CSCI 1010
Models of Computation

Lecture 03
Introduction to Circuits II
Overview

• Normal Form Expansions DNF, CNF and RSE
• Vector operations
• Cyclic shifting
• Logical shifting
• Reductions between binary functions
• Decoder circuit
• Circuit complexity
Review of Normal Form Expansions

• Disjunctive normal form
  – OR of minterms. A minterm of $f : \{0,1\}^n \rightarrow \{0,1\}$ is AND of one literal ($x$ or $\overline{x}$) for each variable of $f$.
  – E.g. $c = \overline{x}yz \lor x\overline{y}z \lor xy\overline{z} \lor xyz$ (the ANDs are implicit)

• Conjunctive normal form
  – AND or maxterms. A maxterm of $f : \{0,1\}^n \rightarrow \{0,1\}$ is OR of one literal ($x$ or $\overline{x}$) for each variable of $f$.
  – E.g. $c = (x \lor y \lor z)(\overline{x} \lor y \lor z)(x \lor \overline{y} \lor z)(x \lor y \lor \overline{z})$
    • $C = 0$ when all inputs 0 and two of them are 0.
DNF and CNF Examples

• DNF Examples

\[ x_1 \oplus x_2 \oplus x_3 = x_1 \bar{x}_2 \bar{x}_3 \lor x_1 x_2 \bar{x}_3 \lor x_1 \bar{x}_2 x_3 \lor x_1 x_2 x_3 \]
\[ x_1 \lor x_2 \lor x_3 = \text{all minterms except } \bar{x}_1 \bar{x}_2 \bar{x}_3 \]
How many minterms are needed for \( x_1 \oplus x_2 \oplus ... \oplus x_n \)?

• CNF Examples

\[ x_1 \oplus x_2 \oplus x_3 = (\bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_3) \land (\bar{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \bar{x}_2 \lor x_3) \land (x_1 \lor x_2 \lor \bar{x}_3) \]
\[ x_1 x_2 x_3 = \text{all maxterms except } (x_1 \lor x_2 \lor x_3) \]
How many maxterms are needed for \( x_1 \oplus x_2 \oplus ... \oplus x_n \)?
Ring Sum Normal Form Expansion

- **Ring sum normal form (RSE)** is the sum (Exclusive OR) of 1 or 0 and products (AND) of un-negated variables.
  - E.g. \( f(x,y,z) = 1 \oplus x \oplus xy \oplus yz \)

- To produce RSE from DNF expand the minterms using the identity \( \overline{x} = (1 \oplus x) \)
  - E.g. \( x\bar{y}z = x(1 \oplus y)z = xz \oplus xyz \)
Size of Normal Form Representations

• In DNF n-input vector XOR $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ requires $2^{n-1}$ minterms.
  – Each minterm uses (n-1) 2-input ANDs and $\leq n$ NOTs.
• In RSE normal form it only requires n-1 XORs, $\oplus$!
• In RSE vector OR, $x_1 \lor x_2 \lor \ldots \lor x_n$, requires all products (ANDs) in 1 or more variables
  – E.g. $x_1 \lor x_2 \lor x_3 \lor x_4 = x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3$
  – Thus, the RSE of $x_1 \lor x_2 \lor \ldots \lor x_n$ has $2^{n-1}$ products!
Other Vector Operations

\[ \mathbf{u} = (u_1, u_2, ..., u_n), \quad \mathbf{v} = (v_1, v_2, ..., v_n) \]

**AND**

\[ \mathbf{u} \land \mathbf{v} = (u_1 \land v_1, u_2 \land v_2, ..., u_n \land v_n) \]

**OR**

\[ \mathbf{u} \lor \mathbf{v} = (u_1 \lor v_1, u_2 \lor v_2, ..., u_n \lor v_n) \]

**NOT**

\[ \neg \mathbf{u} = (\bar{u}_1, \bar{u}_2, ..., \bar{u}_n) \]

**EQUAL**

\[ \mathbf{u} \star \mathbf{v} = 1 \text{ if } u_j = v_j \text{ for all } 1 \leq j \leq n, \text{ else } = 0 \]

\[ \land: \{0,1\}^{2n} \rightarrow \{0,1\}^n, \quad \lor: \{0,1\}^{2n} \rightarrow \{0,1\}^n, \]

\[ \neg: \{0,1\}^n \rightarrow \{0,1\}^n, \quad \star: \{0,1\}^{2n} \rightarrow \{0,1\} \]
Some Notation

• Binary number system:
  \[ s = (s_{k-1}, s_{k-2}, \ldots, s_0) = s_{k-1} s_{k-2} \ldots s_0 \]  represents the integer  \[ |s| = s_{k-1} 2^{k-1} + s_{k-2} 2^{k-2} + \ldots + s_0 2^0 \]

• Example:
  \[ s = 101_2 \] represents  \[ 5_{10} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \]

