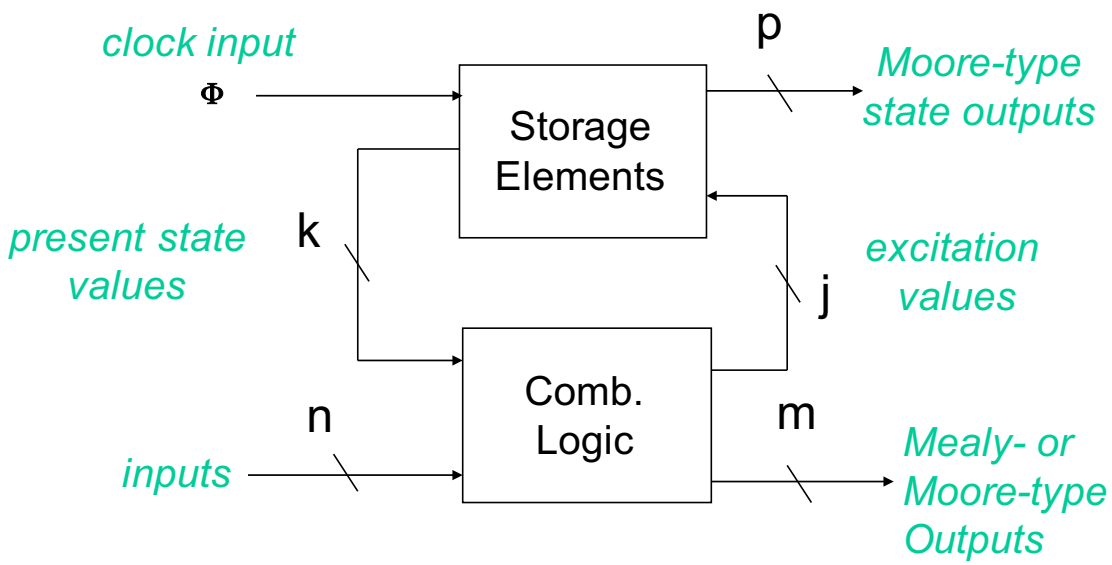
State Assignment (Encoding)



- Encode State Vectors to Enhance Logic Area and Performance

State Encoding Motivation

- FUNCTIONALITY
  - Glitches Can Cause Problems During Transitions
  - Especially when Driving Output Control Signals (Use Synchronous Inputs on Datapath Elements if Possible!!!!)
- AREA
  - Want the Transition Logic to be Minimal in Total Number of Transistors
- SPEED
  - Want Combinational Logic *Maximum Delay Path* to be Small
  - Does Small Logic Path = Small Delay ?????
    NO – NOT NECESSARILY – FAN-IN DEPENDENCE ALSO!!!!
- POWER
  - CMOS is Predominant Implementation Logic Style
    - More Switching Leads to Current Increase
    - Less Switching can lead to Decreased Dynamic Power Dissipation

Example Controller (GLITCH)

```
module contrlr1 (state, ld_xi, iterate, cnt_ld,
                 cnt_en, busy,ld_b, done, clk, reset);
   input        clk, reset, ld_b, done;
   output       ld_xi, iterate, cnt_ld, cnt_en, busy;
   output [1:0] state;
   reg    [1:0] state, pstate, nstate;
   reg          ld_xi, iterate, cnt_ld, cnt_en, busy;

   // State encoding
   parameter S0=2'b00, S1=2'b01, S2=2'b10, S3=2'b11;

   // Register logic
   always @(posedge clk or posedge reset)
     begin
       if (rst == 1'b1) pstate <= S0;
       else pstate <= nstate;
       state <= pstate;
     end
```

