# Advanced Digital Logic Design – EECS 303

http://ziyang.eecs.northwestern.edu/eecs303/


Teacher: Robert Dick
Office: L477 Tech
Email: dickrp@northwestern.edu
Phone: 847–467–2298


NORTHWESTERN UNIVERSITY



## FSM design overview

- Specification
- State diagram for FSM
- State table
- State minimization
- State assignment
- Derive state variable and output functions
- Simplify and implement the functions





## Specification

- Sometimes, system specified in way that naturally maps to FSM
- Sometimes, path from specification to FSM is unclear
- Transform the specifications so they can naturally be represented as FSMs
  - E.g., regular expression → NFA → DFA → FSM
- It's fine to go directly to FSM
  - Use transformations when they help you





## Word description to state diagram

- Design a vending machine controller that will release (output signal $r$) an apple as soon as 30¢ have been inserted
- The machine's sensors will clock your controller when an event occurs. The machine accepts only dimes (input signal $d$) and quarters (input signal $q$) and does not give change
- When an apple is removed from the open machine, it indicates this by clocking the controller with an input of $d$
- The sensors use only a single bit to communicate with the controller





## Word description to state diagram

- We can enumerate the inputs on which an apple should be released

$$ddd + ddq + dq + qd + qq$$
$$d(dd + dq + q) + q(d + q)$$
$$d(d(d + q) + q) + q(d + q)$$

For $d$, $i = 0$, for $q$, $i = 1$

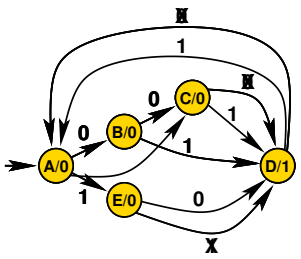
$$0(0(0 + 1) + 1) + 1(0 + 1)$$





## Word description to state diagram

**0(0(0 + 1) + 1) + 1(0 + 1)**








## State diagram to state table

| current state | next state i=0 | next state i=1 | output ($r$) |
|---|---|---|---|
| A | B | E | 0 |
| B | C | D | 0 |
| C | D | D | 0 |
| D | A | A | 1 |
| E | D | D | 0 |





## Implication chart algorithm

1. Construct implication chart, one square for each combination of states taken two at a time.
2. Square labeled $S_i$, $S_j$, if outputs differ than square gets $X$ (0). Otherwise write down implied state pairs for all input combinations.
3. Advance through chart top-to-bottom and left-to-right. If square $S_i$, $S_j$ contains next state pair $S_m$, $S_n$ and that pair labels a square already labeled $X$ (0), then $S_i$, $S_j$ is labeled $X$.
4. Continue executing Step 3 until no new squares are marked with $X$ (0).
5. For each remaining unmarked square $S_i$, $S_j$, then $S_i$ and $S_j$ are equivalent.

## Implication chart minimization

| | | | | |
|---|---|---|---|---|
| B | BC, DE | | | |
| C | BD, DE | CD | | |
| D | 0 | 0 | 0 | |
| E | BD, DE | CD | 1 | 0 |
| | A | B | C | D |

D $\neq$ any other state





## Implication chart minimization

| | | | | |
|---|---|---|---|---|
| B | 0 | | | |
| C | 0 | 0 | | |
| D | 0 | 0 | 0 | |
| E | 0 | 0 | 1 | 0 |
| | A | B | C | D |

Easy – Merge C and E





## Minimization can be more complicated

- Incompletely specified machines are difficult to minimize
- Therefore, some merges can block others
- Need a formulation amenable to *backtracking*





## Incompletely specified Moore state reduction

| | s' | | |
|---|---|---|---|
| s | 0 | 1 | q |
| A | D | B | 0 |
| B | X | A | X |
| C | A | X | 1 |
| D | B | C | 1 |

Consider merging $A$ and $B$ or $B$ and $C$





## State assignment

- Assign values to states
- Keep state variable functions and output variable functions simple
- Allow cubes to be reused among different state variable functions





## State assignment is difficult

Assuming $p$ states and $k$ state variables

$$\frac{(2^k)!}{(2^k-p)!} \text{ possible assignments}$$

Let's simplify that, assume $p$ is an even power of two and

$$k = \lg_2 p$$

then,

$$2^k = p$$





## State assignment is difficult

Therefore, we have

$$\frac{p!}{(p-p)!} = \frac{p!}{0!} = p!/1 = p! \text{ possible assignments}$$

$$p! \in \mathcal{O}\left(2^p\right)$$

- . . . and that's a loose bound
- State assignment has a huge solution space
- It's also a hard problem





