Chapter 6

Logic Design Optimization
Chapter 6

Optimization

- The second part of our design process.
- Optimization criteria:
  - Performance
  - Size
  - Power

Two-level Optimization

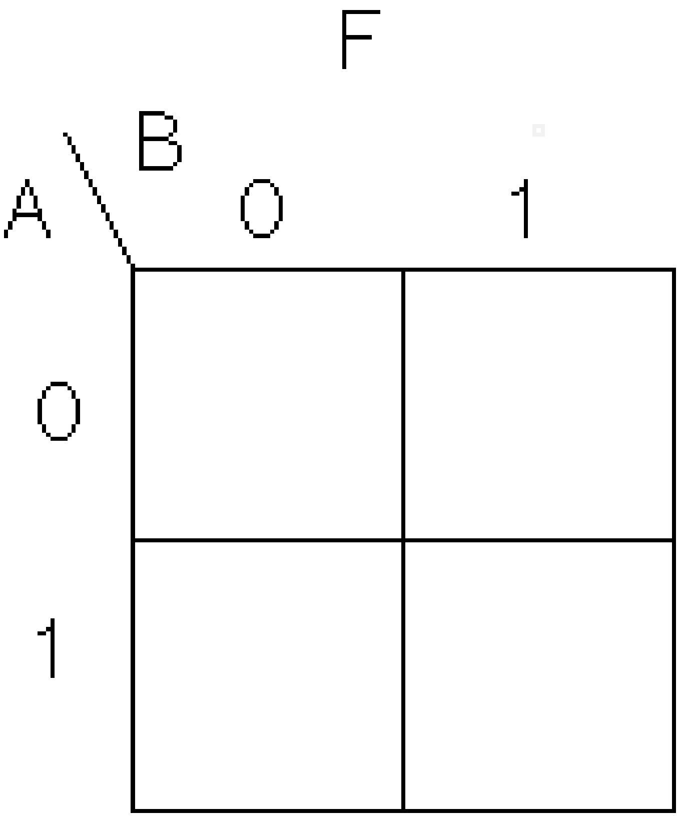
- Manipulating a function until it is in a minimized SOP or POS form.
- Minimizes the number of literals in an equation and results in a smaller circuit than a minimized sum-of-minterms or product of maxterms.
- Use Boolean Algebra
- Use a Karnaugh Map

Karnaugh Maps (K-maps)

- A visual method.
- Good for up to 4 variables. Difficult for 5 or 6 variables. Very difficult for more than 6 variables.
- Start with a truth table for your function.
- A rectangular grid.
- Number of squares is equal to the number of lines in truth table.

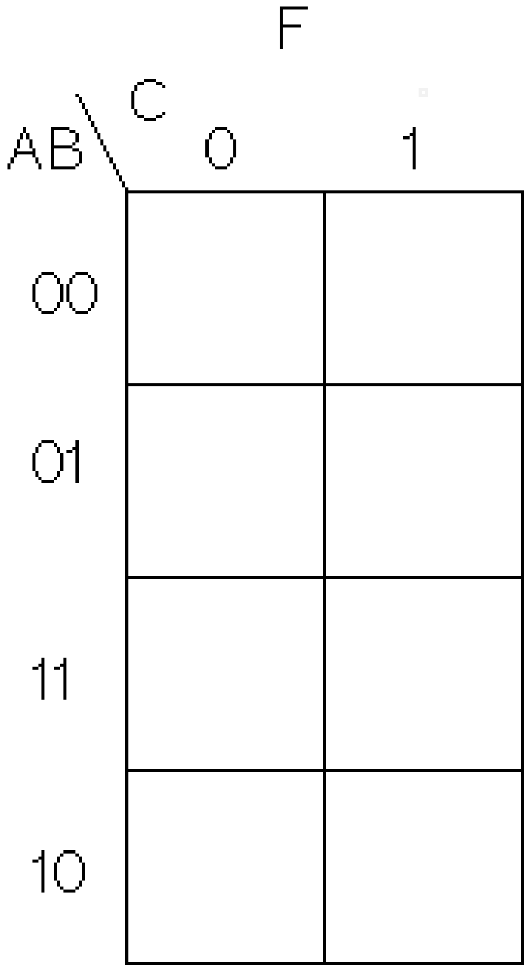
K-Maps

- A two variable K-map.
- Each square is a minterm
- Only one bit may change between adjacent squares.
- Each side of map is a power of 2.