Example Architecture – GLITCH

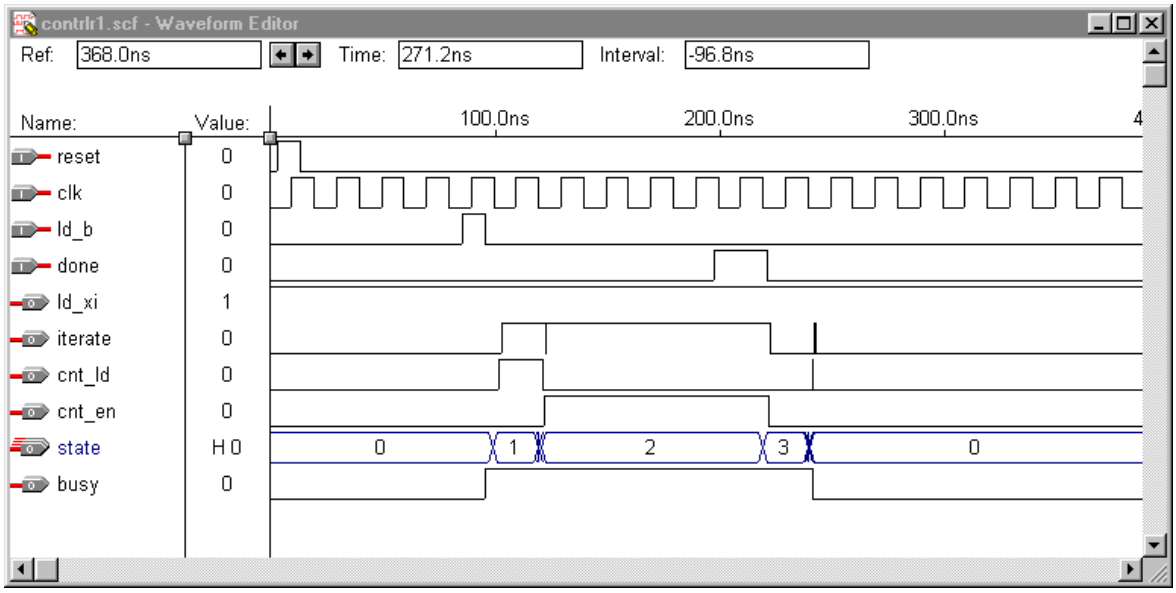
```
     // Transition and output logic
     always @(pstate or ld_b or done)
       begin
         // Default assignments
         iterate = 1'b0;
         cnt_ld = 1'b0;
         busy = 1'b0;
         cnt_en = 1'b0;
         case (pstate)
            S0: if (ld_b==1'b1) nstate = S1;
                else nstate = S0;
            S1: begin
                  iterate = 1'b1;
                  ld_xi = 1'b1;
                  cnt_ld = 1'b1;
                  busy = 1'b1;
                  nstate = S2;
                end
```

Example Architecture – GLITCH

```
        S2: begin
              cnt_en = 1'b1;
              iterate = 1'b1;
              busy = 1'b1
              if (done==1'b1) nstate = S3;
                else nstate = S2;
            end
        S3: begin
              busy = 1'b1;
              nstate = S0;
            end
        default: nstate = S0;
      endcase
    end
endmodule
```

Simulation Waveform (GLITCH)

(*timing simulation result*)



Example Controller (NO GLITCH)

