Two-level Logic Synthesis and Optimization

Dr. Shubhajit Roy Chowdhury,

Centre for VLSI and Embedded Systems Technology,
IIIT Hyderabad, India

Email: src.vlsi@iiit.ac.in

Circuit Optimization

- **Goal: To obtain the simplest implementation for a given function**
- **Optimization is a more formal approach to simplification that is performed using a specific procedure or algorithm**
- **Optimization requires a cost criterion to measure the simplicity of a circuit**
- **Distinct cost criteria we will use:**
  - **Literal cost (L)**
  - **Gate input cost (G)**
  - **Gate input cost including inverters (GN)**

Literal Cost

- **Literal – a variable or it complement**
- **Literal cost – the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram**
- **Example: Boolean expressions for F**
  - **$F = B\,D + A\,\overline{B}\,C + A\,\overline{C}\,\overline{D}$** L = 8
  - **$F = B\,D + A\,\overline{B}\,C + A\,\overline{B}\,\overline{D} + A\,B\,\overline{C}$** L = 11
  - **$F = (A+B)(A+D)(B+C+\overline{D})(\overline{B}+\overline{C}+D)$** L = 10
  - **Which solution is best?**

First solution is best

Gate Input Cost

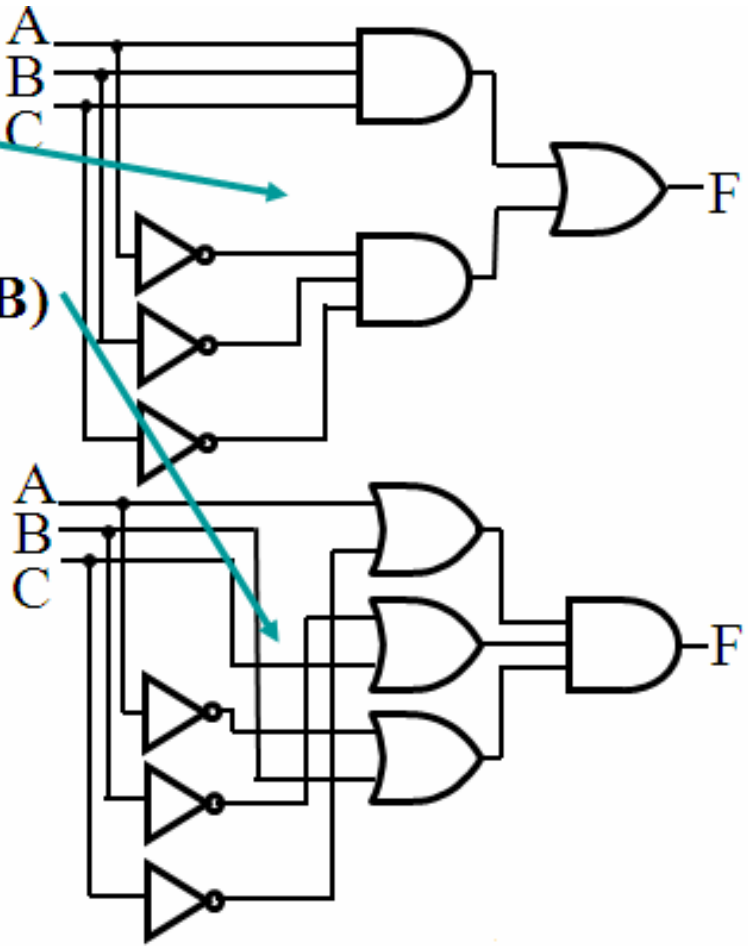
- **Gate Input Cost: Count of total number of inputs to the gates in the logic circuit implementation**
- **Two gate input costs are defined:**

  G = Count of gate inputs without counting Inverters

  GN = Count of gate inputs + count of Inverters
- **For SOP and POS equations, the gate input cost can be found from the Boolean expression by finding the sum of:**
  - **All literal appearances**
  - **Number of terms excluding single literal terms (added to G)**
  - **Number of distinct complemented single literals (added to GN)**
- **Example:**
  - **$F = B\,D + A\,\overline{B}\,C + A\,\overline{C}\,\overline{D}$** L = 8 G = L+3 = 11

Cost Criteria

- Example:
- $F = A\,B\,C + \bar{A}\bar{B}\,\bar{C}$
- L = 6 G = 8 GN = 11
- $F = (A + \bar{C})(\bar{B} + C)(\bar{A} + B)$
- L = 6 G = 9 GN = 12
- Same function and same literal cost
- But first circuit has better gate input count and better gate input count with NOTs
- Select first circuit!



