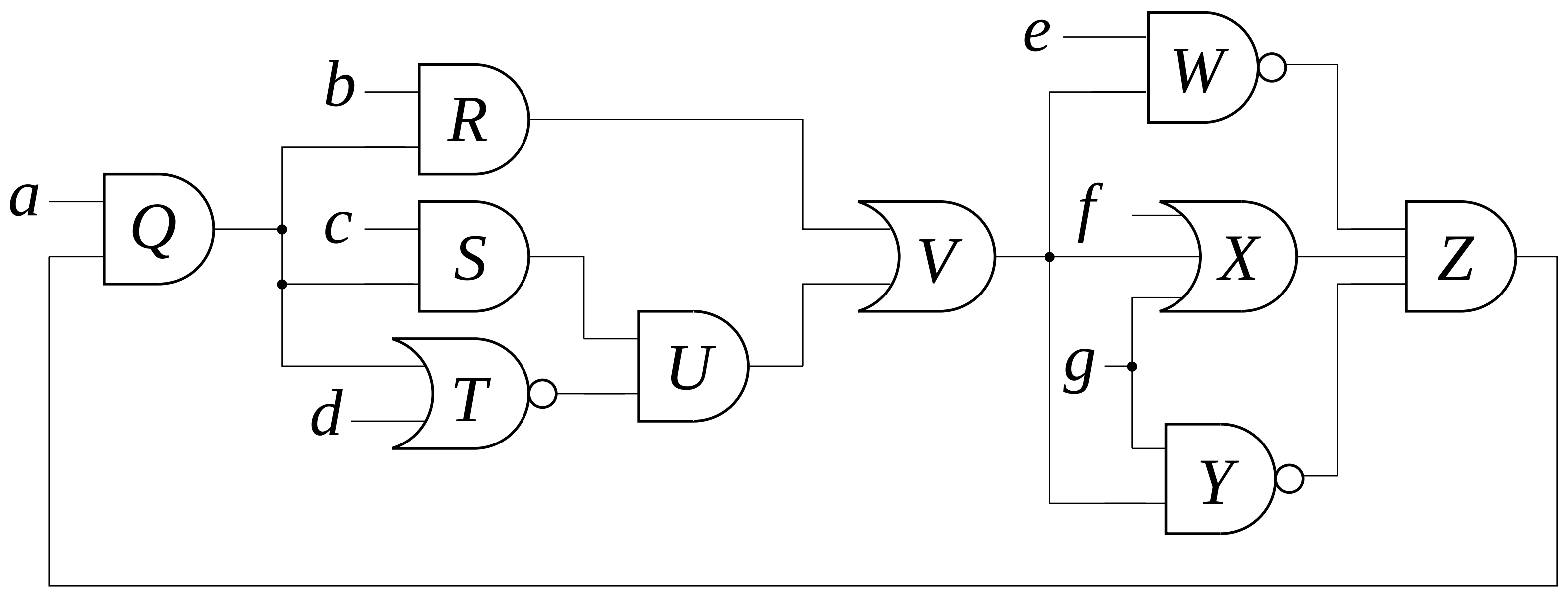
Osama Neiroukh, Stephen A. Edwards, and Xiaoyu Song.

An Efficient Algorithm for the Analysis of Cyclic Circuits.

Proceedings of the International Symposium on VLSI (ISVLSI).

Karlsruhe, Germany, March 2006.

Example



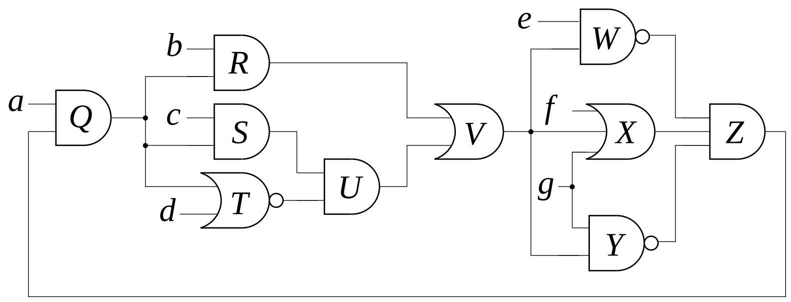
Example

1: Apply controlling values

Assignment	Frontier	At Frontier	Acyclic
$\{a = 0\}$	$\{\}$		$\surd$
$\{b = 0\}$	$\{V\}$	$R = 0$	
$\{c = 0\}$	$\{V\}$	$U = 0$	
$\{d = 1\}$	$\{V\}$	$U = 0$	
$\{e = 0\}$	$\{Z\}$	$W = 1$	
$\{f = 1\}$	$\{Z\}$	$X = 1$	
$\{g = 0\}$	$\{Z\}$	$Y = 1$	
$\{g = 1\}$	$\{Z\}$	$X = 1$	