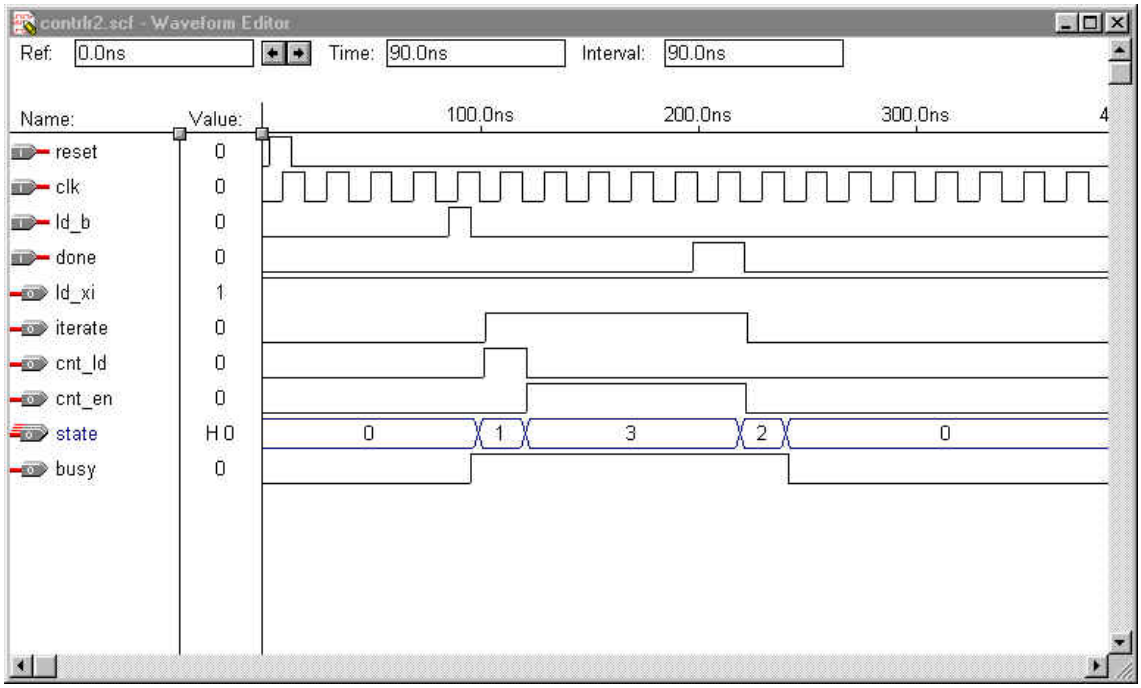
```
module contrlr1 (state, ld_xi, iterate, cnt_ld,
                 cnt_en, busy,ld_b, done, clk, reset);
   input         clk, reset, ld_b, done;
   output        ld_xi, iterate, cnt_ld, cnt_en, busy;
   output [1:0] state;
   reg    [1:0] state, pstate, nstate;
   reg          ld_xi, iterate, cnt_ld, cnt_en, busy;

   // State encoding
   parameter S0=2'b00, S1=2'b01, S2=2'b11, S3=2'b10;

   // Register logic
   always @(posedge clk or posedge reset)
     begin
       if (rst == 1'b1) pstate <= S0;
       else pstate <= nstate;
       state <= pstate;
     end
```

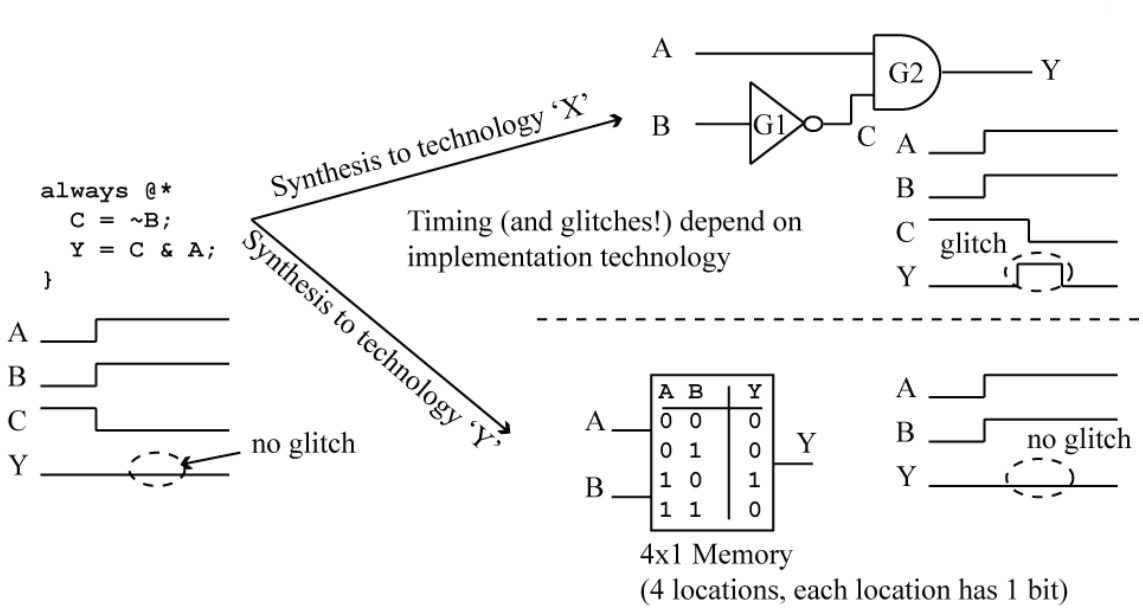
Simulation Waveform (NO GLITCH)

