Switch Fabrics

Switching Technology S38.165
http://www.netlab.hut.fi/opetus/s38165

Switch fabrics

• Basic concepts
• Time and space switching
• Two stage switches
• Three stage switches
• Cost criteria
• Multi-stage switches and path search
Cost criteria for switch fabrics

- Number of cross-points
- Fan-out
- Logical depth
- Blocking probability
- Complexity of switch control
- Total number of connection states
- Path search

Cross-points

- Number of cross-points gives the number of on-off gates (usually “and-gates”) in space switching equivalent of a fabric
  - minimization of cross-point count is essential when cross-point technology is expensive (e.g. electro-mechanical and optical cross-points)
  - Very Large Scale Integration (VLSI) technology implements cross-point complexity in Integrated Circuits (ICs)
    => more relevant to minimize number of ICs than number of cross-points
  - Due to increasing switching speeds, large fabric constructions and increased integration density of ICs, power consumption has become a crucial design criteria
    - higher speed => more power
    - large fabrics => long buses, fan-out problem and more driving power
    - increased integration degree of ICs => heating problem
Fan-out and logical depth

- VLSI chips can hide cross-point complexity, but introduce pin count and fan-out problem
  - length of interconnections between ICs can be long lowering switching speed and increasing power consumption
  - parallel processing of switched signals may be limited by the number of available pins of ICs
  - fan-out gives the driving capacity of a switching gate, i.e. number of inputs (gates/cross-points) that can be connected to an output
  - long buses connecting cross-points may lower the number of gates that can be connected to a bus
- Logical depth gives the number of cross-points a signal traverses on its way through a switch
  - large logical depth causes excessive delay and signal deterioration

Blocking probability

- Blocking probability of a multi-stage switching network difficult to determine
- Lee’s approximation gives a coarse measure of blocking
- Assume uniformly distributed load
  - equal load in each input
  - load distributed uniformly among intermediate stages (and their outputs) and among outputs of the switch
- Probability that an input is engaged is \( a = \lambda S \) where
  - \( \lambda \) = input rate on an input link
  - \( S \) = average holding time of a link
**Blocking probability (cont.)**

- Under the assumption of uniformly distributed load, probability that a path between any two switching blocks is engaged is \( p = \frac{a}{nk} \) (\( k \geq n \))
- Probability that a certain path from an input block to an output block is engaged is \( 1 - (1-p)^2 \) where the last term is the probability that both (input and output) links are disengaged
- Probability that all \( k \) paths between an input switching block and an output switching block are engaged is

\[
B = \left[ 1 - (1 - \frac{a}{nk})^2 \right]^k
\]

which is known as Lee’s approximation

---

**Control complexity**

- Give a graph \( G \), a control algorithm is needed to find and set up paths in \( G \) to fulfill connection requirements
- Control complexity is defined by the hardware (computation and memory) requirements and the run time of the algorithm
- Amount of computation depends on blocking category and degree of blocking tolerated
- In general, computation complexity grows exponentially as a function of the number of terminal
- There are interconnection networks that have a regular structure for which control complexity is substantially reduced
- There are also structures that can be distributed over a large number of control units
Management complexity

- **Network management** involves adaptation and maintenance of a switching network after the switching system has been put in place.
- Network management deals with:
  - failure events and growth in connectivity demand
  - changes of traffic patterns from day to day
  - overload situations
  - diagnosis of hardware failures in switching system, control system as well as in access and trunk network
  - in case of failure, traffic is rerouted through redundant built-in hardware or via other switching facilities
  - diagnosis and failure maintenance constitute a significant part of software of a switching system.
- In order for switching cost to grow linearly in respect to total traffic, switching functions (such as control, maintenance, call processing and interconnection network) should be as modular as possible.

Example 1

- A switch with
  - a capacity of \( N \) simultaneous calls
  - average occupancy on lines during busy hour is \( X \) Erlangs
  - \( Y \)% requirement for internal use
  - notice that two (one-way) connections are needed for a call

requires a switch fabric with \( M = 2 \times \frac{(100+Y)}{100} \times \frac{N}{X} \) inputs and outputs.

- If \( N = 20\,000 \), \( X = 0.72 \) and \( Y = 10\% \)

\[
=> M = 2 \times 1.1 \times 20\,000/0.72 = 61\,112
\]

\[
=> \text{corresponds to 2038 E1 links}
\]
Amount of traffic in Erlangs

- Erlang defines the amount of traffic flowing through a communication system - it is given as the aggregate holding time of all channels of a system divided by the observation time period
- Example 1:
  - During an hour period three calls are made (5 min, 15 min and 10 min) using a single telephone channel => the amount of traffic carried by this channel is (30 min/60 min) = 0.5 Erlang
- Example 2:
  - A telephone exchange supports 1000 channels and during a busy hour (10.00 - 11.00) each channel is occupied 45 minutes on the average => the amount of traffic carried through the switch during the busy hour is (1000x45 min / 60 min) = 75 Erlangs

