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)

```verilog
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          ld_xi, iterate, cnt_ld, cnt_en, busy;
reg    [1:0] state, pstate, nstate;
// 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

```verilog
// 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
    endcase
```

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          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  \textit{REFLECTED GRAY CODE}
  - Reverse Sequence:
    00 01 11 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

\[
N = \prod_{i=0}^{R-1} \left( \frac{2^k - i}{1} \right) = \prod_{i=0}^{R-1} \left( 2^k - i \right) = \left( 2^k \right)! - \left( 2^k - R \right)!
\]

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

State Transition Diagram

*Not Simplified!

\[nd/dc \text{ – nickel or dime/dispense product and/or change product cost is } 15\phi\]

*M. Clive, Bebop to Boolean Boogie*
Minimum Length Codes-Example

- Vending Machine Example: \( R = 5 \)
  - to represent \( R = 5 \) states the length must be \( k = \lfloor \log_2 5 \rfloor = 3 \)

\[
N = \prod_{i=0}^{R-1} \left( \frac{2^k - i}{1} \right) = \left( \frac{8}{1} \right) \left( \frac{7}{1} \right) \left( \frac{6}{1} \right) \left( \frac{5}{1} \right) \left( \frac{4}{1} \right) = \frac{8!}{3!} \cdot \frac{7!}{6!} \cdot \frac{6!}{5!} \cdot \frac{5!}{4!} \cdot \frac{4!}{3!} = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 56 \cdot 30 \cdot 4 = 56 \cdot 120 = 6720
\]

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:

<table>
<thead>
<tr>
<th>Number of Solutions</th>
<th>Number of Product Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>138</td>
<td>7</td>
</tr>
<tr>
<td>852</td>
<td>8</td>
</tr>
<tr>
<td>1876</td>
<td>9</td>
</tr>
<tr>
<td>3094</td>
<td>10</td>
</tr>
<tr>
<td>570</td>
<td>11</td>
</tr>
<tr>
<td>190</td>
<td>12</td>
</tr>
</tbody>
</table>

TOTAL: 6720
\[
\frac{138}{6720} \approx 2.1\%
\]

*M. Clive, Bebop to Boolean Boogie*
Minimum Length State Assignment

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

<table>
<thead>
<tr>
<th>Number of Solutions</th>
<th>Number of Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>17</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>48</td>
<td>19</td>
</tr>
</tbody>
</table>

TOTAL: 138

\[
\frac{66}{6720} \approx 1\%
\]

• 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:

<table>
<thead>
<tr>
<th>Symbolic State</th>
<th>State Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0¢</td>
<td>100</td>
</tr>
<tr>
<td>5¢</td>
<td>101</td>
</tr>
<tr>
<td>10¢</td>
<td>000</td>
</tr>
<tr>
<td>15¢</td>
<td>001</td>
</tr>
<tr>
<td>20¢</td>
<td>011</td>
</tr>
</tbody>
</table>

• 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)

<table>
<thead>
<tr>
<th>Symbolic State</th>
<th>Optimum Min. Length State Vector</th>
<th>1-hot Encoded State Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0¢</td>
<td>00</td>
<td>00001</td>
</tr>
<tr>
<td>5¢</td>
<td>01</td>
<td>00010</td>
</tr>
<tr>
<td>10¢</td>
<td>00</td>
<td>00100</td>
</tr>
<tr>
<td>15¢</td>
<td>01</td>
<td>01000</td>
</tr>
<tr>
<td>20¢</td>
<td>11</td>
<td>10000</td>
</tr>
</tbody>
</table>

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 \[\log_2(K)\] 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)