• Modular arithmetic
  \[ x \ mod \ n \] is remainder after division by n.
  \[ 0 \ mod \ 3 = 0, \ 1 \ mod \ 3 = 1, \ 2 \ mod \ 3 = 2, \]
  \[ 3 \ mod \ 3 = 0, \ 4 \ mod \ 3 = 1, \ 5 \ mod \ 3 = 3, \text{ etc.} \]
Cyclic Shifting

cyclic shift left 3 places

\[ s = 011_2, \ |s| = 3_{10} \]

- **Inputs** \( u = (u_{n-1}, u_{n-2}, ..., u_0), \ s = (s_{k-1}, s_{k-2}, ..., s_0) \)
- **Output** \( v = (v_{n-1}, v_{n-2}, ..., v_0) \) where \( v \) is \( u \) shifted left cyclically by \( |s| \) positions where
  \[ |s| = s_{k-1}2^{k-1} + s_{k-2}2^{k-1} + s_12^1 + s_0. \]
  Thus,
  \[ v_j = u_{(j-|s|) \mod n} \text{ for } 0 \leq j \leq n-1. \]
- \( f_{cyclic} : \{0,1\}^{n+k} \rightarrow \{0,1\}^n, \ v = f_{cyclic}(u, s) \)
Cyclic Shifting Algorithm

- Since $|s| = s_{k-1}2^{k-1} + ... + s_12^1 + s_0$, for $0 \leq j \leq n-1$ we can realize $f_{cyclic}(u, s)$ by cyclic shifting $u$ by 0 positions when $s_j = 0$ or $2^j$ positions when $s_j = 1$. 
Cyclic Shifting Implementation

• One step of cyclic shifting

• **Theorem** \( f_{cyclic} : \{0,1\}^{n+k} \rightarrow \{0,1\}^n \) can be realized by a circuit with \( (3n+1)k \) gates and depth \( 2k+1 \) where \( k = \lceil \log_2 n \rceil \).
Logical Shifting

- Logical shifting function $f_{\text{logic}} : \{0,1\}^{n+k} \rightarrow \{0,1\}^n$ drops high order bits and inserts 0s.
Reductions

- Can cyclic shift (A) be reduced to logical shift (B)?
  - That is, can we use a logical shift function to compute the cyclic shift function?
- Can logical shift (A) be reduced to cyclic shift (B)?
  - That is, can we use a cyclic shift function to compute the logical shift function.
- Reduction: Let functions A and B be on \( n \) and \( m \) inputs. Can we assemble the inputs to B from those of A and process B’s outputs quickly so that outputs of A can be extracted?
Shift Reductions

• Can we do **logical shifting** using **cyclic shifting**?
  – How do we prepare inputs of logical shifter to compute the results of cyclic shifter?
  – Can we repeat the inputs?
  – Can we ignore some outputs?

• Can we do **cyclic shifting** using **logic shifting**?
  – Will a similar approach work here?
Reducing Multiplication to Squaring

• If we don’t have a multiplier, we can multiply x and y, where \(|x|, |y| \leq 2^{n-1}\) and \(n \geq 2\) by squaring \(z\) where \(z=x+cy\), \(c = 2^{2n-1}\), a constant, when \(x, y,\) and \(z\) are represented in binary.
  – \(z^2 = x^2+2cxy+(cy)^2\). Thus, \(x^2 < 2cxy < (cy)^2\) if \(x,y \geq 1\).
  – Thus, the bits of \(x^2, 2cxy,\) and \((cy)^2\) don’t overlap.
  – \(xy\) can found in bit positions \(2n, 2n+1, \ldots, 3n\) of \(z^2\).
  – Algorithm to create inputs, extract outputs is fast!

• If multiplication were hard, so must be squaring!
Circuit Size Complexity

• **Definition:** The circuit complexity of a binary function \( f : \{0,1\}^n \rightarrow \{0,1\}^m \) over the basis \( \Omega \), denoted \( C_\Omega(f) \), is the size of the smallest circuit for \( f \) using functions from \( \Omega \).

  – For \( f_{cyclic} : \{0,1\}^{n+k} \rightarrow \{0,1\}^n \),
    \[
    C_\Omega(f_{cyclic}) \leq (3n+1)k, \quad k = \lceil \log_2 n \rceil
    \]
Circuit Depth Complexity

• **Definition:** The **depth complexity** of a binary function \( f : \{0,1\}^n \rightarrow \{0,1\}^m \), denoted \( D_\Omega(f) \), is the length of the longest path in a circuit for \( f \) over \( \Omega \) for which the longest path is shortest.

  – For \( f_{\text{cyclic}} : \{0,1\}^{n+k} \rightarrow \{0,1\}^n \), \( D_\Omega(f_{\text{cyclic}}) \leq 2k+1 \)
Decoder Circuits

- Decoder has \( n \) binary inputs, \( 2^n \) binary outputs.
- One output has value 1 and the rest are 0.
  - \( y_j(x_3, x_2, x_1, x_0) = 1 \) only if \((x_3, x_2, x_1, x_0)\) is \( j \) in binary.