K-Maps

- 3 variable K-map
- Map the equation $\Sigma m(2,3,5,6)$
- Implicant – any rectangular group of 1's where rectangle is a power of 2 on each side.
- Prime Implicant – An implicant that can't “grow larger”.



K-Maps

- Essential Prime Implicant – A prime implicant that has one square that is not part of another prime implicant.
- Non-essential Prime Implicant – A prime implicant where every one of its squares is part of another prime implicant.

K-Maps

- An optimized SOP has all of the essential prime implicants. It may have non-essential prime implicants only if the essential prime implicants don't cover all squares.
- There can be more than one simplified SOP due to the selection of non-essential prime implicants.

K-Maps

- Determine essential and non-essential prime implicants:
  - $\Sigma m(1,2,3,5,8,9,11,15)$
  - $\Sigma m(5,7,9,13,14,15)$
  - $\Sigma m(0,1,3,4,6,9,11,14,15)$
  - $\Sigma m(1,5,13,14,15)$
  - $\Sigma m(2,3,5,7,8,10,12,13)$

K-Maps

- Determine essential and non-essential prime implicants:
  - $\Sigma m(0,1,3,5,6,7,8,9,10,12,14,15)$
  - $\Sigma m(1,2,3,5,6,13,14,15)$

K-Maps

- To create a simplified POS, select rectangular regions of 0's.
- These are called Implicates.
- A simplified POS is made up of essential prime implicates and maybe non-essential prime implicates.
- Map the equation $\Pi M(0,1,4,6)$

K-Maps

- Selecting the 0's as a minimized SOP gives the inverse of the function.
- Use Demorgan's Theorem to get the function and at the same time minimized POS form.
- Usually go straight from K-Map to minimized POS form.

K-Maps

- Determine essential and non-essential prime implicates:
  - $\prod M(1,2,3,5,8,9,11,15)$
  - $\prod M(5,7,9,13,14,15)$
  - $\prod M(0,1,3,4,6,9,11,14,15)$
  - $\prod M(1,5,13,14,15)$
  - $\prod M(2,3,5,7,8,10,12,13)$

K-Maps

- Determine essential and non-essential prime implicates:
  - $\prod M(0,1,3,5,6,7,8,9,10,12,14,15)$
  - $\prod M(1,2,3,5,6,13,14,15)$

K-Maps

- Sometimes you don't care if a minTerm or maxTerm is a 0 or 1. We call these “don't cares”.
- Use them to your advantage to create larger prime implicants or implicates.
  - $\Sigma m(3,5,8,12), d(1,2,6,7,10,14)$

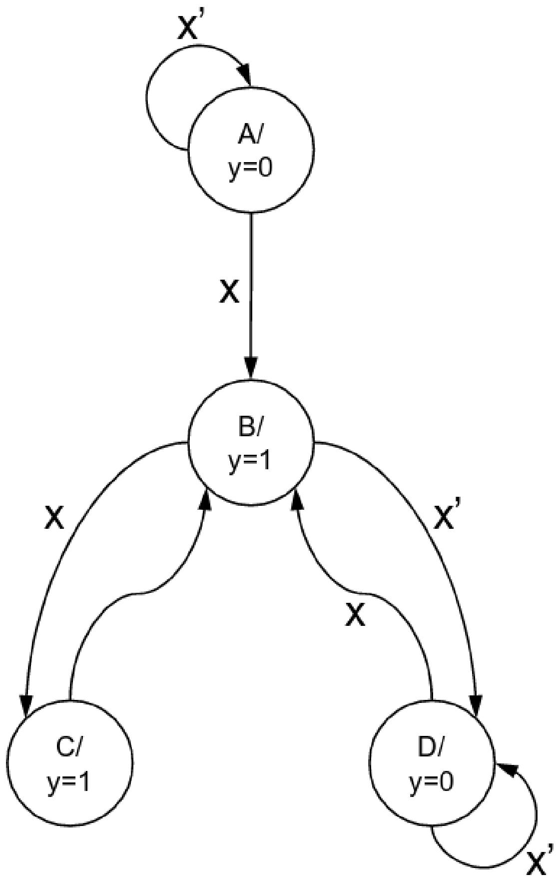
Sequential Logic Optimization

- State reduction
  - A process of reducing the number of FSM states without changing behavior
  - Equivalent states can be removed
  - States are equivalent if
    - Both states assign the same values to outputs
    - For all possible sequences of inputs, FSM outputs will be the same
  - To remove equivalent states:
    - Remove one of the states from the FSM
    - Transfer transitions pointing to the removed state to the other state.

