Permutation Circuits

Stefan Gremalschi
CSc8530 Parallel Algorithms
Dr. Sushil Prasad
Outline

- Introduction
  - Problem Definition
  - Circuit Component
  - Terminology
  - Examples
- Determining lower bounds on size
- Circuit Design
  - A Recursive Permutation Circuit
  - Description
  - Example
- Constructive Proof
- Analysis
- Application
  - Block Ciphers and Advanced Encryption Standard
  - Substitution - Permutation Networks
  - SPN Algorithm
  - Example
- References
Problem Definition

- A permutation circuit is a combinational circuit that applies a given permutation $\psi_n$ to its input $x_1, x_2, \ldots, x_n$ to get an output $y_1, y_2, \ldots, y_n$ such that:

$$y_1, y_2, \ldots, y_n = \psi_n(x_1, x_2, \ldots, x_n)$$

An example:

$$\Psi_8 = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
5 & 6 & 2 & 1 & 8 & 7 & 3 & 4
\end{bmatrix}$$

This means:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
A switch as the name suggests is a simple component that can do the following:

- **OFF state** - Inputs are sent to output in the same order.
- **ON state** - Inputs are *switched* or interchanged at the output.

A switch is therefore a programmable permutation circuit for input arrays of length 2.
Terminology

- **Size** of a circuit - Number of components in the circuit.
- **Depth** of a circuit - Number of stages in the circuit.
- **Width** of a circuit - Maximum number of components in a stage.
Constructing P4 from P2s
Programming P4 to give permutation 2, 1, 4, 3

S1 = on
S2 = on
S3 = off
S4 = off
S5 = off
Lower Bounds

Let us say that for an input size $n$ we need $s$ switches.

- Each switch has 2 stages (ON/OFF);
- $s$ switches will have $2s$ stages

To satisfy any permutation,

$$2^s \geq n!$$

$$s \geq n \log n$$

Lower Bound on size is $\Omega(n \log n)$
Permutation Vs. Sorting

• The order in which the inputs to a Sorting Circuit appear at the output, depends on the values of the input.

• Hence, by having inputs in the form of a pair $(i, j)$ (which implies input $i$ is sent to output $j$) we can perform permutations by using a sorting circuit and sorting by the $j$ values.
Circuit Design

Once again we use a recursive design based on smaller permutation circuits.

The basic idea is to design the circuit in 3 layers:

**Stage 1** - The first layer decides which of the 2 Stage 2 circuits the Input goes to.

**Stage 2** - Permutates the input at one scale less.

**Stage 3** - Decides where Output of Stage 2 goes in the final output sequence.
Figure 3.12: A recursive permutation circuit.
Description

We need to show that any permutation can be performed for the given input.

1. If for some output $y_l$ we trace back to input $x_{2k}$ then select its neighbor in switch $l_k (x_{2k-1})$ and set the switches from there to its correct output. If neighbor is already selected, select any other i/p.

2. If for some input $x_k$ output $y_{2k}$ is reached, select its neighbor in switch $O_k (y_{2k-1})$ and set switches from there to correct input.

*Ping - Pong* Technique
Let us construct the circuit for the example shown earlier. It is given below:

\[
\Psi_8 = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
5 & 6 & 2 & 1 & 8 & 7 & 3 & 4 \\
\end{array}
\]

We shall consider it step by step. Our basic building blocks are based on the following:

- \(n=1\) - No switches needed.
- \(n=2\) - One Switch sufficient.
- \(n>2\) - Input fed into \(I\) switches that direct them towards two \(n/2\) permutation circuits.
Constructive Proof [Waksman67]

• Consider a network like the one above with no links. We are given any arbitrary permutation. The upper n/2 circuit is called $P_a$ and the lower $P_b$.

• Start with $v_1$ and establish a link through $P_a$ to some $u$ through its corresponding $I$. Switch $I$ is set if $u$ is even.