(*timing simulation result*)



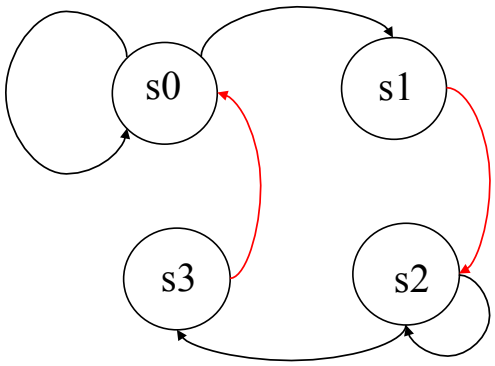
Technology Mapping (GLITCHES)

Previous Example Mapped to Altera EPM7XXX Device



Minimum Bit-Change State Assignment

- Previous Example State Diagram Looked Like:



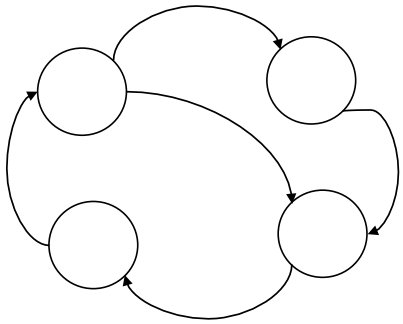
"Glitchy" Encoding:
s0=00, s1=01, s2=10, s3=11

No "Glitch" Encoding:
s0=00, s1=01, s2=11, s3=10

Single bit Change along any transition
- Gray Code is One way to do it

Gray Code

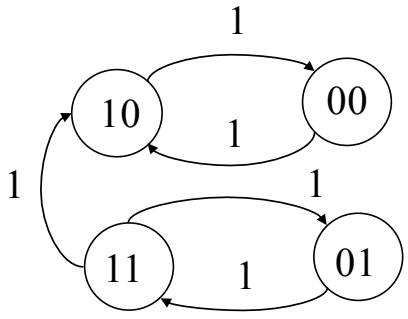
- Used Along Karnaugh Map Edges
  - To Generate:
    00 01 11 10 *REFLECTED GRAY CODE*
  - Reverse Sequence:
    00 01 11 10 10 11 01 00
  - Add Additional 0 to First and 1 to Reversed Sequences:
    000 001 011 010 110 111 101 100
- What About?

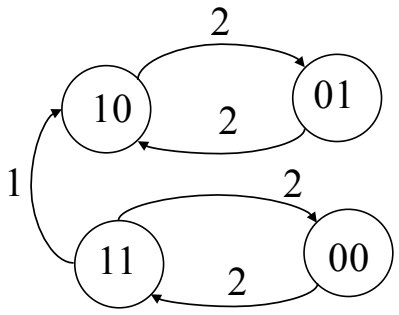


Minimum-Bit Change Code

- Can Assign a "Weight" to Each Transition, W_i

  W_i = # of bit Changes
- Minimize Sum of Weights
- Gray Code Assumes Equal Weight Transitions
- Not Always Possible

