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
### XOR Function: Size of LT vs. AON in Depth 2

<table>
<thead>
<tr>
<th></th>
<th>$XOR(x_1, x_2, x_3)$</th>
<th>$XOR(x_1, x_2, \ldots, x_n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AON</td>
<td>5</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td>LT-I</td>
<td>4</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>LT-nl</td>
<td>2</td>
<td>$\left\lceil \frac{n}{2} \right\rceil + 1$</td>
</tr>
</tbody>
</table>

**Symmetric functions**
How is SYM related to LT$_2$?

The diagram shows two logic gates, TH$_1$ and TH$_2$, with inputs $x_1$ and $x_2$. The outputs of these gates are then summed with an additional input of -1, leading to an XOR output.

The table below summarizes the truth table for the expression $TH_1 + \overline{TH_2} - 1$:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$TH_1$</th>
<th>$\overline{TH_2}$</th>
<th>$TH_1 + \overline{TH_2} - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Generalization to EQ

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

\[ EQ(j, k) = TH_j + \overline{TH}_{k+1} - 1 \]
Q: What is the generalization to arbitrary symmetric functions?
Q: What is the generalization to arbitrary symmetric functions?

A: Consider the symmetric function table, it is a sum of non-overlapping 1-intervals.

\[ f(X) = \sum_{(j, k) \text{ 1-intervals}} EQ(j, k) \]

Sum of two TH functions
**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

\( TH_i \)
LT-1 Circuit Design Algorithm for SYM

Q: using the design algorithm, how many TH gates are used the first layer?

Definition:
Let $T(f)$ be the number of transitions in the symmetric function table.

A: $T(f)$ TH gates are used
$T(f) = 1$ is the single gate case.

Can we do better?
The Layered Construction for SYM Is OPTIMAL
Summary of the Proof

**Theorem (SYM with LT):**
A layered LT circuit for a symmetric function \( f \), with the first layer being TH gates, has at least \( T(f) + 1 \) gates for \( T(f) \) greater than 1.

**Proof:**
A circuit for a symmetric function (not in TH) has at least 3 gates.

Given a layered LT circuit with \( k \) gates for a symmetric function \( f \):

<table>
<thead>
<tr>
<th>( T(f) )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

There exists a symmetric function \( g \), such that:
- \( T(g) = T(f) - 1 \)
- The function \( g \) can be implemented by a layered LT circuit with \( k - 1 \) gates.
The Layered Construction for SYM Is OPTIMAL

Proof: **Idea:** start with $T(f) = 2$

Need to prove: the LT circuit for $f$ has at least 3 gates

$f$ is not in TH, hence we need at least 2 LT gates

Layered construction

Single input: identity or complement, hence this gate is redundant

Need at least 3 gates!
The Layered Construction for SYM Is OPTIMAL

Say if we prove *:

Given a layered LT circuit with $k$ gates for a symmetric function $f$

There exists a symmetric function $g$, such that:
- $T(g) = T(f) - 1$
- The function $g$ can be implemented by a layered LT circuit with $k-1$ gates

Then it follows that $k \geq T(f) + 1$

Why?

<table>
<thead>
<tr>
<th>$T(f)$</th>
<th>$k$ (circuit size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

By *

contradiction
The Layered Construction for SYM Is OPTIMAL

Need to prove: Given a layered LT circuit with $k$ gates for a symmetric function $f$

There exists a symmetric function $g$, such that:
- $T(g) = T(f) - 1$
- The function $g$ can be implemented by a layered LT circuit with $k-1$ gates

Assume wlog that this is the symmetric function table of $f$
The Layered Construction for SYM Is OPTIMAL

Assume wlog that this is the symmetric function table of $f$.

Claim: There must be an TH gate in the first layer that detects the change from $|X| = i$ to $|X| = i + 1$.

Proof:
Note that it is a layered circuit, namely, inputs are connected only to the first layer!!
The Layered Construction for SYM Is OPTIMAL

**Claim:** There must be an TH gate in the first layer that detects the change from $|X| = i$ to $|X| = i + 1$

**Proof:**

It is a layered circuit, namely, inputs are connected only to the first layer!!