2: Merge to “break logjams”

Gate	Assignment	Frontier	Acyclic
V	$\{b = 0, c = 0\}$	$\{\}$	$\surd$
V	$\{b = 0, d = 1\}$	$\{\}$	$\surd$
Z	$\{e = 0, f = 1, g = 0\}$	$\{\}$	$\surd$



Result:

$\{a = 0\}$

$\{b = 0, c = 0\}$

$\{b = 0, d = 1\}$

$\{e = 0, f = 1, g = 0\}$

Experimental Results

Circuit	Netlist Gates	SCC Gates	[Edwards 03] PAs	[Edwards 03] time	New PAs	New time	Acyclic PAs
arbiter5	213	25	257	1.3	25	0.1	14
arbiter6	248	30	745	8	29	0.1	16
arbiter7	283	35	2205	69	33	0.2	18
arbiter8	318	40	6581	656	37	0.3	20
exp	124	69	54517	2868	23260	2	338
ex1	150	47	43777	2341	232	1	10
gary	177	32	-	-	290	0.6	11
planet	253	51	-	-	1489	0.3	22
s1488	272	61	-	-	588	0.2	89
table3	311	49	-	-	3604	1	38

Much faster than the DAC 2003 paper’s algorithm

Separate Compilation for Esterel

Jia Zeng and Stephen A. Edwards.

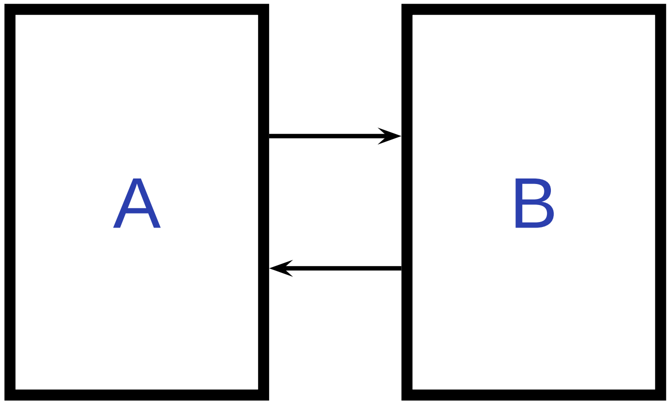
Separate Compilation of Synchronous Modules.

Proceedings of the 2nd International Conference on Embedded Software and Systems (ICESS).

Xian, China, December 2005.