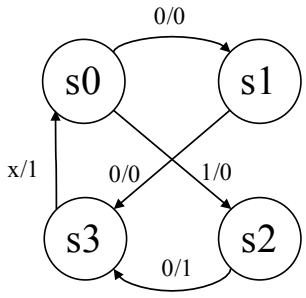


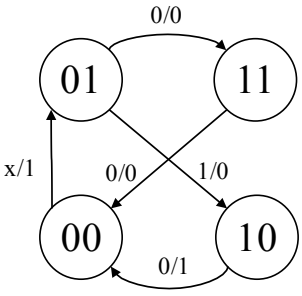


Prioritized Adjacency Strategy

- Assign Adjacent Encodings (differ by 1-bit Only) to:
  1. States With Common Destination
  2. States With Common Source
  3. States With Common Output
- Priority is:
  1. States With Same Next State for a Given Input Value
  2. Destination (next) States With Common Source (present) State
  3. States With Same Output Value for Same Input Values

Prioritized Adjacency Strategy example





Priorities

1. (s1, s2) - Both have same NS (s3) with same input (0)
2. (s1,s2) - Both are NS from Common PS (s0)
3. (s0,s1), (s2,s3) - Both have Same Output 0(1) for Same Input 0(0)

Minimum Length Codes

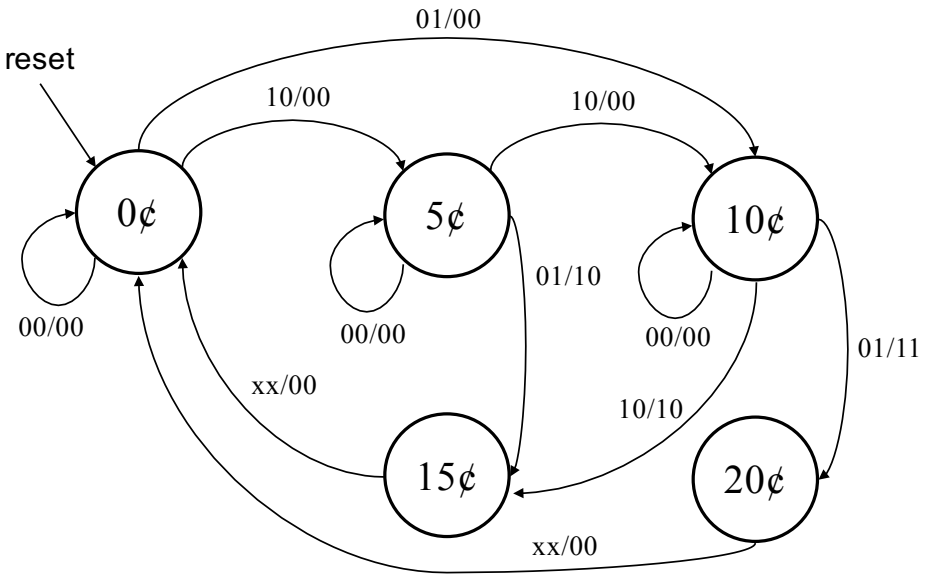
- Smallest Possible Number of Bits per State Vector
  - to represent R states the length must be $k=\lceil \log_2 R \rceil$
- Minimum Length Codes Guarantee Minimal Number of Memory Elements are Used
- N is Number of Possible State Assignments

$$N = \prod_{i=0}^{R-1} \binom{2^k - i}{1} = \prod_{i=0}^{R-1} \left(2^k - i\right) = \left(2^k\right)! - \left(2^k - R\right)!$$

$$\binom{A}{B} = \frac{A!}{(A-B)!B!}$$

State Transition Diagram

Not Simplified!



nd/dc – nickel or dime/dispense product and/or change
product cost is 15¢

**M. Clive, Bebop to Boolean Boogie*

Minimum Length Codes-Example

- Vending Machine Example: $R=5$
  - to represent $R=5$ states the length must be $k=\lceil \log_2 5 \rceil = 3$

$$N = \prod_{i=0}^{R-1} \binom{2^k - i}{1} = \binom{8}{1}\binom{7}{1}\binom{6}{1}\binom{5}{1}\binom{4}{1}$$

$$= \left(\frac{8!}{7!}\right)\left(\frac{7!}{6!}\right)\left(\frac{6!}{5!}\right)\left(\frac{5!}{4!}\right)\left(\frac{4!}{3!}\right) = \left(\frac{8!}{3!}\right)$$

$$= 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 56 \cdot 30 \cdot 4 = 56 \cdot 120 = 6720$$