If all TH gates in the first layer output the same for $|X| = i$ and $|X| = i + 1$ then the circuit will not compute $f$: $f(|X| = i) = f(|X| = i + 1)$

Call this “sensitive” gate $S$

Can we eliminate $S$ ??
The Layered Construction for SYM Is OPTIMAL

Can we eliminate $S$??

Easy: $x_1 = x_2 = \cdots = x_{i+1} = 1$

The output of $S$ will be a constant 1 (or 0)

Setting the variables to $x_1 = x_2 = \cdots = x_{i+1} = 1$ results in a new circuit and a new function

Q: are we done?
The Layered Construction for SYM Is OPTIMAL

Given a layered LT circuit with \( k \) gates for a symmetric function \( f \)

There exists a symmetric function \( g \), such that:
- \( T(g) = T(f) - 1 \)
- The function \( g \) can be implemented by a layered LT circuit with \( k-1 \) gates

Setting the variables to \( x_1 = x_2 = \cdots = x_{i+1} = 1 \) results in a new circuit and a new function

- \( g \) is a symmetric function
- \( T(g) = T(f) - 1 \)
- We removed the gate \( S \) (or more), hence, the circuit for \( g \) is layered and has at most \( k-1 \) gates
The Layered Construction for SYM Is OPTIMAL

Remarks

**Theorem (SYM with LT):**
A layered LT circuit for a symmetric function $f$, with the first layer being TH gates, has at least $T(f)+1$ gates for $T(f)$ greater than 1.

**R1: circuit's depth**
- Our proof works for arbitrary depth
- Depth-2 has optimal size
- Depth does not help in reducing the size (layered, TH)

**R2: Open problems**
- A lower bound for arbitrary LT gates (beyond TH)
- Or, a better layered construction using LT gates
### XOR Function: Size of LT vs. AON in Depth 2

<table>
<thead>
<tr>
<th></th>
<th>$XOR(x_1, x_2, x_3)$</th>
<th>$XOR(x_1, x_2, \ldots, x_n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AON</strong></td>
<td>5</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td><strong>LT-I</strong></td>
<td>4</td>
<td>$n + 1$</td>
</tr>
<tr>
<td><strong>LT-nl</strong></td>
<td>2</td>
<td>$\left\lceil \frac{n}{2} \right\rceil + 1$</td>
</tr>
</tbody>
</table>

**Symmetric functions;** $T(f)$ TH gates

The optimal size for $T(f)$ with TH gates.
XOR Function: Size of LT vs. AON in Depth 2

<table>
<thead>
<tr>
<th></th>
<th>$XOR(x_1, x_2, x_3)$</th>
<th>$XOR(x_1, x_2, \ldots, x_n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AON</td>
<td>5</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td>LT-I</td>
<td>4</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>LT-nl</td>
<td>2</td>
<td>$\left\lceil \frac{n}{2} \right\rceil + 1$</td>
</tr>
</tbody>
</table>

Symmetric functions

optimal
LT-nl Construction

$T(f) + 1 = 4$

LT-nl = LT non-layered, inputs go to any layer

\[
\begin{array}{c|c|c|c}
|X| & A & -1-2A+|X| & sgn( ) \\
\hline
0 & 0 & -1 & 0 \\
1 & 0 & 0 & 1 \\
2 & 1 & -1 & 0 \\
3 & 1 & 0 & 1 \\
\end{array}
\]
### LT-nl Construction

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

<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>

$k = 5$

$q_1 = 1$
$q_2 = 3$
$q_3 = 4$
$q_4 = 6$
$q_5 = 7$
LT-nl Construction
Minnick 1961

$TH(q_1+1)$

$TH(q_2+1)$

$TH(q_k+1)$
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 $q_k = n$
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

$k = 3$

$q_1 = 1$
$q_2 = 4$
$q_3 = 5$

<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>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
The Gate M

\[ q_1 = 1 \]
\[ q_2 = 4 \]
\[ q_3 = 5 \]

\[-(q_2-q_1) = -3\]
\[-(q_3-q_2) = -1\]
\[-q_1 = -1\]

