CNS 188a Overview

- Boolean algebra as an **axiomatic system**
- Boolean functions and their **representations using** Boolean formulas and **spectral methods**
- **Implementing** Boolean functions with relay circuits, circuits of **AON (AND, OR, NOT) gates** and **LT (Linear Threshold) gates**
- Analyzing the **complexity (size and depth)** of circuits
- **Relations** (as opposed to functions) and their implementation in circuits
- **Feedback** and convergence in LT circuits
**LT Constructions for Symmetric Functions**

<table>
<thead>
<tr>
<th>circuit kind</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LT-l</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>LT-nl, d-2</td>
<td>$\left\lfloor \frac{n}{2} \right\rfloor + 1$</td>
</tr>
<tr>
<td>LT-nl -XOR</td>
<td>$\lceil \log_2(n + 1) \rceil$</td>
</tr>
</tbody>
</table>

**Binary representation**

- Kautz

**Transitions**

- Muroga

**Intervals**

- Minnick

**Optimal**
Muroga's Idea

\[ EQ(j, k) = \begin{cases} 
1 & \text{if } j \leq |X| \leq k \\
0 & \text{else}
\end{cases} \]

\[ EQ(j, k) = TH_j + \overline{TH}_{k+1} - 1 \]
## LT-1 Circuit Design Algorithm for SYM

### Muroga 1959

\[
f(X) = \overline{TH_2} + TH_3 + \overline{TH_5} + TH_6 - 1
\]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( f(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

- Subtract 1 for every isolated 1-block
- \( 0 \Rightarrow 1 \)
- \( 1 \Rightarrow 0 \)
Given a symmetric function $f$ with $k$ 1s in the symmetric function table

At locations: $q_1, q_2, q_3, \ldots, q_k$

Construct an LT-nl circuit with $k+1$ gates

$k = 5$

$\begin{align*}
q_1 &= 1 \\
q_2 &= 3 \\
q_3 &= 4 \\
q_4 &= 6 \\
q_5 &= 7
\end{align*}$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$f(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
LT-nl Construction
Minnick 1959

\[ |X| TH_{(q_1+1)} |X| \]

\[ |X| TH_{(q_2+1)} |X| \]

\[ |X| TH_{(q_k+1)} |X| \]

\[ |X| \]
The Gate $M$

From the first layer of $k$ TH gates

$$|X|$$

\[-(q_2-q_1)\]
\[-(q_3-q_2)\]
\[-(q_k-q_{k-1})\]
\[-(n+1-q_k)\]

\[-q_1\]

0 if $q_k = n$
\[ F(X) = -\left( q_1 + \sum_{i=1}^{k-1} a_i (q_{i+1} - q_i) + a_k (n + 1 - q_k) \right) + |X| \]

**Q:** how do the \( a_i \)'s look?

- \( a_1 \)
- \( -\left( q_2 - q_1 \right) \)
- \( q_1 \)

- \( a_2 \)
- \( -\left( q_3 - q_2 \right) \)

- \( a_k \)
- \( -\left( q_k - q_{k-1} \right) \)

- \( -\left( n + 1 - q_k \right) \)

0 if \( q_k = n \)

**Conditions:**
- \( |X| \leq q_1 \)
- \( q_1 < |X| \leq q_2 \)
- \( q_2 < |X| \leq q_3 \)
What was Minnick’s key idea?

\[
|X| = (q_1 (q_2 + 1) + a_i (q_i + 1 - b_i) + 1) + (q_2 - 1)
\]
The Key: Telescopic Sum

\[ q_4 < |X| \leq q_5 \]

\[
\begin{array}{cccc}
\text{a}_1 &= 1 & 1 & 1 \\
\text{a}_2 &= 1 & 1 & 1 \\
\text{a}_3 &= 1 & 1 & 0 \\
\text{a}_4 &= 1 & 0 & 0 \\
\text{a}_5 &= 0 & 0 & * \\
\end{array}
\]

\[ F(X) = -q_5 + |X| \]

We are always left with the last element in the sum.

\[
F(X) = -q_1 - \sum_{i=1}^{k-1} a_i (q_{i+1} - q_i) + a_k (n + 1 - q_k) + |X|
\]
**The Minnick Construction**

**Claim:** The Minnick construction is correct

**Proof:**

\[
F(X) = - \left( q_1 + \sum_{i=1}^{k-1} a_i (q_{i+1} - q_i) + a_k (n + 1 - q_k) \right) + |X|
\]

Consider \( |X| \) in each possible range:

- \( 0 \leq |X| \leq q_1 \) \[ F(X) = -q_1 + |X| \]
- \( q_1 < |X| \leq q_2 \) \[ F(X) = -q_2 + |X| \]
- \( q_2 < |X| \leq q_3 \) \[ F(X) = -q_3 + |X| \]
- \( q_3 < |X| \leq q_4 \) \[ F(X) = -q_4 + |X| \]
- \[ \vdots \]
- \( q_{k-1} < |X| \leq q_k \) \[ F(X) = -q_k + |X| \]

**Telescopic sum**
Example of Minnick's Construction

<table>
<thead>
<tr>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X$</th>
<th>$f(X)$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$-3a_1-a_2-1$</th>
<th>$\ldots + \vert X \vert$</th>
<th>$sgn(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-4</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-4</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The Kautz Construction for XOR
1961

Implement the algorithm for computing the binary representation using LT gates

Algorithm:  \( A \leftarrow m \)

For \( i = j - 1 \) to 0 do:
\[
\begin{cases} 
  \text{if } A \geq 2^i \text{ then } a_i = 1 \\
  \text{else } a_i = 0 \\
  A \leftarrow A - a_i 2^i
\end{cases}
\]
Questions on LT Constructions

Q1: We understand how to implement a single symmetric function, what if we need to implement a circuit that computes a number of symmetric functions? Can we save in circuit size?

A1: YES - we can ‘reuse’ the first layer! By putting all required TH gates.
# Questions on LT Constructions

Q2: Why do we care about **symmetric** functions? What about **general** Boolean functions?

A2: A general Boolean function can be treated as a symmetric function by introducing linear ordering.

<table>
<thead>
<tr>
<th>$x_1x_2x_3$</th>
<th>$f(x_1, x_2, x_3)$</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LT-l</th>
<th>5 gates with DNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LT-nl</td>
<td>4 gates with LT-l</td>
</tr>
<tr>
<td></td>
<td>3 gates with LT-nl</td>
</tr>
</tbody>
</table>

$TH_i$
What do we know about AON (AND, OR, NOT) circuits?

<table>
<thead>
<tr>
<th>circuit kind</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LT-1</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>LT-nl, d-2</td>
<td>$\left\lfloor \frac{n}{2} \right\rfloor + 1$</td>
</tr>
<tr>
<td>LT-nl -XOR</td>
<td>$\lceil \log_2(n + 1) \rceil$</td>
</tr>
</tbody>
</table>

**LT Constructions for Symmetric Functions**

- Transitions: Muroga
- Intervals: Minnick
- Binary representations: Kautz
- Optimal
DNF Theorem

**DNF Theorem:**

Every Boolean function can be expressed in DNF.

\[(a \cdot \overline{b} + a \cdot c) + \overline{a} = (a \cdot \overline{b}) \cdot (a \cdot c) + \overline{a} = (\overline{a} + b)(\overline{a} + \overline{c}) + \overline{a} = \overline{a} + b \cdot \overline{c} + \overline{a} = \overline{a} + b \cdot \overline{c}\]

**Absorption:**

\[a + \overline{a} = 1 \quad \text{and} \quad a \cdot \overline{a} = 0\]

**T4. DeMorgan Laws:**

\[\overline{a + b} = \overline{a} \cdot \overline{b} \quad \text{and} \quad \overline{a \cdot b} = \overline{a} + \overline{b}\]

**A2. Complements:**

\[a + \overline{a} = 1 \quad \text{and} \quad a \cdot \overline{a} = 0\]

**A4. Distributivity:**

\[a + b \cdot c = (a + b) \cdot (a + c) \quad \text{and} \quad a \cdot (b + c) = a \cdot b + a \cdot c\]
Every 0-1 Boolean Function Can be Implemented Using A Depth Two Circuit

How?

Implement the DNF representation: $OR$ of many $ANDs$
XOR of Four Variables with AON

Depth = 2
Size = 9

4 ANDs for |X| = 1
4 ANDs for |X| = 3

Can we do better?
No  Yes

Size 9 optimal for depth 2

How?
Size 8 AON Circuit for XOR of Four Variables

Size 5

Size 3

$\text{XOR}(x, y, z)$

$\text{XOR}(x, y)$

$\text{XOR}(a, b, c, d)$

optimal size
AON Constructions for XOR

<table>
<thead>
<tr>
<th>circuit kind</th>
<th>size</th>
<th>optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AON, d-2</td>
<td>$2^{n-1} + 1$</td>
<td>maybe optimal</td>
</tr>
<tr>
<td>AON</td>
<td>$\lceil 2.5(n - 1) \rceil$</td>
<td></td>
</tr>
</tbody>
</table>

lower bound: $2n - 1$
Theorem: An optimal size depth-2 AON circuit for $XOR(x_1, x_2, \ldots, x_n)$ has $2^{n-1} + 1$ gates.