• Proceed next with the second $u$ associated with this $I$ and establish a link through $P_b$ to its $v$ through the $0$ associated with it. Set this $0$ if $v$ is even.
Constructive Proof [Waksman67] (cont.)

- Repeat the process until all input-output pairs have been matched.
- Now, since by construction $P_a$ and $P_b$, are each associated with exactly $N/2$ inputs and $N/2$ outputs, and since by assumption $P_a$ and $P_b$ are permutation networks the assignment is complete and the link pattern is as in the figure.
Analysis

1. Depth

\[ d(n) = d \left( \frac{n}{2} \right) + 2 \]

\[ = \left[ d(\frac{n}{4}) + 2 \right] + 2 \]

\[ = d(\frac{n}{2^k}) + 2k \quad (d(2)=1) \]

\[ \frac{n}{2^k} = 2 \]

\[ \log n = k + 1; \quad k = \log n - 1 \]

\[ d(n) = 2 \log n - 1 \]
Analysis (cont.)

2. Width: \( n/2 \)

3. Size \((p)\):

   \[
   p(1) = 0, \quad p(2) = 1
   \]

   \[
   p(n) = 2p(n/2) + n - 1
   \]
Application

- The main interest in permutation circuits is due to their use in routing. This is known as Permutation based routing.
- The problem is that we need $O(n \log n)$ just to set the switches! Several solutions have come up for algorithms that enable the switches to be set in $O(k \log n)$. [Nassimi82]
- Substitution-Permutation Networks. [Douglas2002]
A commonly used design for modern-day block ciphers is that of an iterated cipher:

- The cipher requires the specification of a **round function** and a **key schedule**, and the encryption of a plaintext will proceed through $N_r$ similar **rounds**.

  - **Random key $K$**: used to construct $N_r$ **round keys** (also called **subkeys**), which are denoted $K^1, ..., K^{N_r}$.
  - **Key schedule $(K^1, ..., K^{N_r})$**: constructed from $K$ using a fixed, public algorithm.
  - **Round function $g$**: takes two inputs: a round key ($K^r$) and a current **state** ($w^{r-1}$). $w^r = g(w^{r-1}, K^r)$ is the next state.
  - **Plaintext $x$**: the initial state $w^0$.
  - **Ciphertext $y$**: the state after all $N_r$ rounds done.
Sample Encryption

Encryption operations:

\[ w^0 \leftarrow x \]
\[ w^1 \leftarrow g(w^0, K^1) \]
\[ w^2 \leftarrow g(w^1, K^2) \]
\[ \vdots \]
\[ w^{Nr-1} \leftarrow g(w^{Nr-2}, K^{Nr-1}) \]
\[ w^{Nr} \leftarrow g(w^{Nr-1}, K^{Nr}) \]
\[ y \leftarrow w^{Nr} \]

Decryption operations:

\[ w^{Nr} \leftarrow y \]
\[ w^{Nr-1} \leftarrow g^{-1}(w^{Nr}, K^{Nr}) \]
\[ \vdots \]
\[ w^1 \leftarrow g^{-1}(w^2, K^2) \]
\[ w^0 \leftarrow g^{-1}(w^1, K^1) \]
\[ x \leftarrow w^0 \]

Note: function \( g \) is injective (one-to-one)
Substitution - Permutation Networks (SPN)

- **Cryptosystem: SPN**
  - $l, m$ and $Nr$ are positive integers
  - $\pi_s : \{0,1\}^l \rightarrow \{0,1\}^l$ is a permutation
  - $\pi_p : \{1,\ldots,lm\} \rightarrow \{1,\ldots,lm\}$ is a permutation.
  - $P = C = \{0,1\}^{lm}$, and $K \subseteq (\{0,1\}^{lm})^{Nr+1}$ consist of all possible key schedules that could be derived from an initial key $K$ using the key scheduling algorithm.
  - For a key schedule $(K^1,\ldots,K^{Nr+1})$, we encrypt the plaintext $x$ using Algorithm SPN.
Algorithm SPN