0 if \( q_k = n \)
\[ q_1 = 1 \]
\[ q_2 = 4 \]
\[ q_3 = 5 \]

The Gate M
\[ q_1 = 1 \]
\[ q_2 = 4 \]
\[ q_3 = 5 \]
The Circuit

$q_1 = 1$
$q_2 = 4$
$q_3 = 5$

\[
\begin{array}{l}
| X | \quad f(X) \quad a_1 \quad a_2 \\
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
2 & 0 & 1 & 0 \\
3 & 0 & 1 & 0 \\
4 & 1 & 1 & 0 \\
5 & 1 & 1 & 1 \\
\end{array}
\]

\(-3a_1 - a_2 - 1 \quad \ldots + \quad |X| \quad \text{sgn}(\cdot)\)
The Minnick Construction
Why Does it work?

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

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

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

| \( |X| \) | \( TH(q_1 + 1) \) | \( a_1 \) | \( -(q_2 - q_1) \) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | * | * | * |
| 0 | 0 | 0 | 0 |

| \( |X| \) | \( TH(q_2 + 1) \) | \( a_2 \) | \( -(q_3 - q_2) \) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | * | * | * |
| 0 | 0 | 0 | 0 |

| \( |X| \) | \( TH(q_k + 1) \) | \( a_k \) | \( -(q_k - q_{k-1}) \) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | * | * | * |
| 0 | 0 | 0 | 0 |

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

\[ a_1 = 0 \quad a_1 = 1 \quad a_2 = 1 \quad a_3 = 1 \]

\[ a_k = 1 \]

\[ |X| \leq q_1 \]

\[ q_1 < |X| \leq q_2 \]

\[ q_2 < |X| \leq q_3 \]

\[ q_3 < |X| \leq q_4 \]

\[ q_k < |X| \]

\[ - (q_2 - q_1) \]

\[ - (q_3 - q_2) \]

\[ - (q_k - q_{k-1}) \]

\[ - (n + 1 - q_k) \]

0 if \( q_k = n \)
What was Minnick’s key idea?

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

| X | \leq q_1 
| q_1 < |X| \leq q_2 
| q_2 < |X| \leq q_3 
| q_3 < |X| \leq q_4 
| q_k < |X| 

\[ q_1 \]
\[ q_2 - q_1 \]
\[ q_3 - q_2 \]
\[ q_4 - q_3 \]
\[ q_5 - q_4 \]
\[ q_k - q_{k-1} \]
The Key: Telescopic Sum

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

\[
\begin{array}{c}
\text{We are always left with the last element in the sum}
\end{array}
\]

\[ F(X) = -q_5 + |X| \]
The Key: Telescopic Sum

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

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

\begin{align*}
\text{if} & \quad |X| = q_5 \quad \text{then} \quad F(X) = 0 & \implies f(x) = 1 \\
\text{if} & \quad q_4 < |X| < q_5 \quad \text{then} \quad F(X) < 0 & \implies f(x) = 0
\end{align*}

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

\( \sum_{i=1}^{k} a_i(q_{i+1} - q_i) = 0 \) for all the upper bounds of the ranges and negative otherwise
The Example Circuit

<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>$X$</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 Example Circuit

Note that the second TH gate is redundant.

| $|X|$ | $f(X)$ | $a_1$ | $-3a_1-1$ | $\ldots+|X|$ | $\text{sgn}(,:)$ |
|-----|-------|------|-----------|--------------|----------------|
| 0   | 0     | 0    | -1        | -1           | 0              |
| 1   | 1     | 0    | -1        | 0            | 1              |
| 2   | 0     | 1    | -4        | -2           | 0              |
| 3   | 0     | 1    | -4        | -1           | 0              |
| 4   | 1     | 1    | -4        | 0            | 1              |
| 5   | 1     | 1    | -4        | 1            | 1              |
The Minnick Construction
Remarks

R1: The Minnick construction is more general than what we described, it works on intervals; we recommend that you read his paper (it is on the class web site).