Erlang’s first formula

Erlang 1st formula

\[ E_1(n, A) = \frac{A^n}{n!} \left(1 + \frac{A}{2!} + \cdots + \frac{A^n}{n!}\right) \]

- Erlang 1st formula applies to systems fulfilling conditions
  - A failed call is disconnected (loss system)
  - Full accessibility
  - Time between subsequent calls vary randomly
  - Large number of sources
- \( E_1(5, 2.7) \) implies that we have a system of 5 inlets and offered load is 2.7 Erlangs - blocking calculated using the formula is 8.5%
Example 2

- An exchange for 2000 subscribers is to be installed and it is required that the blocking probability should be below 10%. If E2 links are used to carry the subscriber traffic to telephone network, how many E2 links are needed?
  - average call lasts 6 min
  - a subscriber places one call during a 2-hour busy period (on the average)
- Amount of offered traffic is \((2000 \times 6 \text{ min} / 2 \times 60 \text{ min}) = 100 \text{ Erl.}\)
- Erlang 1st formula gives for 10% blocking and load of 100 Erl. that \(n = 97\)
  \(\Rightarrow\) required number of E1 links is \(\text{ceil}(97/30) = 4\)

Example 3

- Suppose driving current of a switching gate (cross-point) is 100 mA and its maximum input current is 8 mA
- How many output gates can be connected to a bus, driven by one input gate, if the capacitive load of the bus is negligibly small?
  - Fan-out = \(\text{floor}[100/8] = 12\)

  - How many output gates can be connected to a bus driven by one input gate if load of the bus corresponds to 15% of the load of a gate input?)
  - Fan-out = \(\text{floor}[100/(1.15 \times 8)] = 10\)
Switch fabrics

- Basic concepts
- Time and space switching
- Two stage switches
- Three stage switches
- Cost criteria
- Multi-stage switches and path search

Multi-stage switching

- Large switch fabrics could be constructed by using a single $N \times N$ crossbar, interconnecting $N$ inputs to $N$ outputs
  - such an array would require $N^2$ cross-points
  - logical depth = 1
  - considering the limited driving power of electronic or optical switching gates, large $N$ means problems with signal quality (e.g. delay, deterioration)
- Multi-stage structures can be used to avoid above problems
- Major design problems with multi-stages
  - find a non-blocking structure
  - find non-conflicting paths through the switching network
Multi-stage switching (cont.)

- Let’s take a network of $K$ stages
- Stage $k$ ($1 \leq k \leq K$) has $r_k$ switch blocks (SB)
- Switch block $j$ ($1 \leq j \leq r_k$) in stage $k$ is denoted by $S(i,j,k)$
- Switch $i$ has $m_k$ inputs and $n_k$ outputs
- Input $i$ of $S(i,j,k)$ is represented by $e(i,j,k)$
- Output $i$ of $S(i,j,k)$ is represented by $o(i,j,k)$
- Relation $o(i,j,k)= e(i',j',k+1)$ gives interconnection between output $i$ and input $i'$ of switch blocks $j$ and $j'$ in consecutive stages $k$ and $k+1$
- Special class of switches:
  - $n_k = r_{k+1}$ and $m_k = r_{k-1}$
  - each SB in each stage connected to each SB in the next stage

Clos network

$\begin{align*}
m_k &= \text{number of inputs in a SB at stage } k \\
n_k &= \text{number of outputs in a SB at stage } k \\
r_k &= \text{number of SBs at stage } k
\end{align*}$

- parameter $m_1$, $n_3$, $r_1$, $r_2$, $r_3$ chosen freely
- other parameters determined uniquely by $n_1 = r_2$, $m_2 = r_1$, $m_3 = r_2$
Graph presentation of a Clos network

Every SB in stage $k$ is connected to all $r_{k+1}$ SBs in the following stage $k+1$ with a single link.

Path connections in a 3-stage network

- An input of SB $x$ may be connected to an output of SB $y$ via a middle stage SB $a$
- Other inputs of SB $x$ may be connected to other outputs of SB $y$ via other middle stage SBs ($b$, $c$, ...)
- Paull’s connection matrix is used to represent paths in three stage switches
Paull's matrix

- Middle stage switch blocks \((a, b, c)\) connecting 1st stage SB \(x\) to 3rd stage SB \(y\) are entered into entry \((x, y)\) in \(r_1 \times r_3\) matrix
- Each entry of the matrix may have 0, 1 or several middle stage SBs
- A symbol \((a,b,..)\) appears as many times in the matrix as there are connections through it

Paull’s matrix (cont.)

Conditions for a legitimate point-to-point connection matrix:

1. Each row has at most \(m_1\) symbols, since there can be as many paths through a 1st stage SB as there are inputs to it
2. Each column has at most \(n_3\) symbols, since there can be as many paths through a 3rd stage SB as there are outputs from it
Pauli’s matrix (cont.)