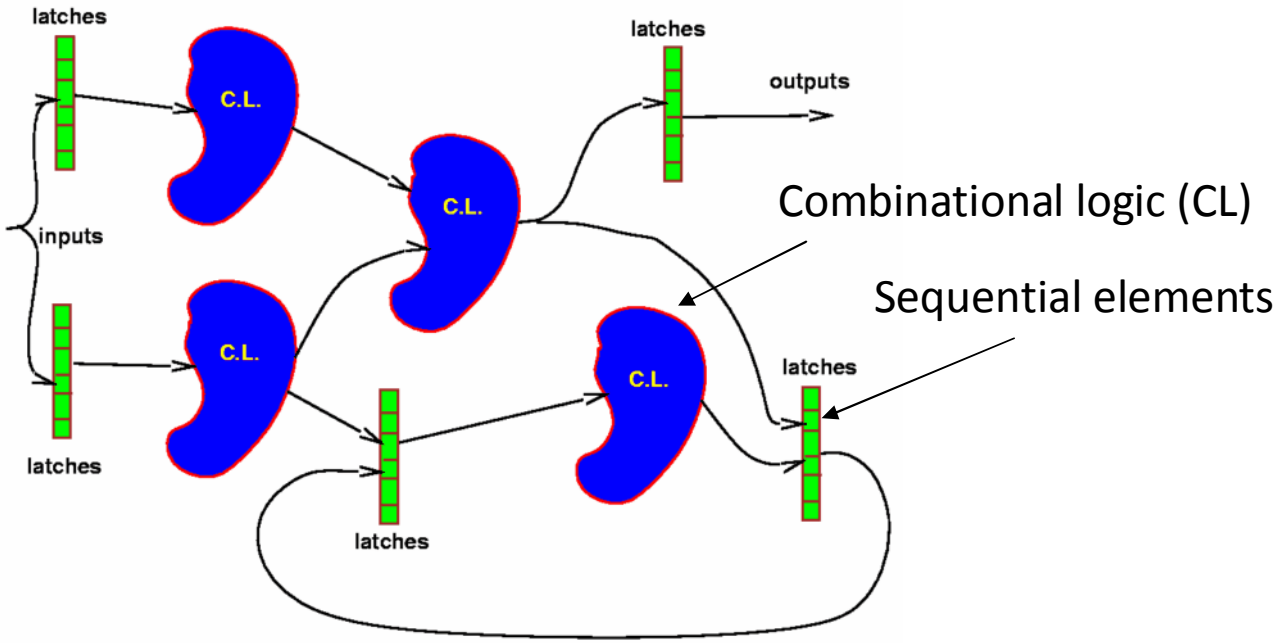
Cost Criteria Summary

- Literal Count:
  - Simple to evaluate by counting all literals
  - However, does not represent circuit complexity accurately in all cases
- Gate Input Cost (or Count):
  - Good measure of logic implementation
  - Proportional to the number of transistors and wires used in the implementation
  - Important when measuring cost of circuits with more than two levels

Boolean Function Optimization

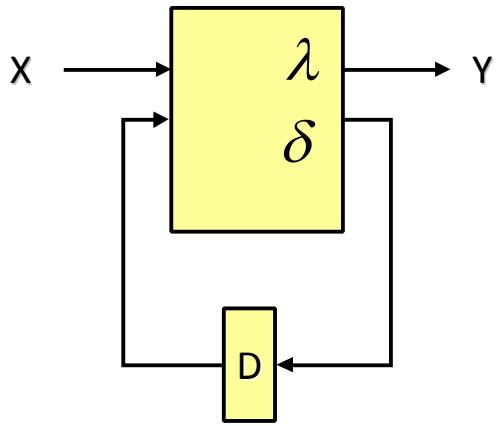
- **Minimizing the gate input (or literal) cost of a Boolean equation reduces the circuit cost**
- **We choose gate input cost**
- **Boolean Algebra and graphical techniques are tools to minimize cost criteria values**
- **Some important questions:**
  - **When do we stop trying to reduce the cost?**
  - **Do we know when we have a minimum cost?**
- **Treat optimum or near-optimum cost functions for two-level (SOP and POS) circuits first**
- **Introduce a graphical technique using Karnaugh maps (K-maps for short)**

General Logic Structure



- Combinational optimization
  - keep latches/registers at current positions, keep their function
  - optimize combinational logic in between
- Sequential optimization
  - change latch position/function (retiming)

What is logic synthesis?