Partitioning method

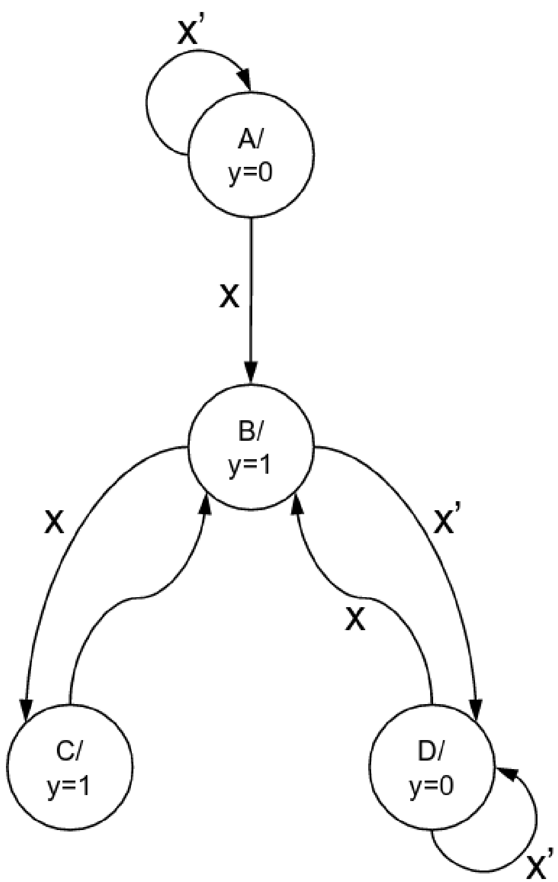
- Partition states into groups based on the values they assign to outputs
- List next state values for each state in a group for all input values.
- Compare states in the group with the same input values
  - If for the same input value, two states transition to states in different groups, they can not be equivalent.
  - Partition states that are not equivalent into sub groups and repeat

Partitioning example



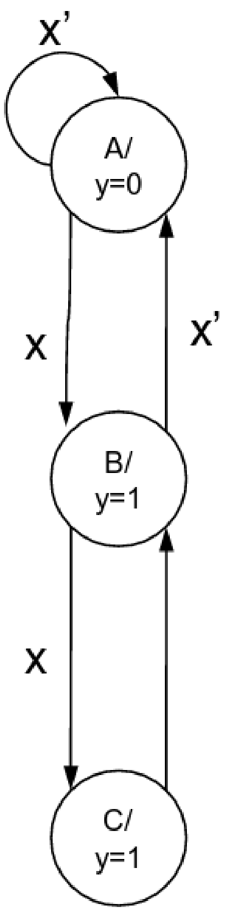
- G1 = {A,D}
  - X = 0
    - A goes to A (G1)
    - D goes to D (G1)
  - X = 1
    - A goes to B (G2)
    - D goes to B (G2)
- G2 = {B,C}
  - X = 0
    - B goes to D (G1)
    - C goes to B (G2)
    - Different
  - X = 1
    - B goes to C (G2)
    - C goes to B (G2)

