Prior and Future Research

Prof. Stephen A. Edwards

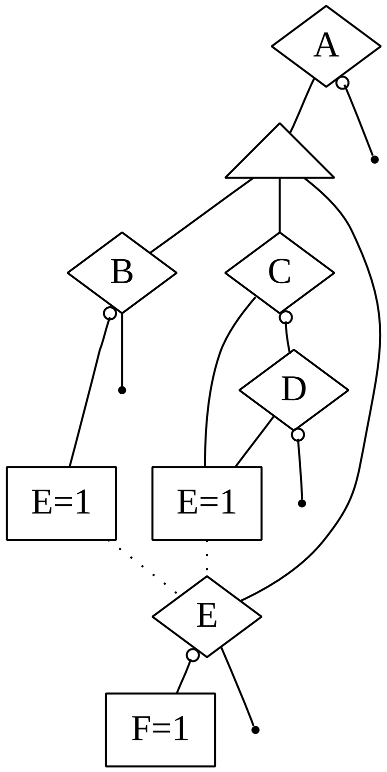
sedwards@cs.columbia.edu

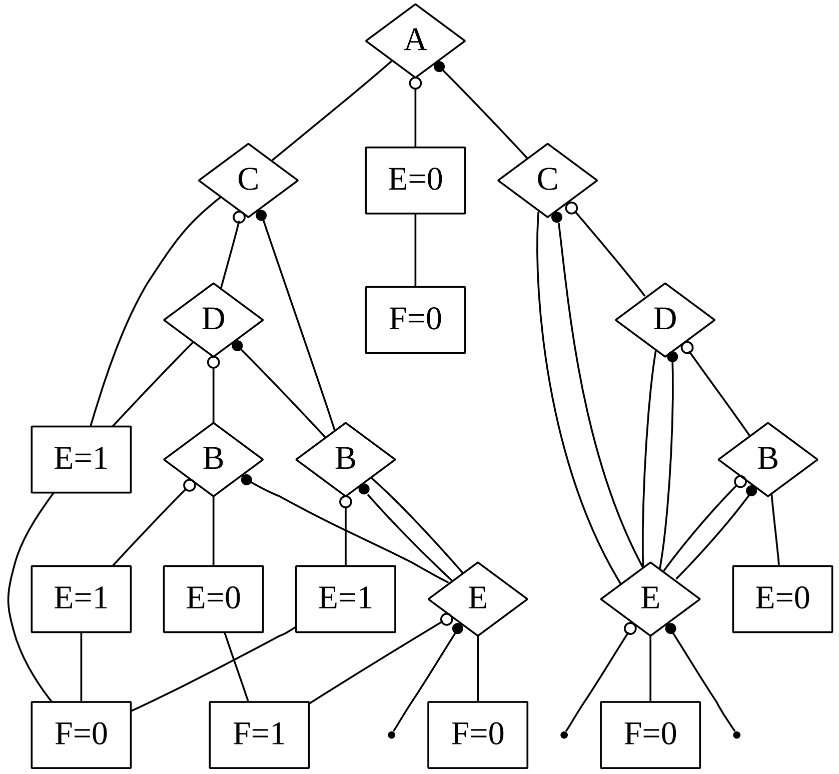
Columbia University

Compiling Esterel

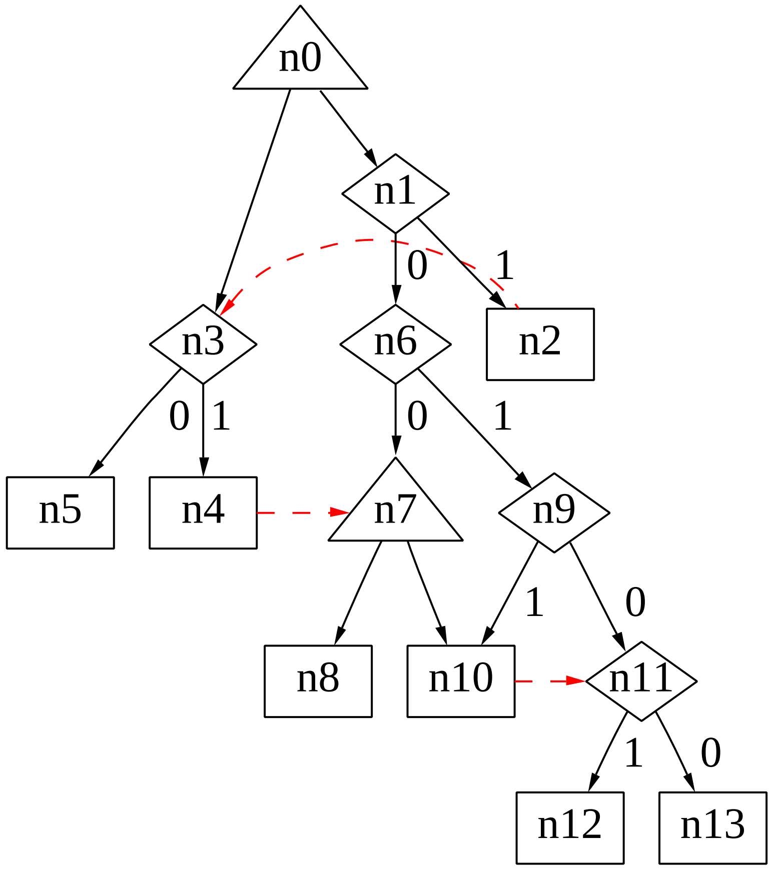
- Two- to three-valued extrapolation for code generation (LCTES, 2005)
- Very efficient C code generation from Program Dependence Graphs (LCTES, 2004)
- Static Event-driven C code generation (SLAP, 2004)
- Open-source Columbia Esterel Compiler (ongoing)
- Efficient C code generation from Esterel (DAC, 2000)