Proof:

The construction follows from the DNF representation:

$2^{n-1}$ normal terms + one OR gate

Lower bound: WLOG

(i) Every AND gate must have all $n$ inputs

(ii) Every AND gate computes a normal term

DNF is a representation, hence, there are $2^{n-1}$ AND gates
Depth-2 AON Circuit for XOR

**Theorem:** An optimal size depth-2 AON circuit for \( \text{XOR}(x_1, x_2, \ldots, x_n) \) has \( 2^{n-1} + 1 \) gates.

**Proof (cont):**

Need to prove: (i) Every **AND** gate must have all \( n \) inputs.

By contradiction: Assume that there is a gate \( G \) with \( n-1 \) inputs. Say \( x_1 \) is missing from \( G \).

Assume that:

\[
G(\hat{x}_2, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n) = 1
\]

Hence, the output of the circuit is 1 (the **OR** gate has 1-input).

Note that for both assignments the output of the circuit is 1:

\[
(0, \hat{x}_2, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n) \quad (1, \hat{x}_2, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n)
\]

**QED**

Contradiction!! Those assignments have different parities.
### AON Constructions for XOR

For $n=4$,

<table>
<thead>
<tr>
<th>Circuit Kind</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AON, d-2</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td>AON</td>
<td>$\lceil 2.5(n - 1) \rceil$</td>
</tr>
</tbody>
</table>

**Lower bound:** $2n - 1$

**Optimal:**

- $n=4$: $9$
- $n=8$: $8$

**Maybe optimal:**

- $n=4$: $9$
- $n=8$: $8$
Size 8 AON Circuit for XOR of Four Variables

Idea:
Compute a large XOR by using a circuit of small XOR gates

8 is optimal size
AON Circuit for XOR

Idea:
Compute a large XOR by using a circuit of small XOR gates

8 variables

in-degree = 2
edge = wire

Tree
No cycles

leaf = input edge
node = XOR gate

XOR
AON Circuit for XOR

Idea:
Compute a large XOR by using a circuit of small XOR gates

8 variables

Circuit size in AON gates?

Size = Node size $\times$ number of nodes

$3 \times 7 = 21$

Q: Can we do better for 8 variables

Note that we need size 129 in depth-2...
AON Circuit for XOR

Idea: Use a larger in-degree?

9 variables

Size = Node size × number of nodes

5 × 4 = 20

Size 18 for 8 variables

Note that we need size 21 with in-degree 2

XOR
**AON Circuit for XOR**

**Idea:** Use a larger in-degree?

\[ 2^{d-1} + 1 \]

Circuit size = (node size in AON gates) \( \times \) (number of nodes)

Both are functions of the in-degree \( d \) and the number of variables \( n \) (leaves)

**Lemma:** a tree with in-degree \( d \) and with \( k \) nodes has

\[ n = k(d-1) + 1 \]  leaves

**Proof:**

**Idea:** induction on \( k \); the number of nodes in the tree
AON Circuit for XOR

**Lemma:** A tree with in-degree $d$ and with $k$ nodes has $n = k(d-1) + 1$ leaves.

**Proof:** **Idea:** induction on $k$; the number of nodes in the tree.

Assume true for $k=m$:

$$n = m(d-1) + 1$$

How many leaves we add when we attach a new node?

Hence, for $k=m+1$:

$$n = m(d-1) + 1 + (d-1)$$

$$= (m+1)(d-1) + 1$$

QED
AON Circuit for XOR

Lemma: a tree with in-degree \( d \) and with \( k \) nodes has
\[ n = k(d-1) + 1 \]
leaves

Circuit size = (node size in AON gates) \( \times \) (number of nodes)

\[ 2^{d-1} + 1 \]

\[ k = \frac{n - 1}{d - 1} \]

Circuit size = \( (2^{d-1} + 1) \times \frac{n - 1}{d - 1} = \frac{2^{d-1} + 1}{d - 1} \times (n - 1) \)

<table>
<thead>
<tr>
<th>( d )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4.25</td>
</tr>
</tbody>
</table>

Optimal for \( d = 3 \)

Find an optimal \( d \)

Number of inputs is given
Claim: Given a regular XOR circuit (same in-degree for all XOR gates), the optimal circuit size for XOR of $n$ variables is: $\lceil 2.5(n - 1) \rceil$

Q: Is it optimal?

A1: maybe? We do not know...