Given: Finite-State Machine F(X,Y,Z, , λ, δ) where:

X: Input alphabet

Y: Output alphabet

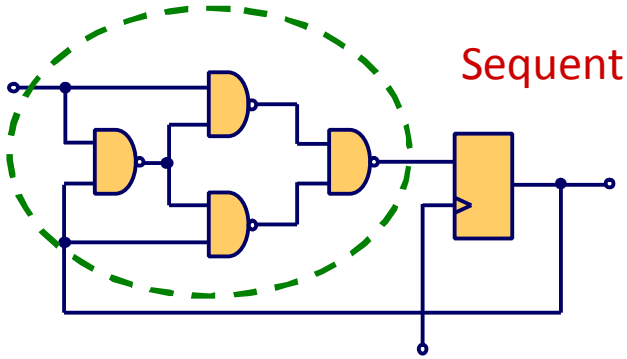
Z: Set of internal states

λ : X x Z $\rightarrow$ Z (next state function, *Boolean*)

δ : X x Z $\rightarrow$ Y (output function, *Boolean*)

Combinational logic

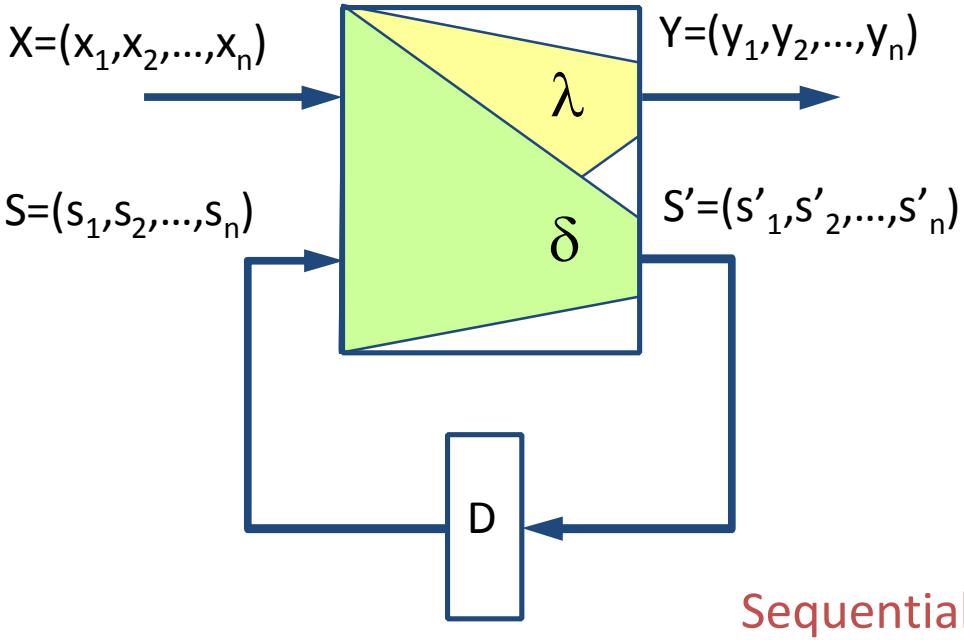
Sequential logic



Target: Circuit *C(G, W)* where:

- *G*: set of circuit components
  {Boolean gates, flip-flops, etc}
- *W*: set of wires connecting *G*

Basic Model of Sequential Circuit: FSM



$M(X,Y,S,S_0,\delta,\lambda)$:

X: Inputs

Y: Outputs

S: Current State

S_0: Initial State(s)

δ: $X \times S \rightarrow S$ (next state function)

λ: $X \times S \rightarrow Y$ (output function)

Sequential synthesis:

find (multi-level) implementation of δ (X) and λ(X) that minimize its cost (area, delay, power)

Delay elements:

- Clocked: synchronous
  - single-phase clock, multiple-phase clocks
- Unclocked: asynchronous