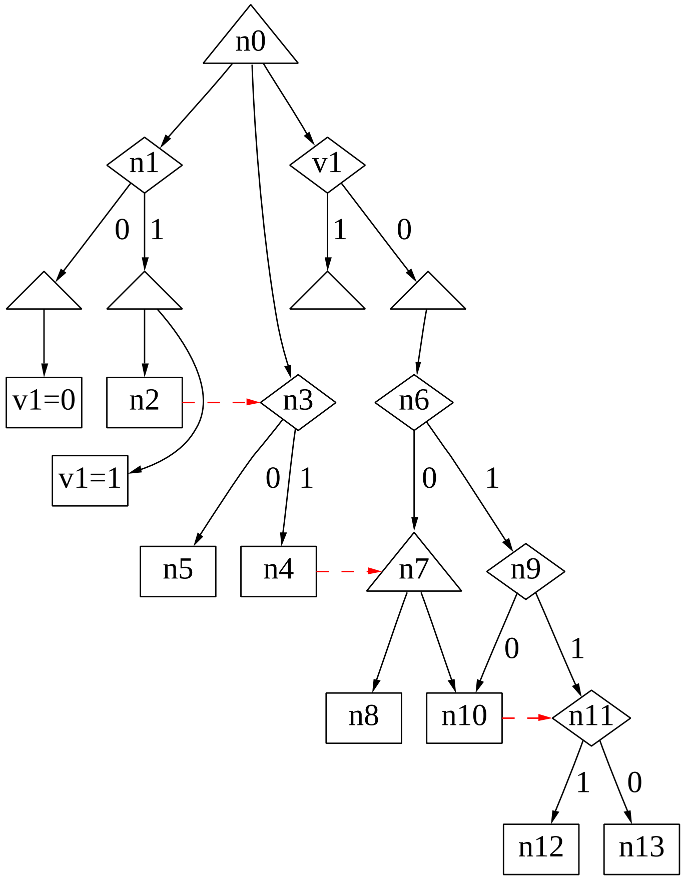
Two- to Three-valued Code Generatio



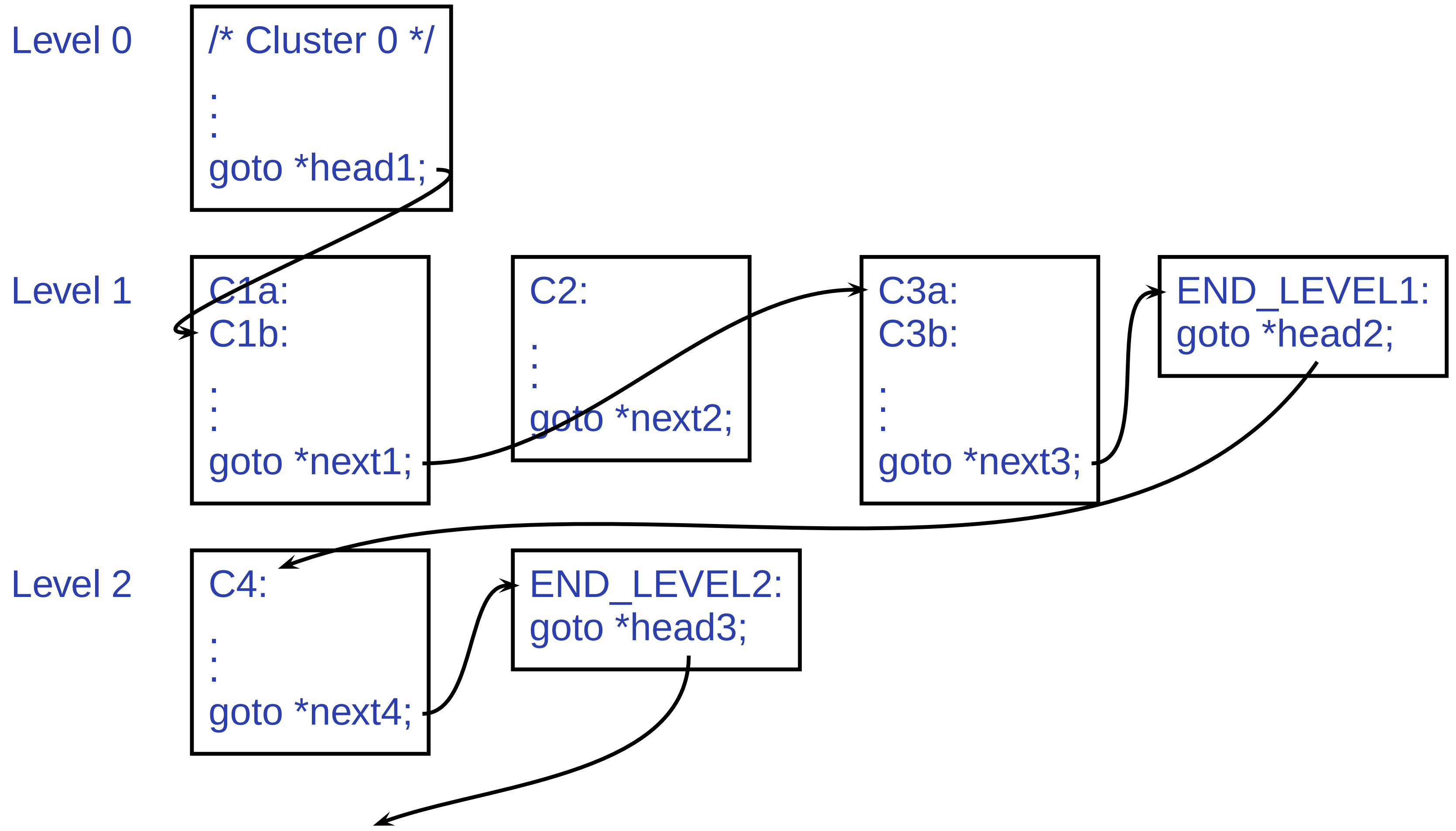


C-code generation from PDGs

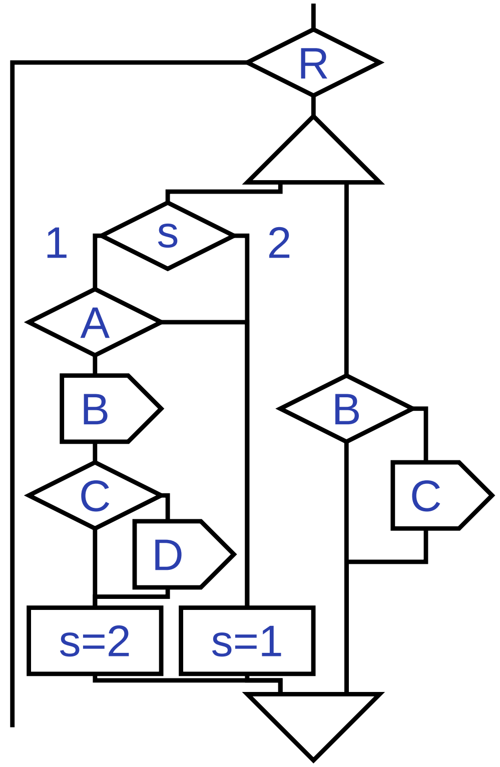


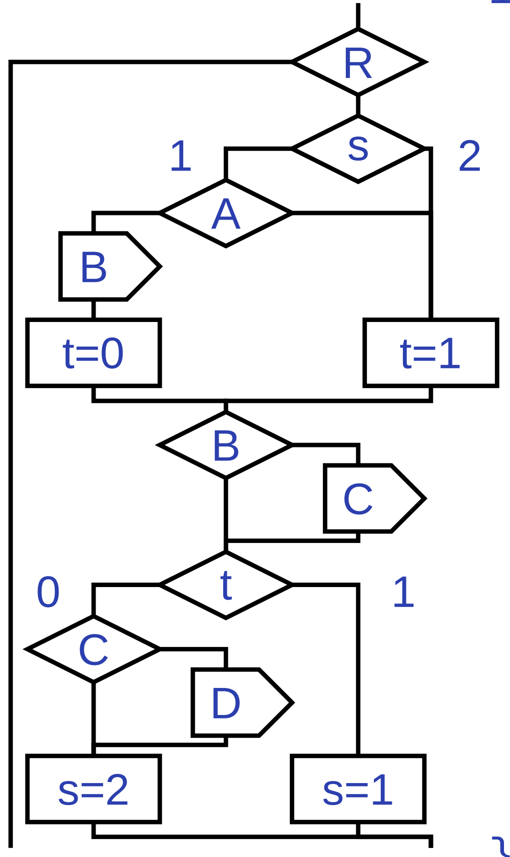


Static Event-Driven C Generation



Efficient C Generation from Esterel





```
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
  }
}
```

Hardware/Software Codesign

- SHIM: A language for hardware/software integration (SLAP, 2005)
- NDL: A language for device drivers (LCTES, 2004)
- Porting a network service to the RMC2000 (DATE, 2003)
- The Synchronous/Reactive domain for Ptolemy (SCP, 2003)

SHIM

```
module timer {
  shared uint:32 counter; // Visible to HW and SW

  hw void count() {           // Hardware process
    counter = counter + 1;
  }

  out void reset_timer() {  // Software function
    counter = 0;
  }

  out uint get_time() {       // Software function
    return counter;
  }
}
```

NDL: Driver for NE2000

```
ioports {
  command = {