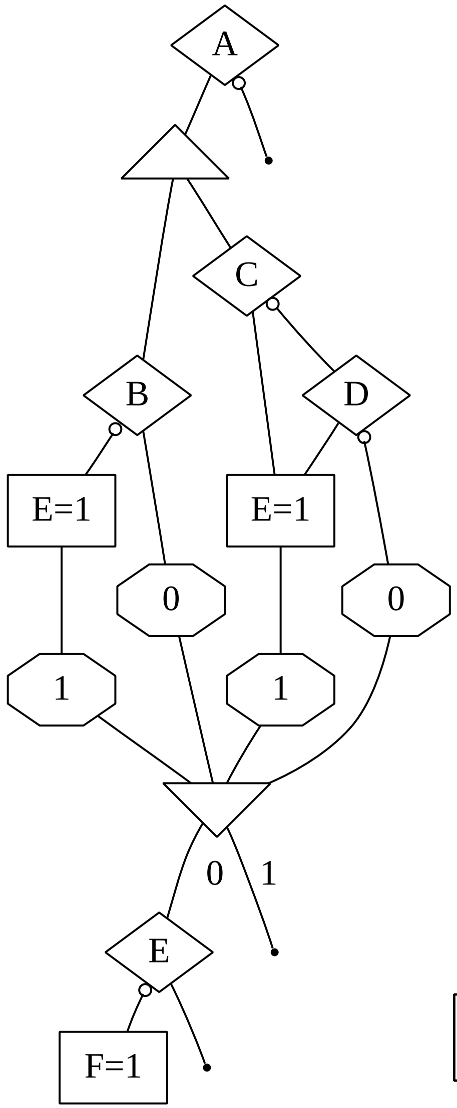
The Separate Compilation Challenge

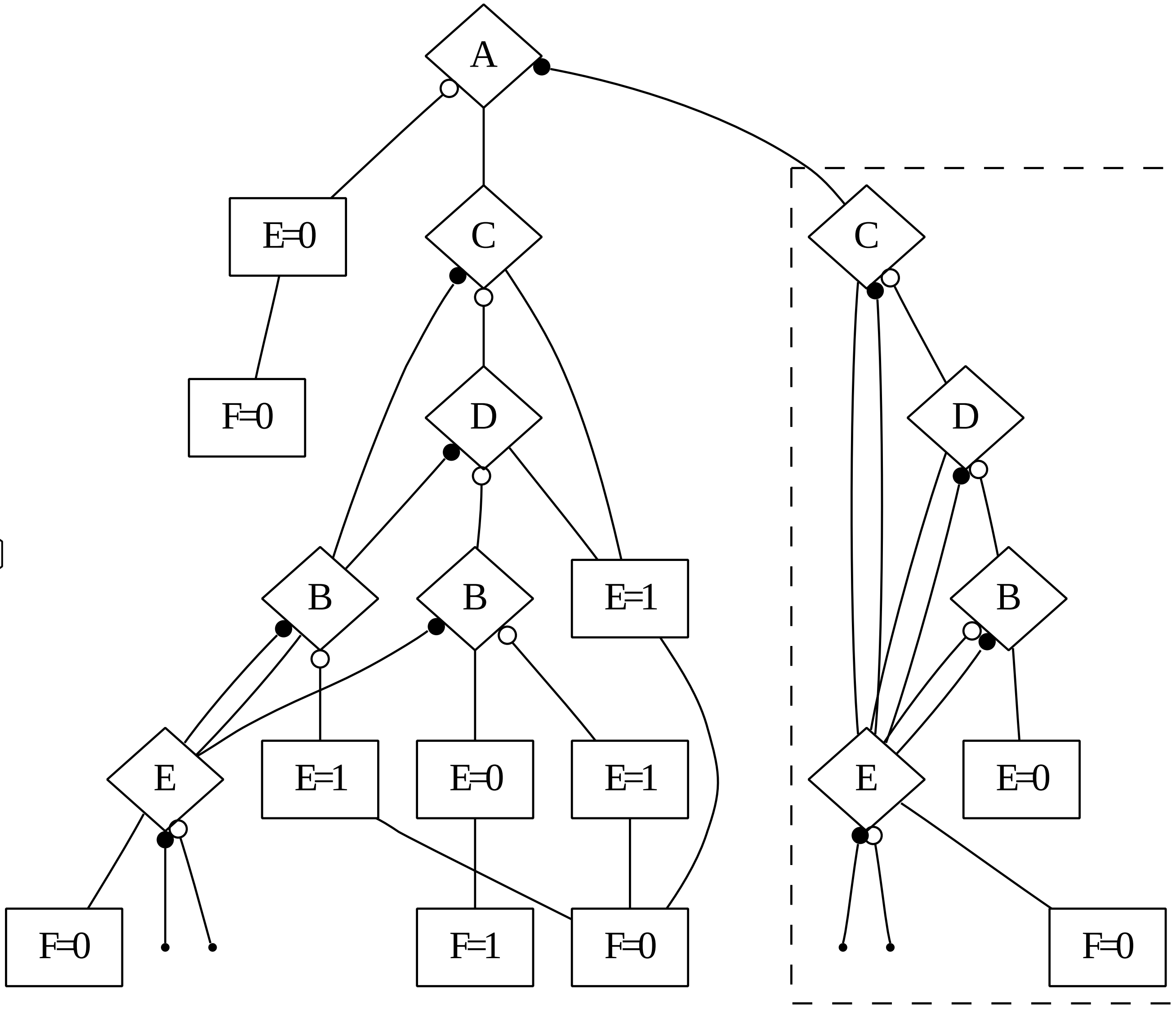
Connecting two synchronous blocks tricky: in what order should they be simulated?



Our solution: compile A and B such that they respond to “don’t know yet” inputs.

Convert If-Else to If-Else-Don’t-Know





Experimental Results

Example	Lines	Average cycle times		
		Esterel V5	SCFG	3-Valued
comexp	88	1.67s	0.61s	0.80s
iwls3	70	1.04s	0.35s	0.26s
3vsim2	48	0.68s	0.32s	0.46s
multi3	120	1.39s	0.45s	0.47s

Shows the cost of adding code that handles the “don’t-know” case is reasonable.