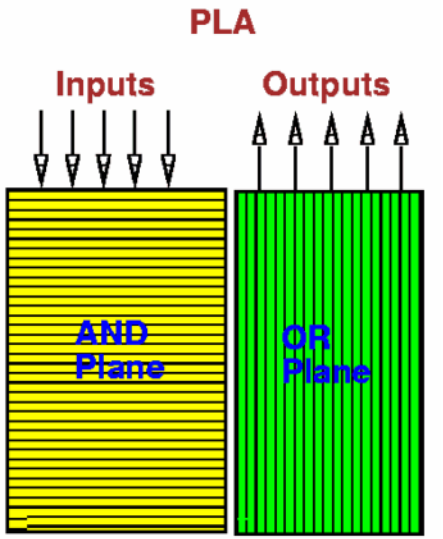
Optimization Criteria for Synthesis

The optimization criteria for logic optimization is to *minimize* some function of:

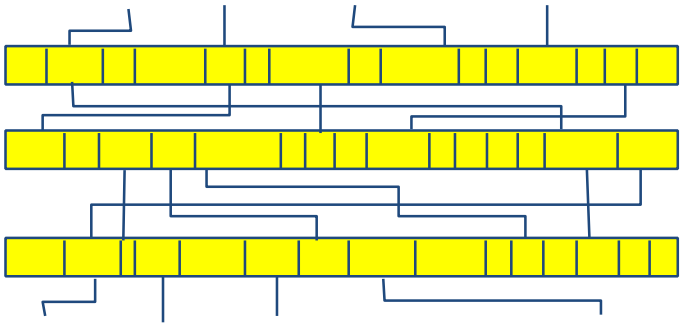
- Area occupied by the logic gates and interconnect (approximated by literals = transistors in technology independent optimization)
- Critical path delay of the longest path through the logic
- Degree of testability of the circuit, measured in terms of the percentage of faults covered by a specified set of test vectors for an approximate fault model (e.g. single or multiple stuck-at faults)
- Power consumed by the logic gates
- Noise Immunity
- Place-ability, Wire-ability

while simultaneously satisfying misc. constraints

Two-Level (PLA) vs. Multi-Level



E.g. Standard Cell Layout



PLA

- control logic
- constrained layout
- highly automatic
- technology independent
- multi-valued logic
- input, output, state encoding