    0:  stop : trigger except 0,
    1:  start : trigger except 0,
    2:  transmit : trigger except 0,
    3..5:
        dmaState : {
          READING = #001
          WRITING = #010
          SENDING = #011
          DISABLED = #1**
        } volatile,
    6..7:
        registerPage : int{0..2}
  },

  critical function @(countersIrq) {
    rxFrameErrors += frameAlignErrors;
    rxCrcErrors += crcErrors;
    rxMissedErrors += packetErrors;
    countersIrq = ACK;
  }
```

Porting to an 8-bit microcontroller

```
int echo_server() {
  int sock, newsock, len;
  struct sockaddr_in addr;
  char buf[LEN];

  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    return -1;

  memset(&addr, 0, sizeof(addr));
  addr.sin_family      = AF_INET;
  addr.sin_addr.s_addr = htonl(INADDR_ANY);
  addr.sin_port        = htons(MYPORT);
  if ( bind(sock, (struct sockaddr *) &addr,
            sizeof(struct sockaddr_in)) < 0 ) return -1;
  if ( listen(sock, LISTENQ) < 0 ) return -1;
  for (;;) {
    if ((newsock = accept(sock, NULL, NULL) ) < 0 )
      return -1;
    if ((len = recv(newsock, buf, LEN, 0)) < 0)
      return -1;
    if (send(newsock, buf, len, 0) < 0) return -1;
    close(conn_s);
  }
}
```

Berkeley Sockets (Original)

```
int echo_server()
{
  tcp_Socket sock;
  int status;
  char buf[LEN];

  sock_init();
  for (;;) {
    tcp_listen(&sock, PORT, 0, 0, NULL, 0);
    sock_wait_established(&sock, 0, NULL, &status);
    sock_mode(&sock, TCP_MODE_ASCII);
    while (tcp_tick(&sock)) {
      sock_wait_input(&sock, 0, NULL, &status);
      if (sock_gets(&sock, buf, LEN))
        sock_puts(&sock, buf);
    }
  }
}
```

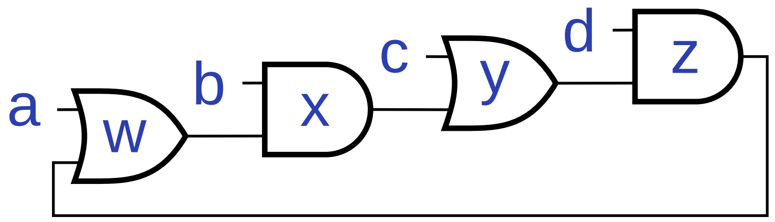
Dynamic C API

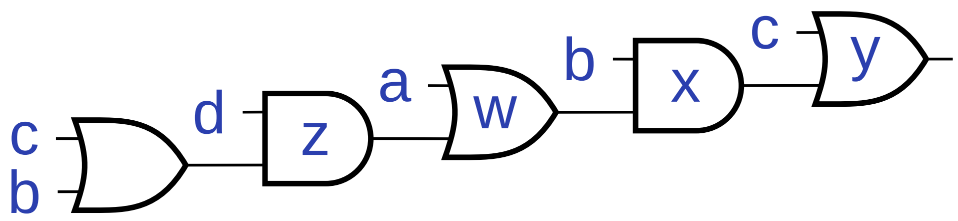
Logic Synthesis

- Combined Shannon Decomposition and Retiming (IWLS, 2005)
- Making Cyclic Circuits Acyclic (DAC, 2003)

Making Cyclic Circuits Ayclic

Given a cyclic circuit that is combinational for some inputs, create a similarly-structured acyclic circuit that computes the same combinational function.