## State assignment

- Allow the extraction of common cubes for different state variable functions
  - State variables
    - States connected by transitions should be adjacent
  - Output functions
    - States with equivalent outputs should be adjacent
- Heuristic assignment popular
  - MUSTANG is popular
  - Attraction between states based on ability to extract common cubes

## State assignment

Can do reasonably good state assignment by following guidelines.
Make states have adjacent assignments (differing by only one bit) if:

- They have the same next (child) state in the state diagram for the same input
- They have the same previous (parent) state in the state diagram
- They have the same output for the same input





## State map

- Recall that Karnaugh maps help us visualize adjacency
- Use state maps to visualize state adjacency
- Recall that we have $n$ states, so we require $\lceil \lg_2(4) \rceil$ bit state variables
- $\lceil \lg_2(4) \rceil = 2$
- What if we hadn't done state minimization?
  - Five states → three state variable bits required





## State map

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | A | C | E | G |
| 1 | B | D | F | H |

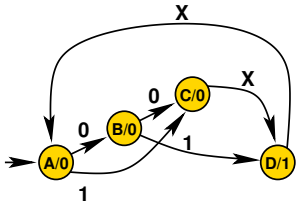
- $a$, $b$, and $c$ are state variable bits
- $A$, $B$, $C$, etc. are states
- State maps help select adjacent assignments for adjacent states





## State assignment

$$0(0(0 + 1) + 1) + 1(0 + 1)$$







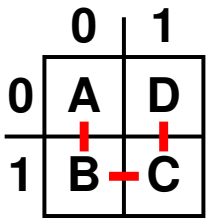

## State assignment guidelines

- States with same child for same input: $\{B, C\}$
- States with same parent: $\{B, C\}, \{C, D\}$
- States with same output: $\{A, B, C\}$
- Prioritize: $\{B, C\}, \{C, D\}, \{A, B, C\}$, etc.





## State assignment








## Symbolic state state table

| current state | next state i=0 | next state i=1 | output ($r$) |
|---|---|---|---|
| A | B | C | 0 |
| B | C | D | 0 |
| C | D | D | 0 |
| D | A | A | 1 |





## State table

| current state ($jk$) | next state ($j^+k^+$) i=0 | next state ($j^+k^+$) i=1 | output ($r$) |
|---|---|---|---|
| 00 | 01 | 11 | 0 |
| 01 | 11 | 10 | 0 |
| 11 | 10 | 10 | 0 |
| 10 | 00 | 00 | 1 |

## Logic function definitions

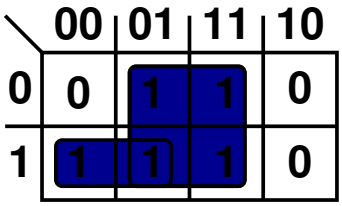
Use Karnaugh maps (or other methods) to simplify functions

- $j^+(j, k, i)$
- $k^+(j, k, i)$
- $r(j, k)$





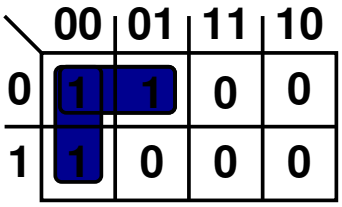
## State variable and output simplification







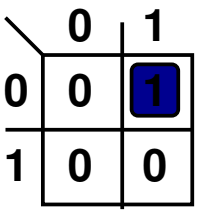

## State variable and output simplification








## State variable and output simplification







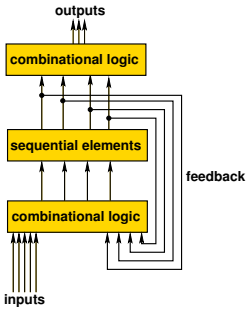

## Implementation

- Implement the state variable functions in combinational logic
- Use sequential elements along feedback paths
- Implement the output variable functions in combinational logic





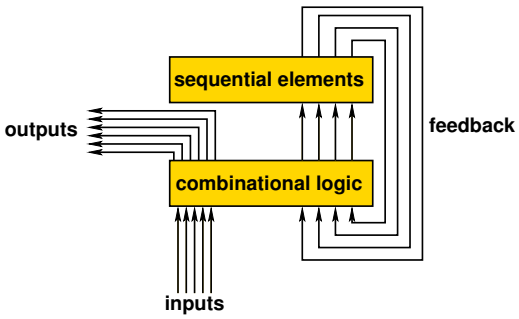
## Moore block diagram








## Mealy block diagram








## State minimization with Don't-Cares

- Can use advanced technique introduced in previous lecture
- Find the maximal compatibles
- Use these to generate the prime compatibles
- Write expression in POS form
- Multiply to get SOP form
- Formulate as a binate covering problem
- This technique is optimal but difficult
- State minimization for incompletely specified machines is a hard problem