Partitioning example



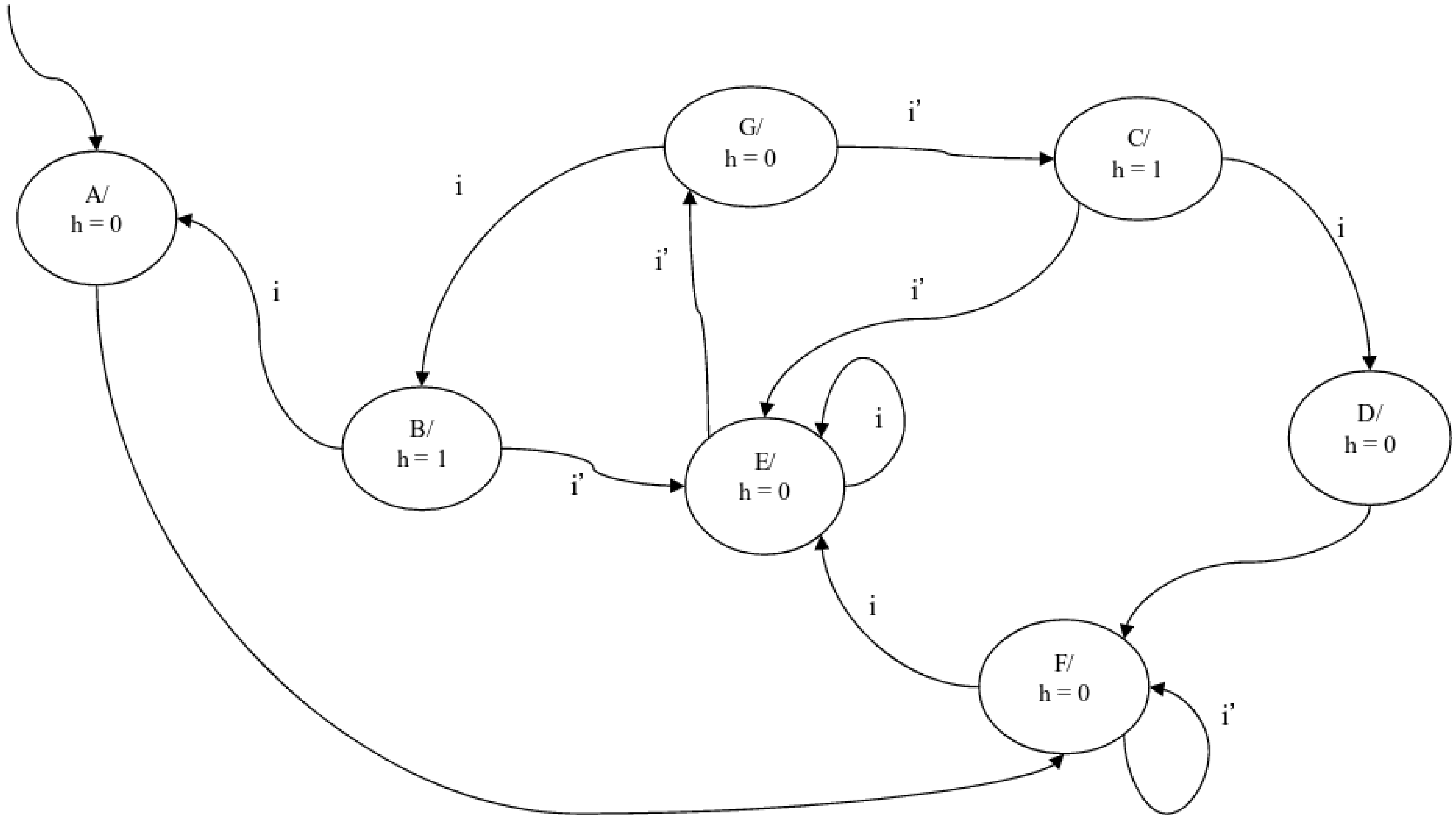
- G1 = {A,D}
  - X = 0
    - A goes to A (G1)
    - D goes to D (G1)
  - X = 1
    - A goes to B (G2)
    - D goes to B (G2)
- G2 = B
  - Only one state
- G3 = C
  - Only one state

Partitioning example



- A and D are equivalent
- Rework the FSM

Partition example



Datapath component tradeoffs

- Faster adders
- Partial full adder (PFA)
  - Break the ripple carry of the Full Adder
  - Instead of carry out it has a generate and propagate carry signal
- Carry lookahead
  - Generates multiple carries
- PFA and Carry Lookahead work together

RTL Design Tradeoffs

- Pipelining
  - Tradeoff between bandwidth and latency
- Concurency
  - Tradeoff between operations per second and size.
- Operator scheduling
  - Tradeoff between operations per second and size.