R2: The Minnick construction is optimal for LT-nl of depth 2 We will not prove it in the lectures.
### XOR Function: Size of LT vs. AON in Depth 2

<table>
<thead>
<tr>
<th></th>
<th>$XOR(x_1, x_2, x_3)$</th>
<th>$XOR(x_1, x_2, \ldots, x_n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AON</strong></td>
<td>5</td>
<td>$2^{n-1} + 1$</td>
</tr>
<tr>
<td><strong>LT-I</strong></td>
<td>4</td>
<td>$n + 1$</td>
</tr>
<tr>
<td><strong>LT-nl</strong></td>
<td>2</td>
<td>$\lfloor \frac{n}{2} \rfloor + 1$</td>
</tr>
</tbody>
</table>

**Symmetric functions**

**Intervals**

**Transitions**

**Optimal**
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?

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

Q3: For LT-I we proved that depth does not help in reducing the circuit size, does it help in the case of LT-nl?
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.

\[
\text{TH}_i + f_1 + f_2 + f_3
\]
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?

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

Q3: For LT-I we proved that depth does not help in reducing the circuit size, does it help in the case of LT-nI?
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 2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>0 7</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>1 6</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1 5</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>3 4</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>2 2</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>1 1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0 7</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>0 7</td>
</tr>
</tbody>
</table>
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?

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

Q3: For LT-l we proved that depth does not help in reducing the circuit size, does it help in the case of LT-nl?
Questions on LT Constructions

Q3: of for LT-I we proved that depth does not help in reducing the circuit size, does it help in the case of LT-nl?

A3: Yes!

Idea: compute the base-2 representation of |X|

<table>
<thead>
<tr>
<th>X</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XOR(X)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1011</td>
<td>2</td>
</tr>
</tbody>
</table>
Idea: compute the base-2 representation of $|X|$

Algorithm for computing the binary representation:

**Input:** an integer $m$

**Assume:** $2^{j-1} \leq m < 2^j$

$$\sum_{i=0}^{j-1} 2^i = 2^j - 1$$

**Need to compute:** $j$ bits of the binary representation

**Algorithm:**

$A \leftarrow m$

For $i = j - 1$ to 0 do:

$$\{ \text{ if } A \geq 2^i \text{ then } a_i = 1 \text{ else } a_i = 0 \}$$

$A \leftarrow A - a_i 2^i$
**Idea:** compute the base-2 representation of $|X|$

**Algorithm for computing the binary representation:**

**Input:** an integer $m$

**Assume:** $2^{j-1} \leq m < 2^j$

**Need to compute:** $j$ bits of the binary representation

**Algorithm:**

\begin{equation}
A \leftarrow 6 \quad \text{Need 3 bits}
\end{equation}

For $i = \lfloor \frac{j}{2} \rfloor$ from $0$ to $0$ do:

\begin{equation}
\left\{ \begin{array}{l}
\text{if } 6 \geq 4 \text{ then } a_i = 1 \\
\text{else } a_i = 0
\end{array} \right.
\end{equation}

\begin{equation}
A \leftarrow A - 4 \cdot 2^i
\end{equation}

\begin{equation}
a_2 = 1
\end{equation}
Idea: compute the base-2 representation of $|X|$

Algorithm for computing the binary representation:

**Input:** an integer $m$

**Assume:** $2^{j-1} \leq m < 2^j$

**Need to compute:** $j$ bits of the binary representation

**Algorithm:**

\[ A \leftarrow 2 \]

For $i = j \leftarrow 1$ to 0 do:

\[
\begin{cases} 
  \text{if } 2^i \geq 2^{i+1} \text{ then } a_i = 1 \\
  \text{else } a_i = 0
\end{cases}
\]

\[ A \leftarrow A \cdot 2^{-2^i} \]

Need 3 bits

\[
\sum_{i=0}^{j-1} 2^i = 2^j - 1
\]
Idea: compute the base-2 representation of $|X|$

Algorithm for computing the binary representation:

Input: an integer $m$

Assume: $2^{j-1} \leq m < 2^j$

Need to compute: $j$ bits of the binary representation

Algorithm: $A \leftarrow 0$

For $i = j$ to 0 do:

\[
\begin{cases}
    \text{if } 0 \geq 1 \text{ then } a_i = 1 \\
    \text{else } a_i = 0 \\
\end{cases}
\]

$A \leftarrow A \cdot 0 - 0 \cdot 2^i$

Need 3 bits

\[
\sum_{i=0}^{j-1} 2^i = 2^j - 1
\]

$a_2 = 1$

$a_1 = 1$

$a_0 = 0$
Idea: compute the base-2 representation of $|X|$.

Algorithm for computing the binary representation:

**Input:** an integer $m$

**Assume:** $2^{j-1} \leq m < 2^j$

**Need to compute:** $j$ bits of the binary representation

**Algorithm:** $A \leftarrow m$

For $i = j - 1$ to 0 do:

\[
\begin{cases}
    \text{if } A \geq 2^i \text{ then } a_i = 1 \\
    \text{else } a_i = 0
\end{cases}
\]

$A \leftarrow A - a_i2^i$

$a_2 = 1$  $a_1 = 1$  $a_0 = 0$
The Kautz Construction
1961

Implement the algorithm for computing the binary representation using LT gates

Algorithm: $A \leftarrow m$

For $i = j - 1$ to 0 do:

$\begin{cases} 
\text{if } A \geq 2^i \text{ then } a_i = 1 \\
\text{else } a_i = 0 
\end{cases}$

$A \leftarrow A - a_i2^i$
**LT Constructions for Symmetric Functions**

<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>

**optimal**
Linear-Input Logic

ROBERT C. MINNICK†, MEMBER, IRE

Summary—Techniques are developed for the logical design of magnetic core circuits to produce arbitrary single-output combinational switching functions. The approach is based on the relationship of a single magnetic core circuit to a linearly separable switching function. A synthesis procedure is developed which uses a pair of logical primitives, AND with NOT and OR with NOT, which are similar to the STROKE primitive and its inverse. Procedures are developed for the synthesis of symmetric functions which require no more than the integral part of \((n+3)/2\) cores, approximately half the number used in previously published procedures. The synthesis of arbitrary switching circuits is treated as a linear programming problem, and a table of all four-variable circuits is presented in which no circuit requires more than three cores.

IRE TRANSACTIONS ON ELECTRONIC COMPUTERS

Linear-Input Logic*

ROBERT C. MINNICK†, MEMBER, IRE

* Received by the PGEC, August 17, 1960; revised manuscript received, October 3, 1960. This paper is the result of work done while the author was employed by Burroughs Corp., ElectroData Div., Pasadena, Calif. Portions of this paper were delivered at the Sixth Annual Symp. on Computers and Data Processing of the Denver Res. Inst., Denver, Colo.; July, 1959.

The Realization of Symmetric Switching Functions with Linear-Input Logical Elements

WILLIAM H. KAUTZ†, MEMBER, IRE

Previously published solutions for the case of the parity function have required from $1 + \lceil n/2 \rceil$ to $1 + n$ elements. The improved solution presented below makes possible simple parity-checking and error-correction circuits for applications to the recording and transmission of digital data. A 7-input parity gate, for example, requires only 3 linear-input elements (Fig. 2); a gate with up to 15 inputs requires 4 elements, etc.
The Kautz Construction
1961

Fig. 2—Minimal linear-input network for the 7-variable parity function.

ACKNOWLEDGMENT

The author is indebted to R. Minnick of Stanford Research Institute, whose work stimulated his serious interest in the subject area of this paper, and with whom many profitable technical discussions have taken place.
William H. Kautz (Sc.D., M.I.T.) was Staff Scientist at Stanford Research Institute, where for 35 years he conducted scientific and technical research in computer science and various other fields.

He then founded and directed the Center for Applied Intuition, which carried out research on intuition and its applications.

He lives in Prague, Czech Republic, and Tucson, Arizona.