A2: we know how to prove a lower bound of $2^{n-1}$ using the forcing-elimination-reverse induction approach.
AON Circuit for XOR

We have a construction of size \([2.5(n-1)]\) we know how to prove a lower bound of \(2n-1\)

Matt Cook proved that an AON circuit of size 7 for XOR does not exist he used a computer search

<table>
<thead>
<tr>
<th>(n)</th>
<th>(2n - 1)</th>
<th>([2.5(n - 1)])</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>18</td>
</tr>
</tbody>
</table>
AON Circuit for XOR

We have a construction of size \(2.5(n - 1)\)
we know how to prove a lower bound of \(2n-1\)

Matching upper/lower bounds = MSc in CS

Matt Cook proved that an AON circuit of size 7 for XOR does not exist
he used a computer search

<table>
<thead>
<tr>
<th>(n)</th>
<th>(2n - 1)</th>
<th>([2.5(n - 1)])</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>18</td>
</tr>
</tbody>
</table>

Prove Matt’s result wo/computer search and get A+ in class

next gap
### AON/ LT Constructions for Symmetric Functions

<table>
<thead>
<tr>
<th>circuit kind</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AON, d-2</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td>AON</td>
<td>$\lceil 2.5(n - 1) \rceil$</td>
</tr>
<tr>
<td>LT-1</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>LT-nl, d-2</td>
<td>$\left\lceil \frac{n}{2} \right\rceil + 1$</td>
</tr>
<tr>
<td>LT-nl-XOR</td>
<td>$\lceil \log_2(n + 1) \rceil$</td>
</tr>
</tbody>
</table>

*Exponential improvement in size using LT gates*
### AON/ LT Constructions for Arithmetic Functions

#### Q: for polynomial size circuit, what is the smallest depth?

For the details, see paper on class web site: Siu and B., 1990

<table>
<thead>
<tr>
<th>Function</th>
<th>AON</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition: $A+B$</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Product: $AxB$</td>
<td>$O(\log_2 n)$</td>
<td>3</td>
</tr>
<tr>
<td>Sorting</td>
<td>$O(\log_2 n)$</td>
<td>3</td>
</tr>
</tbody>
</table>

For integers with $n$ bits

$n$ integers

For integers with $n$ bits
CNS 188a Overview

- Boolean algebra as an axiomatic system
- Boolean functions and their representations using Boolean formulas and spectral methods
- Implementing Boolean functions with relay circuits, circuits of AON (AND, OR, NOT) gates and LT (Linear Threshold) gates
- Analyzing the complexity (size and depth) of circuits
- Relations (as opposed to functions) and their implementation in circuits
- Feedback and convergence in LT circuits
Spectral Analysis of Boolean Functions

Q: why bother?

- Upper and lower bounds on the size of LT circuits
- Synthesis techniques for LT circuits
- Linear vs. Polynomial threshold functions
- Best known learning algorithm for Boolean functions
Representations of Boolean Functions

We will represent the 0 element by 1 and the 1 element by -1.

\[ f : \{1, -1\}^n \rightarrow \{1, -1\} \]

The AND function of two variables:

**DNF:** \[ f(x_1, x_2) = x_1 \cdot x_2 \]

**Truth table:**

<table>
<thead>
<tr>
<th>( x_1x_2 )</th>
<th>( f(x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>1-1</td>
<td>1</td>
</tr>
<tr>
<td>-11</td>
<td>1</td>
</tr>
<tr>
<td>-1-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[ a \mapsto (-1)^a \]

**LT function:** \[ f(x_1, x_2) = sgn(1 + x_1 + x_2) \]

<table>
<thead>
<tr>
<th>( x_1x_2 )</th>
<th>( f(x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>1-1</td>
<td>1</td>
</tr>
<tr>
<td>-11</td>
<td>1</td>
</tr>
<tr>
<td>-1-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
The star of the Show: Polynomial Representation

\[ f(x_1, x_2) = \frac{1}{2}(1 + x_1 + x_2 - x_1x_2) \]

\[ f(1, 1) = \frac{1}{2}(1 + 1 + 1 - 1) = 1 \]

\[ f(1, -1) = \frac{1}{2}(1 + 1 - 1 + 1) = 1 \]

\[ f(-1, 1) = \frac{1}{2}(1 - 1 + 1 + 1) = 1 \]

\[ f(-1, -1) = \frac{1}{2}(1 - 1 - 1 - 1) = -1 \]
The star of the Show: Polynomial Representation

\[ f(x_1, x_2) = \frac{1}{2} (1 + x_1 + x_2 - x_1 x_2) \]

The spectrum is the set of the coefficients of the polynomial representation

Important fact: it works for arbitrary Boolean functions