# Challenges in Synthesizing Fast Control-Dominated Circuits

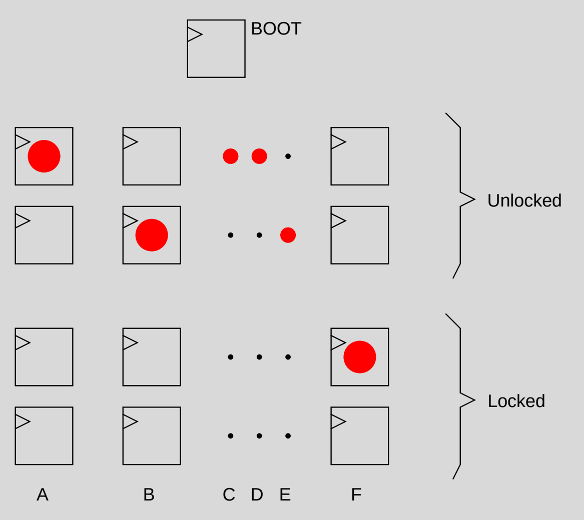
Cristian Soviani and Stephen A. Edwards, Columbia University

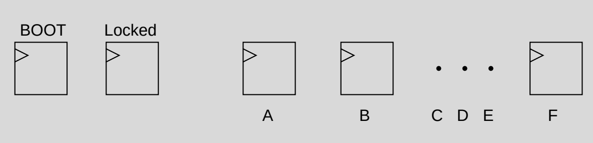
**Question:** Can we use the high-level information in Esterel source to generate efficient control circuits?

**Approach:** Manually optimize circuits generated by a syntax-directed translation and understand what insight was needed.

## Locking Button Controller (abcdef)

```
module ONE_BUTTON:
input BUTTON, LOCK;
output SELECTED_ON, SELECTED_OFF;
output LOCKED_ON,  LOCKED_OFF;
inputoutput SELECTED, LOCKED, UNLOCKED;

emit SELECTED_OFF; emit LOCKED_OFF;
loop
  trap WAIT_FOR_SELECTION in

    trap NOW_SELECTED in  % Wait to be selected
      loop
        abort
          await BUTTON do % S1
            exit NOW_SELECTED % We were pressed
          end await
        when LOCKED;      % We have been locked out
        await UNLOCKED    % S2: Wait to be released
    end end;

    loop                  % Selected or locked
      emit SELECTED_ON;   % We were selected
      emit SELECTED;      % Disable other buttons
      abort
        await             % S3
          case BUTTON     % User disabled us or
          case SELECTED   % other button pressed
        end await;
        emit SELECTED_OFF; % We lost the selection
        exit WAIT_FOR_SELECTION
      when LOCK;

      emit LOCKED_ON;     % We are now locked and
      emit SELECTED_OFF;  % no longer selected.
      emit LOCKED;        % Lock out others and
      await LOCK;         % S4: wait for unlock
      emit LOCKED_OFF;    % Announce it and
      emit UNLOCKED       % release other buttons.
end end end end
end module
```



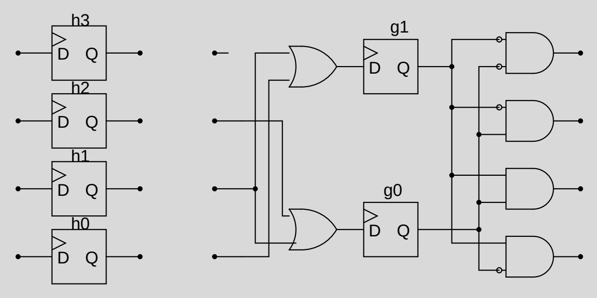


Observation: Each machine may be in one of four states, but together all must be in first two or second two.

Optimization: Replace local one-of-four encoding with a global encoding that uses one register for locked/unlocked status.

## Grey Code Counter

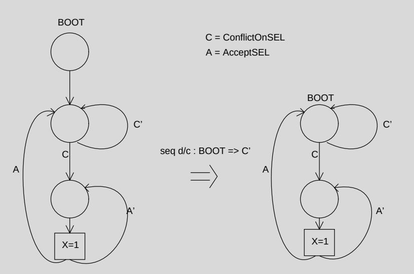
```
module Bit:
input CLK;
output B, CY;
loop
  await CLK;
  abort sustain B when CLK;
  emit CY;
  abort sustain B when CLK;
  await CLK;
  emit CY
end loop
end module
```



Optimization: Re-encode each bit of the counter using two instead of four registers.

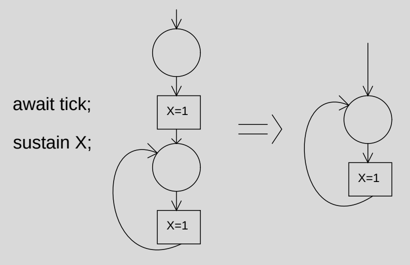
## Local Optimizations

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



Optimization: Merge initial state with later states. Legal only because actions in later state never occur in first cycle.