E.g. \( y_5(x_3, x_2, x_1, x_0) = 1 \) only when \((x_3, x_2, x_1, x_0) = (0, 1, 0, 1)\)
Decoder Circuits

• Decoder function $f_{dec}^{(n)} : \{0,1\}^n \rightarrow \{0,1\}^m$, $m=2^n$.

• Let $y = f_{dec}^{(n)} (x)$.
  – Then, $y_j = 1$, when $j = |x|$, and $y_j = 0$, otherwise.

• In general, $y_{|x|} = |x|$th minterm on $x$.
  – E.g. for $n = 2$, $y_3 = x_1 x_0$, $y_2 = x_1 \overline{x}_0$, $y_1 = \overline{x}_1 x_0$, $y_0 = \overline{x}_1 \overline{x}_0$
  – Because each minterm uses $\leq (2n-1)$ ANDs and NOTs, $C_\Omega(f) \leq (2n-1) \cdot 2^n$.

• Since each output function is different, $C_\Omega(f) \geq 2^n$.
  • Note: Good upper & lower bounds help determine circuit size.
Big Oh Notation for Functions

• Let $N = \{0,1,2,3,4, \ldots \}$. Given functions $f : N \to N$, $g : N \to N$, the values of $f$ and $g$ on input $n$ are $f(n)$ and $g(n)$.

• **Definition** $f(n) = O(g(n))$ if there exist $K > 0$ and $n_0$ in $N$ s.t. for $n \geq n_0$, $f(n) \leq K \cdot g(n)$. 
Efficient Decoder Circuit

- A minterm on $n$ inputs is AND of two minterms on $n/2$ inputs, e.g. $x_3 \overline{x_2} x_1 x_0 = (x_3 \overline{x_2})(x_1 x_0)$

$$C_\Omega(f_{dec}^{(n)}) \leq 2^n + 2 C_\Omega(f_{dec}^{(n/2)}) \leq 2^n + 4(n-1) 2^{n/2} = O(2^n)$$
Construction of Other Circuits

• See Chapter 2 of book for efficient circuits for
  – Encoder – one of $2^n$ inputs is 1, output is its index
  – Prefix comp. – $f(x_1, \ldots, x_n) = (x_1, x_1 \odot x_2, \ldots, x_1 \odot x_2 \ldots \odot x_n)$
    $\odot$ is associative. Used in addition, parallel computation
  – Binary arithmetic functions
  – Symmetric functions (counting, e.g.)
  – Multiplexer – select one of $2^n$ inputs
  – Demultiplexer – move one input to one of $2^n$ outputs
Challenge Assignment

• Show that $f(x_1, x_2, ..., x_n) = x_1 \land x_2 \land ... \land x_n$ requires at least $n-1$ two-input ANDs and has depth at least $\lceil \log_2 n \rceil$.

• That is, $C_\Omega(f) \geq n-1$ and $D_\Omega(f) \geq \lceil \log_2 n \rceil$. 
Upper Bounds to Circuit Complexity

• For the AND of $n$ inputs, the following upper bounds apply:
  – $C_\Omega(f) \leq n-1$. Thus, $C_\Omega(f) = n-1$
  – $D_\Omega(f) \leq \lceil \log_2 n \rceil$. Thus, $D_\Omega(f) = \lceil \log_2 n \rceil$.

• What upper bounds apply to arbitrary Boolean functions?
General Upper Bounds

• What about $C_{\Omega}(f)$ and $D_{\Omega}(f)$ for arbitrary $f: \{0,1\}^n \rightarrow \{0,1\}$?

• Use DNF to construct a logic circuit.
  – $f$ has $2^n$ minterms; they are generated by the decoder function $f_{dec}^{(n)}$ where $C_{\Omega}(f_{dec}^{(n)}) = O(2^n)$
  – Since $f$‘s minterms can be combined using at most $2^n-1$ ORs, $C_{\Omega}(f) = O(2^n)$

• This bound can be improved to $C_{\Omega}(f) = O(2^n/n)$
General Lower Bound

• When $0 < \epsilon < 1$ and $n$ is large enough at least a fraction $1 - 2^{-d}$ of functions $f: \{0,1\}^n \rightarrow \{0,1\}$, with $d = \epsilon 2^{n-1}$, $\epsilon > 0$ small, have

$$C_\Omega(f) \geq \left(\frac{2^n}{n}\right)(1-\epsilon) - 2n^2.$$ 

• Most $n$-input Boolean functions are complex and have $C_\Omega(f) \approx \frac{2^n}{n}$
Review

- Normal Form Expansions DNF, CNF and RSE
- Vector operations
- Cyclic shifting
- Logical shifting
- Reductions between binary functions
- Decoder circuit
- Circuit complexity