## Incompletely specified Moore state reduction

| s | $s^+$ 0 | $s^+$ 1 | q |
|---|---|---|---|
| A | D | B | 0 |
| B | X | A | X |
| C | A | X | 1 |
| D | B | C | 1 |

| B | 1 | | |
|---|---|---|---|
| C | 0 | 1 | |
| D | 0 | 0 | AB |
| | A | B | C |

From Hachtel and Somenzi's Logic Synthesis and Verification Algorithms





## Maximal compatibles

- Choosing the compatible state sets is not straight-forward
- Some combinations block potential future combinations
- Know conflicting states from the compatibility table





## Compatibles and implications

| B | 1 | | |
|---|---|---|---|
| C | 0 | 1 | |
| D | 0 | 0 | AB |
| | A | B | C |

{C, D}, {B, C}, {A, B}





## Class sets

- The combination of a pair of states may require the combination of another state pair
- Thus, using the maximal compatibles is insufficient
- Need additional prime compatibles that have smaller *class sets*
- Starting from the maximal compatibles, which are primes, generate other primes
- In order of decreasing compatible size, if the compatible has a non-empty class set, enter all subsets that are not already contained in other prime compatibles with empty class sets in the table





## Prime compatibles

| B | 1 | | |
|---|---|---|---|
| C | 0 | 1 | |
| D | 0 | 0 | AB |
| | A | B | C |

| A, B | |
|---|---|
| B, C | |
| C, D | AB |
| D | |





## Prime compatible selection

Once the prime compatibles are known, it is necessary to select a subset that

- Has minimal number of selected prime compatibles
- Covers all states
  - Would be unate covering
- Contains all class sets implied by the selected prime compatibles
  - This makes it binate covering
- Recall unate covering, similar solution works
  - Branch and bound





## Binate covering

$$\{C, D\} \rightarrow \{A, B\}$$
$$\overline{\{C, D\}} + \{A, B\}$$

$\{A, B\}$ need to include A
$(\{A, B\} + \{B, C\})$ need to include B
$(\{B, C\} + \{C, D\})$ need to include C
$(\{C, D\} + \{D\})$ need to include D
$(\overline{\{C, D\}} + \{A, B\})$ {C, D} class set requirements





## Binate covering

$$J_1 = \{A, B\}$$
$$J_2 = (\{A, B\} + \{B, C\})$$
$$J_3 = (\{B, C\} + \{C, D\})$$
$$J_4 = (\{C, D\} + \{D\})$$
$$J_5 = (\overline{\{C, D\}} + \{A, B\})$$

| term | prime compatible A, B | B, C | C, D | D |
|---|---|---|---|---|
| $J_1$ | 1 | | | |
| $J_2$ | 1 | 1 | | |
| $J_3$ | | 1 | 1 | |
| $J_4$ | | | 1 | 1 |
| $J_5$ | 1 | | 0 | |

## Binate covering

| term | prime compatible | | | |
|---|---|---|---|---|
| | A, B | B, C | C, D | D |
| $J_1$ | 1 | | | |
| $J_2$ | 1 | 1 | | |
| $J_3$ | | 1 | 1 | |
| $J_4$ | | | 1 | 1 |
| $J_5$ | 1 | | 0 | |

- Find a set of columns, $S$, such that, for every row
  - A 1-column in the row is in $S$ or...
  - ...a 0-column in the row is not in $S$





## Additional examples

| CS | NS(I) | | Out |
|---|---|---|---|
| | 0 | 1 | |
| A | A | C | 0 |
| B | B | B | X |
| C | A | C | 1 |





## Today's Topics

- FSM design example (trying to use more examples)
- State minimization with don't-cares





## General CAD references

If you will be working in digital circuit design, bookmark these sites

- http://www.deepchip.com
  - Designers talk about the current state of CAD and circuit design
- http://www.eetimes.com
  - Electronics design trade journal
  - Won't find current research
  - Will find industry trends





## Recommended reading

- http://www.deepchip.com/gadfly/gad040703.html
- http://www.deepchip.com/gadfly/gad042803.html
- Jayaram Bhasker. *A VHDL Primer*. Prentice-Hall, NJ, 1992
- Chapter 1
- Chapter 2





## Next lecture

- Design representations
- VHDL for combinational and sequential systems