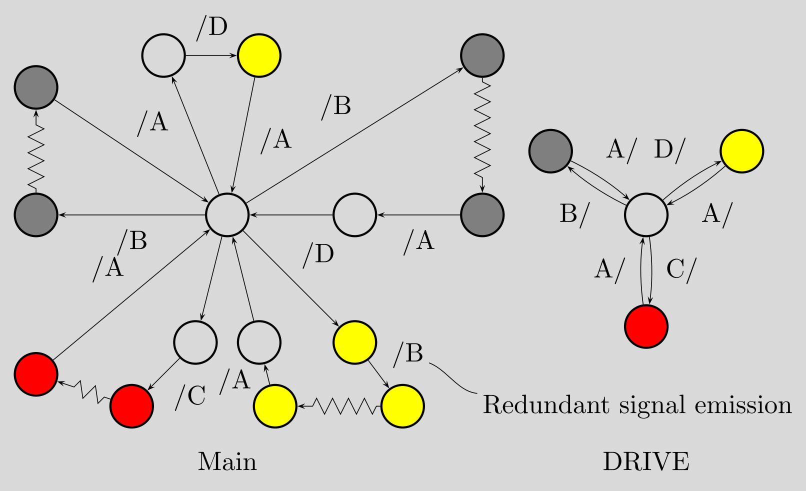
```
trap AckReceived in
  await tick;
  sustain TCRegOutCkDis
||
  await immediate ACK;
  exit AckReceived
end trap;
```



Optimization: Merge adjacent equivalent states. Classical state minimization induced by natural, but unfortunate coding style.

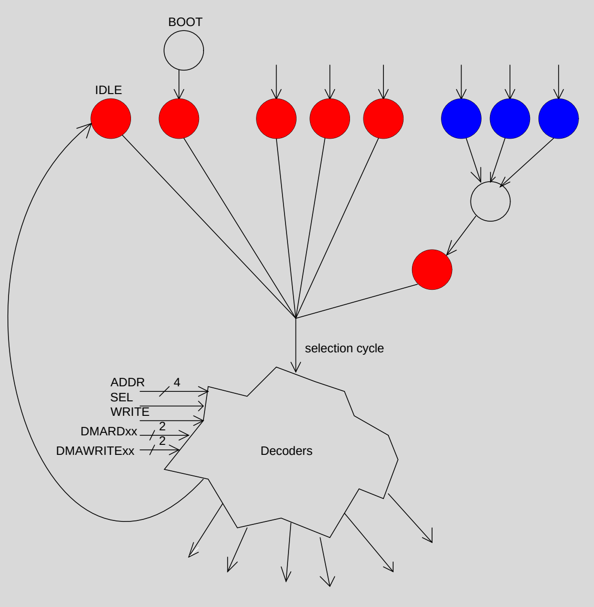
## Turbochannel Bus Controller (tcint)

```
pause;  % to avoid problems at boot time!
loop
  await  % DMA request or SEL
    case immediate [Fo_HF and DMAWrAddrRdy] do
      run DMA_WRITE
    case immediate [not Fi_HF and DMARdAddrRdy] do
      run DMA_READ

    case immediate SEL do  % SEL : decode opcode
      emit TagFlag;
      trap ReadSharedEnd, WriteSharedEnd in
        present [SEL and WRITE and not ADB24 and
                          ADB23 and not ADB22] then
          run WPOM
        else present [SEL and not WRITE and
            not ADB24 and ADB23 and not ADB22] then
          run RPOM; exit ReadSharedEnd
        else present [SEL and WRITE and ADB24] then
          run WPAM
        else present [SEL and not WRITE and
                                    ADB24] then
          run RPAM; exit ReadSharedEnd
        else present [SEL and WRITE and not ADB24
                          and ADB23 and ADB22] then
          run WFIFO
        else present [SEL and not WRITE and
                not ADB24 and ADB23 and ADB22] then
          run RFIFO; exit ReadSharedEnd
        else present [SEL and not WRITE and not
            ADB24 and not ADB23 and not ADB22] then
          run RROM; exit ReadSharedEnd
        else present [SEL and WRITE and not ADB24
                      and not ADB23 and ADB22] then
          run WLCA
        else present [SEL and not WRITE and
            not ADB24 and not ADB23 and ADB22] then
          run RLCA; exit ReadSharedEnd
        else
          halt
        end end end end end end end end end

      handle ReadSharedEnd do
        % drive final data word on next cycle
        emit pDriveTBC;
        pause;
        % send RDY and pHostDrives, wait one cycle
        emit RDY;
        emit pHostDrives;
        pause
      end trap
  end await
end loop
```



Optimization: Form product machine because parallel machines actually operate in lock-step.

Optimization: Remove redundant signal emission known never to be "heard."



Optimization: Merge equivalent states because leaving them separate requires substantial, slow decoding logic just before the most critical, complex set of decisions in the machine.

## Experimental Results

| example | lines of code | synthesis method | levels of logic | look-up tables | latches |
|---|---|---|---|---|---|
| graycounter | 91 | V5 + blifopt | 5 | 66 | 27 |
| | | manual | 4 | 51 | 17 |
| abcdef | 142 | V5 + blifopt | 5 | 114 | 25 |
| | | manual | 3 | 128 | 8 |
| mem-ctrl | 80 | V5 + blifopt | 3 | 24 | 16 |
| | | CEC + comb | 3 | 52 | 17 |
| | | CEC + blifopt | 3 | 27 | 15 |
| | | manual | 2 | 31 | 13 |
| | | Original VHDL | 2 | 17 | 11 |
| mem-ctrl2 | 36 | V5 + blifopt | 2 | 17 | 8 |
| | | CEC + comb | 2 | 23 | 9 |
| | | CEC + blifopt | 2 | 18 | 8 |
| | | manual | 2 | 14 | 3 |
| | | JEDI + comb | 2 | 14 | 3 |
| tcint | 689 | V5 + blifopt | 5 | 93 | 52 |
| | | manual | 3 | 118 | 52 |

**Answer:** Effective optimization required both local and global information.

Semi-global reachable state information crucial to get best results.