Algorithm SPN: $\langle x, \pi_S, \pi_p, (K^1, \ldots, K^{Nr+1}) \rangle$

1. $w^0 \leftarrow x$
2. for $r \leftarrow 1$ to $Nr - 1$
  a. $u^r \leftarrow w^{r-1} \oplus K^r$
  b. for $i \leftarrow 1$ to $m$
     i. do $v^r_{(i)} \leftarrow \pi_S(u^r_{(i)})$
     ii. $w^r \leftarrow (v^r_{\pi_p(1)}, \ldots, v^r_{\pi_p(m)})$
  c. $u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$
  d. for $i \leftarrow 1$ to $m$
     i. do $v^{Nr}_{(i)} \leftarrow \pi_S(u^{Nr}_{(i)})$
3. $y \leftarrow v^{Nr} \oplus K^{Nr+1}$
4. output $(y)$

$u^r$ is the input to the S-boxes in round $r$.
$v^r$ is the output of the S-boxes in round $r$.
$w^r$ is obtained from $v^r$ by applying $\pi_p$.
$u^{r+1}$ is constructed from $w^r$ by xor-ing with the round key $K^{r+1}$ (called round key mixing).
The very first and last operations are xors with subkeys (called whitening).
Substitution - Permutation Networks (SPN) (cont.)

- **Example 3.1:**
  - Suppose \( l = m = Nr = 4 \). Let \( \pi_S \) be defined as follows, where the input and the output are written in hexadecimal:

<table>
<thead>
<tr>
<th>input</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>E</td>
<td>4</td>
<td>D</td>
<td>1</td>
<td>2</td>
<td>F</td>
<td>B</td>
<td>8</td>
<td>3</td>
<td>A</td>
<td>6</td>
<td>C</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

- Let \( \pi_P \) be defined as follows:

<table>
<thead>
<tr>
<th>input</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

- See next slide for a pictorial representation of this particular SPN, where \( S_{ir} \) means \( i \)-th round, \( r \)-th S-box.
A Substitution – Permutation Network

Suppose the plaintext is \( x = 0010\ 0110\ 1011\ 0111 \)
Then the encryption of \( x \) proceeds as follows:
\[
\begin{align*}
\text{w}_0 &= 0010\ 0110\ 1011\ 0111 \\
\text{K}_1 &= 0011\ 1010\ 1001\ 0100 \\
\text{u}_1 &= 0001\ 1100\ 0010\ 0011 \\
\text{v}_1 &= 0100\ 0101\ 1101\ 0001 \\
\text{w}_1 &= 0010\ 1110\ 0000\ 0111 \\
\text{K}_2 &= 1010\ 1001\ 0100\ 1101 \\
\text{u}_2 &= 1000\ 0111\ 0100\ 1010 \\
\text{v}_2 &= 0011\ 1000\ 0010\ 0110 \\
\text{w}_2 &= 0100\ 0001\ 1011\ 1000 \\
\text{K}_3 &= 1001\ 0100\ 1101\ 0110 \\
\text{u}_3 &= 1101\ 0101\ 0110\ 1110 \\
\text{v}_3 &= 1001\ 1111\ 1011\ 0000 \\
\text{w}_3 &= 1110\ 0100\ 0110\ 1110 \\
\text{K}_4 &= 0100\ 1101\ 0110\ 0011 \\
\text{u}_4 &= 1010\ 1001\ 0000\ 1101 \\
\text{v}_4 &= 0110\ 1010\ 1110\ 1001 \\
\text{K}_5 &= 1101\ 0110\ 0011\ 1111 , \text{ and} \\
\text{y} &= 1011\ 1100\ 1101\ 0110 \text{ is the ciphertext.}
\end{align*}
\]
References

Thank you!