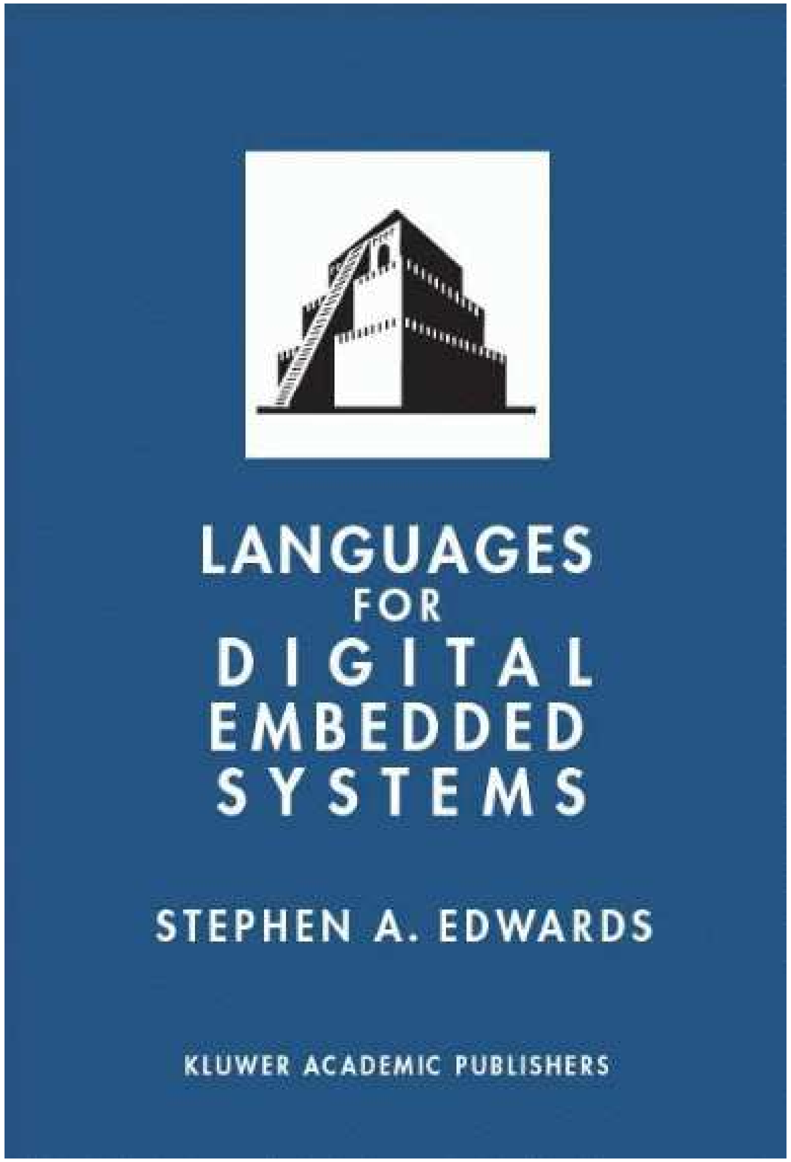


abcd	wxyz
0000	0000
0001	0000
0010	0010
0011	1011
0100	0000
0101	
0110	0010
0111	1111
1000	1000
1001	1000
1010	1010
1011	1011
1100	1110
1101	1111
1110	1110
1111	1111

this input not combinational

Circuit after Rivest [1977]

Book (Kluwer, 2000)


LANGUAGES FOR DIGITAL EMBEDDED SYSTEMS
STEPHEN A. EDWARDS
KLUWER ACADEMIC PUBLISHERS

Verilog
VHDL
Assembly
C
C++
Java
Operating Systems
Kahn Process Networks
Synchronous Dataflow
Esterel
Polis
SDL
SystemC
CoCentric System Studio

Future Work

- Asynchronous, concurrent, deterministic hardware/software development environment (SHIM II)
- Source-level dead-code elimination for Esterel
- Accelerating concurrent Java code through static scheduling
- Static analysis and model-checking for NDL