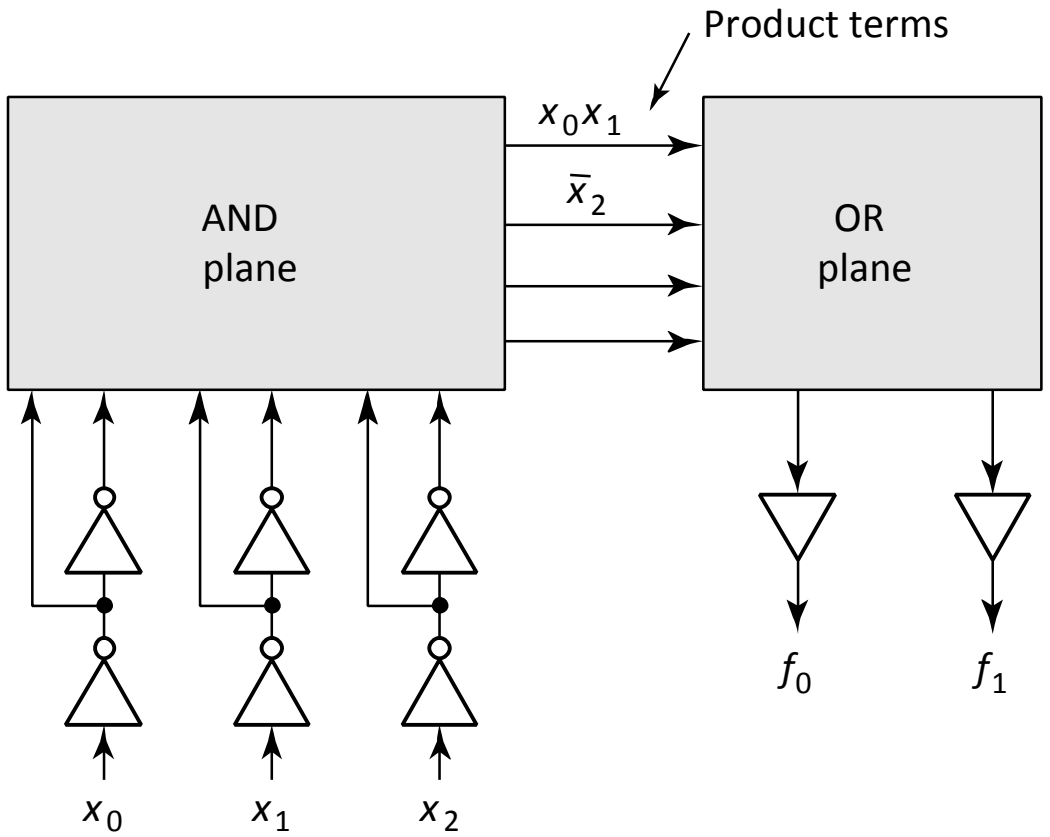
Very predictable

Multi-level Logic

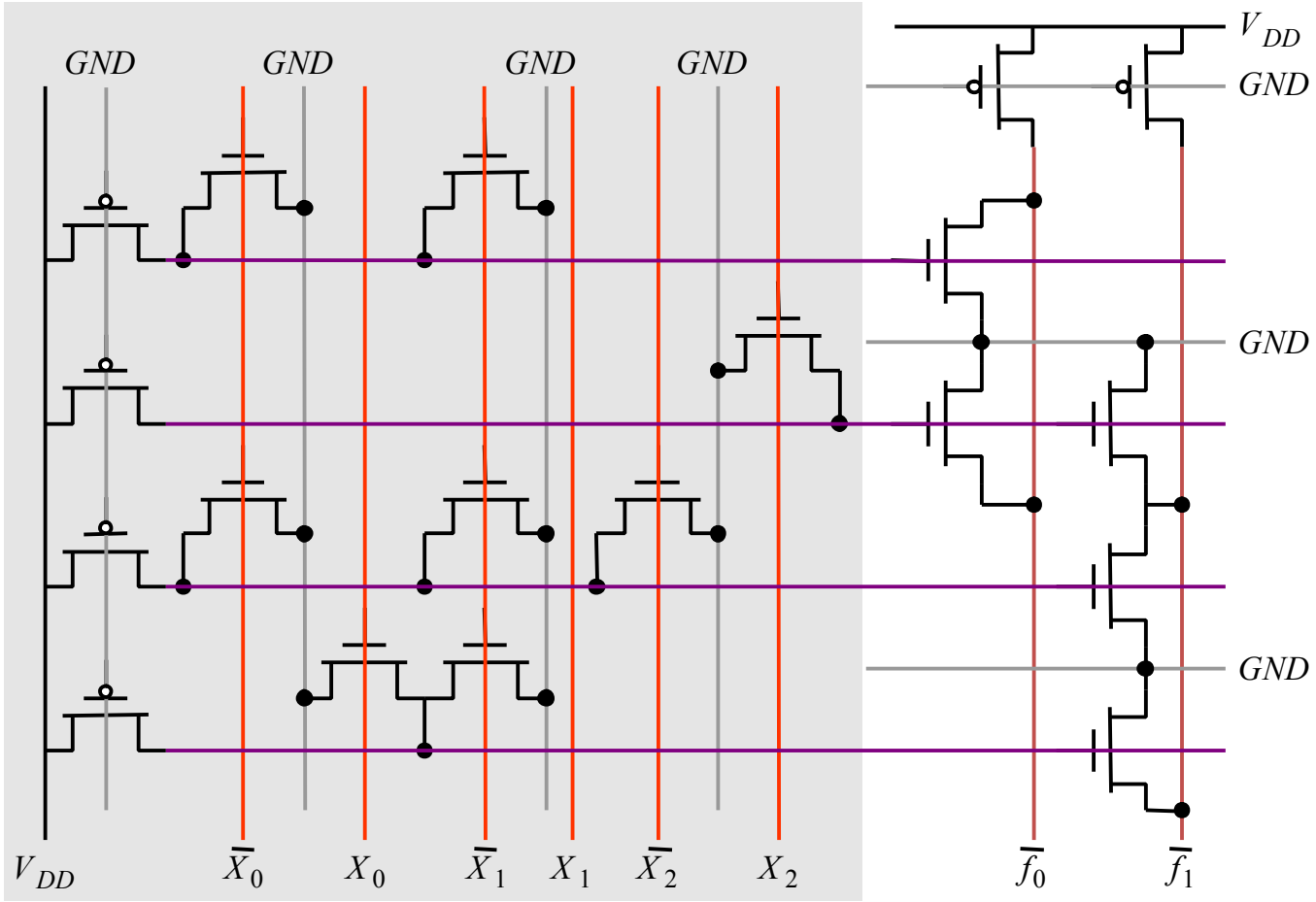
- all logic
- general (standard cells, macro cells, blocks)
- automatic
- partially technology independent
- part of multi-level logic