Approximate Reachability for Esterel

Olivier Tardieu and Stephen A. Edwards.

Approximate Reachability for Dead Code Elimination in Esterel*.

In *Proceedings of the Third International Symposium on Automated Technology for Verification and Analysis (ATVA)*.

Taipei, Taiwan, October 2005.

An Esterel Virtual Machine

Becky Plummer, Mukul Khajanchi, and Stephen A. Edwards.

An Esterel Virtual Machine for Embedded Systems.

In *Proceedings of Synchronous Lanugages, Applications, and Programming (SLAP)*.

Vienna, Austria, March 2006.

An Esterel Virtual Machine

Goal: software code generation for small embedded systems.

Basic idea: trade speed for program size by building a language-specific virtual machine.

Contributions: instruction-level support for concurrency, mating code synthesis algorithm.

Experimental Results

Code sizes (percentage saved by VM):

Example	BAL	x86		H8	
dacexample	369	917	60%	842	57%
abcd	870	2988	71%	2648	68%
greycounter	1289	3571	64%	2836	55%
tcint	5667	11486	51%	10074	51%
atds-100	10481	38165	73%	26334	60%

Execution Speeds (slowdown due to VM):

Example	x86	BAL	
dacexample	0.06μs	1.1μs	$18\times$
tcint	0.28μs	1.1μs	$4\times$
atds-100	0.20μs	1.4μs	$7\times$

Publications 1

Cristian Soviani, Ilija Hadžić, and Stephen A. Edwards.
Synthesis of High-Performance Packet Processing Pipelines.
Submitted to the *Design Automation Conference (DAC)*, San Francisco, California, July 2006.

Osama Neiroukh, Stephen A. Edwards, and Xioyu Song.
An effi cient algorithm for the analysis of cyclic circuits.
In *Proceedings of the Symposium on VLSI (ISVLSI)*, Karlsruhe, Germany, March 2006.

Cristian Soviani, Olivier Tardieu, and Stephen A. Edwards.
Optimizing Sequential Cycles through Shannon Decomposition and Retiming.
In *Proceedings of Design Automation and Test in Europe (DATE),* Munich, Germany, March 2006.

Publications 2

Jia Zeng and Stephen A. Edwards.
Separate Compilation of Synchronous Modules.
In *Proceedings of the 2nd International Conference on Embedded Software and Systems (ICESS)*, Xian, China, December 2005.

Olivier Tardieu and Stephen A. Edwards.
Approximate Reachability for Dead Code Elimination in Esterel*.
In *Proceedings of the Third International Symposium on Automated Technology for Verification and Analysis (ATVA),* Taipei, Taiwan, October 2005.

Becky Plummer, Mukul Khajanchi, and Stephen A. Edwards.
An Esterel Virtual Machine for Embedded Systems.
In *Proceedings of Synchronous Lanugages, Applications, and Programming (SLAP)*.
Vienna, Austria, March 2006.

Publications 3

Stephen A. Edwards and Olivier Tardieu.
SHIM: A Deterministic Model for Heterogeneous Embedded Systems.
In *Proceedings of the ACM Conference on Embedded Software (Emsoft)*, Jersey City, NJ, September 2005.

Stephen A. Edwards and Olivier Tardieu.
Deterministic Receptive Processes are Kahn Processes.
In *Proceedings of the 3rd International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, Verona, Italy, July 2005.

Cristian Soviani and Stephen A. Edwards.
Challenges in Synthesizing Fast Control-Dominated Circuits.
In *Proceedings of the International Workshop on Logic and Synthesis (IWLS)*, Lake Arrowhead, California, June, 2005.

Publications 4

Stephen A. Edwards.
SHIM: A Language for Hardware/Software Integration.
In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP)*, Edinburgh, Scotland, April 2005.

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

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

Publications 5

Stephen A. Edwards, Vimal Kapadia, and Michael Halas.
Compiling Esterel into Static Discrete-Event Code.
In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP 2004)*. Barcelona, Spain, March 28, 2004.

Stephen A. Edwards.
Making Cyclic Circuits Acyclic.
In *Proceedings of the 40th Design Automation Conference (DAC 2003).* Anaheim, California, June 2-6, 2003. pp. 159-162.

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