Minimum Length State Assignment

- Each Possible Assignment can Require Different Logic
- Register/Memory Element Type Also Affects the Logic
- Analyzing all 6720 Assignments for Example for Minimized SOP 2-Level Logic Yields the Following:

Number of Solutions	*Number of Product Terms*
138	7
852	8
1876	9
3094	10
570	11
190	12

TOTAL: 6720

$$\frac{138}{6720} \cong 2.1\%$$

**M. Clive, Bebop to Boolean Boogie*

Minimum Length State Assignment

- Of the 2.1% that Yield 7 Product Terms
  Look at Literal Count

Number of Solutions	*Number of Literals*
66	17
24	18
48	19

TOTAL: 138 $\frac{66}{6720} \cong 1\%$

- Given that Minimum Length State Assignment is Required
- Chances of Optimal Assignment if "Randomly" Assign State Vectors is Less Than 1%!!!!

**M. Clive, Bebop to Boolean Boogie*

Optimum State Assignment for Example

- One of the 66 Optimum (in terms of minimized product and literal count) is:

Symbolic State	*State Vector*
0¢	100
5¢	101
10¢	000
15¢	001
20¢	011

- Brute Force Approach for FPGA Based State Assignment Using Minimum Length Vectors:
  1) Choose Device
  2) Synthesize For all Possible Encodings
  3) Use Analysis tools (timing and area reports)

**M. Clive, Bebop to Boolean Boogie*

State Assignment for Delay Minimization

- Approaches can be Based on Area, Delay, Power Minimization
- "One-hot" is Usual Approach for Speed
  - One Bit in Vector for Each State
  - Only a Single Bit is Set, All Others are Zero-valued
  - Always 2-bit Adjacent State Change
- Can Result in Simple and Fast Transition Logic Networks
- Requires Number of Storage Elements Equal to Number of States
- Usually Not Helpful for Small Number of States in Controller
- Often Used for FPGA Designs Since Operating Speed is Typically Harder Constraint

Example (from Vending Machine Example Earlier)

Symbolic State	*Optimum Min. Length State Vector*	*1-hot EncodedState Vector*
0¢	100	00001
5¢	101	00010
10¢	000	00100
15¢	001	01000
20¢	011	10000

There are Still Multiple Choices for a One-hot Encoding!!

State Assignment (Encoding) Summary

- Encoding of FSM states is an implementation decision
- For K states, need a minimum of $\lceil \log_2(K) \rceil$ DFFs
- Minimal encoding examples for two FF Controller:
  - S0 = 00, S1 = 01, S2 = 10 (Binary-counting order)
  - S0 = 00, S1 = 01, S2 = 11, (Gray code for S0->S1->S2) Gray code usually faster less logic than counting order
  - Gray Code not Always Possible - Try to Use Minimal Adjacent Bit-Change Coding
  - Multiple Minimal Bit-Change Encodings Possible
  - Can Consider Which State Transitions Occur Most Frequently for Minimal Bit-change Encoding Assignment

State Assignment (Encoding) Summary

- One-hot encoding, one FF per state
  - S0 = 001, S1 = 010, S2 = 100
  - For large FSMs (in terms of states), one-hot can be faster than minimal adjacent bit-change encoding
  - One-hot Still has Many Different State Assignments
- Optimal Assignment is an NP-hard Problem
  - Good heuristic Methods have been Developed
  - Public Domain Programs KISS, NOVA, MUSTANG, JEDI
  - Different Methods for Different Criteria: Area, Delay, Power
  - Different Methods for Different Implementation Technology: PLDs, FPGAs, Custom
- Symbolic Encoding Support
  - New Generation HDLs Support User-defined Types (**#define**) Such as SystemVerilog and SystemC
  - Some Synthesis Tools Incorporate Automated State Assignment (Quartus does not have this Feature)