Very hard to predict

Two-level Logic: the PLA



Programmable Logic Array: Pseudo NMOS PLA



AND-plane **OR-plane**

Multiple-Level Optimization

- **Multiple-level circuits - circuits that are more than two levels (inverters are not counted)**
- **Multiple-level circuits can have reduced gate input cost compared to two-level (SOP and POS) circuits**
- **Multiple-level optimization is performed by applying transformations to circuits represented by equations while evaluating cost**

Transformations

- **Factoring: finding a factored form from SOP or POS expression**
- **Decomposition: expressing a function as a set of new functions**
- **Substitution of H into F: Expressing F as a function of H and some of its original variables**
- **Elimination: Inverse of substitution, called also flattening**
- **Extraction: decomposition applied to multiple functions simultaneously**

Factorization

- **Algebraic Factoring**

  $F = \bar{A}\,\bar{C}\,\bar{D} + \bar{A}\,B\,\bar{C} + A\,B\,C + A\,C\,\bar{D}$ (G = 16)

- **Factoring:**

  $F = \bar{A}\,(\bar{C}\,\bar{D} + B\,\bar{C}) + A\,(B\,C + C\,\bar{D})$ (G = 16)

- **Factoring again:**

  $F = \bar{A}\,\bar{C}\,(B + \bar{D}) + A\,C\,(B + \bar{D})$ (G = 12)

- **Factoring again:**

  $F = (\bar{A}\,\bar{C} + A\,C)\,(B + \bar{D})$ (G = 10)

This factoring example has reduced G from 16 to 10
Resulting circuit has three levels plus input inverters

Decomposition Example

- **Given the following function:**

  F = (A (B + C) + D) (B + C) **(G = 10)**

- **Define 2 new functions X and Y as follows:**

  X = (B + C) and Y = (A X + D)

- **Then function F can be decomposed as follows:**

  F = X Y, where

  X = (B + C) and

  Y = (A X + D) **(G = 8)**

Elimination

- **Given a set of functions:**

  $X = B+C,\ Y = A+B$, and $Z = \overline{A}\,X + C\,Y$ (G = 10)

- **Eliminate X and Y from Z:**

  $Z = \overline{A}\,(B + C) + C\,(A + B)$ (G = 10)

- **Flatten Z (Convert to SOP expression):**

  $Z = \overline{A}\,B + \overline{A}\,C + A\,C + B\,C$ (G = 12)

- **Two-Level Optimization (using K-map):**

  $Z = \overline{A}\,B + C$ (G = 4)

  This example shows that elimination begins with any set of functions. It can increase gate input cost (G) temporarily, but can result in a final solution with optimum cost (G)

Substitution

- **Given the following function:**

  $H = A(\overline{C} + \overline{D})(E + F) + B\,C\,D\,\overline{E}\,\overline{F}$ **(G = 14)**

- **Define X and Y as new functions :**

  $X = C\,D$ **and** $Y = E + F$

- **The complement of X and Y are:**

  $\overline{X} = (\overline{C} + \overline{D})$ **and** $\overline{Y} = \overline{E}\,\overline{F}$

- **Substitute C D with X and** $(\overline{C} + \overline{D})$ **with** $\overline{X}$

  Substitute (E + F) with Y and $\overline{E}\,\overline{F}$ **with** $\overline{Y}$

  $H = A\,\overline{X}\,Y + B\,X\,\overline{Y}$

  where, $X = C\,D$ **and** $Y = E + F$ **(G = 12)**

Extraction

- **Given the following two functions:**

  E = A B D + A B D

  H = B C D + B C D **(G = 16)**

- **Find a common factor for E and H**
- **Define the common factor as a function:**

  F = B D + B D

- **Perform extraction by expressing E and H as a function of F:**

  F = B D + B D, E = A F, H = C F **(G = 10)**

- **Reduced cost because of the sharing of F**

Thank You!!!