Conditions of a legitimate point-to-point connection matrix (cont.):

3 Symbols in each row must be distinct, since only one edge connects a 1st stage SB to a 2nd stage SB
=> there can be at most $r_2$ different symbols

4 Symbols in each column must be distinct, since only one edge connects a 2nd stage SB to a 3rd stage SB and an edge does not carry signals from several inputs
=> there can be at most $r_2$ different symbols

In case of multi-casting, conditions 1 and 3 may not be valid, because a path from the 1st stage may be directed via several 2nd stage switch blocks. Conditions 2 and 4 remain valid.

Strict-sense non-blocking Clos

Definitions:

• $T'$ is a subset of set $T$ of transmitting terminals

• $R'$ is a subset of set $R$ of receiving terminals

• Each element of $T'$ is connected by a legitimate multi-cast tree to a non-empty and disjoint subset $R'$

• Each element of $R'$ is connected to one element of $T'$

A network is strict sense non-blocking if any $t \in T - T'$ can establish a legitimate multi-cast tree to any subset $R - R'$ without changes to the previously established paths.

A rearrangeable network satisfies the same conditions, but allows changes to be made to the previously established paths.
Clos theorem

Clos theorem:
A Clos network is **strict-sense non-blocking** if and only if the number of 2nd stage switch blocks fulfills the condition

\[ r_2 \geq m_1 + n_3 - 1 \]

- A symmetric Clos network with \( m_1 = n_3 = n \) is strict-sense non-blocking if

\[ r_2 \geq 2n - 1 \]

---

Proof of Clos theorem

**Proof 1:**
- Let’s take some SB \( x \) in the 1st stage and some SB \( y \) in the 3rd stage, which both have maximum number of connection minus one. 
  \( \Rightarrow \) \( x \) has \( m_1 - 1 \) and \( y \) has \( n_3 - 1 \) connections
- One additional connection should be established between \( x \) and \( y \)
- In the worst case, existing connections of \( x \) and \( y \) occupy distinct 2nd stage SBs 
  \( \Rightarrow \) \( m_1 - 1 \) SBs for paths of \( x \) has and \( n_3 - 1 \) SBs for paths of \( y \)
- To have a connection between \( x \) and \( y \) an additional SB is needed in the 2nd stage 
  \( \Rightarrow \) required number of SBs is \( (m_1 - 1) + (n_3 - 1) + 1 = m_1 + n_3 - 1 \)
Visualization of proof

Paull’s matrix and proof of Clos theorem

Proof 2:

- A connection from an idle input of a 1st stage SB $x$ to an idle output of a 3rd stage SB $y$ should be established
- $m_1-1$ symbols can exist already in row $x$, because there are $m_1$ inputs to SB $x$.
- $n_3-1$ symbols can exist already in row $y$, because there are $n_3$ outputs to SB $y$.
- In the worst case, all the $(m_1-1 + n_3-1)$ symbol are distinct
- To have an additional path between $x$ and $y$, one more SB is needed in the 2nd stage
  $\Rightarrow m_1 + n_3 - 1$ SBs are needed
Procedure for making connections

- Keep track of symbols used by row \( x \) using an occupancy vector \( u_x \) (which has \( r_2 \) entries that represent SBs of the 2nd stage)
- Enter “1” for a symbol in \( u_x \) if it has been used in row \( x \), otherwise enter “0”
- Likewise keep track of symbols used by column \( y \) using an occupancy vector \( u_y \)
- To set up a connection between SB \( x \) and SB \( y \) look for a position \( j \) in \( u_x \) and \( u_y \) which has “0” in both vectors
- Amount of required computation is proportional to \( r_2 \)

\[
\begin{array}{c|ccc|c|c}
 & 1 & 2 & 3 & j & r_i \\
\hline
u_x & 0 & 1 & 1 & 0 & 0 & 1 \\
& 1 & 2 & 3 & j & r_i \\
\hline
u_y & 1 & 1 & 0 & 0 & 1 & 0 \\
& 1 & 2 & 3 & j & r_i \\
\end{array}
\]

common “0”

Rearrangeable networks

Slepian-Duguid theorem:
A three stage network is rearrangeable if and only if
\[ r_2 \geq \max(m_1, n_2) \]

A symmetric Clos network with \( m_1 = n_2 = n \) is rearrangeably non-blocking if
\[ r_2 \geq n \]

Paull’s theorem:
The number of circuits that need to be rearranged is at most
\[ \min(r_n, r_d) - 1 \]
Connection rearrangement by Paull's matrix

- If there is no common symbol (position j) found in \( u_x \) and \( u_y \), we look for symbols in \( u_x \) that are not in \( u_y \) and symbols in \( u_y \) not found in \( u_x \).
  
- Let's suppose there is symbol \( a \) in \( u_x \) (not in \( u_y \)) and symbol \( b \) in \( u_y \) (not in \( u_x \)) and let's choose either one as a starting point.
  
- Let it be \( a \) then \( b \) is searched from the column in which \( a \) resides (in row \( x \)) - let it be column \( j_1 \) in which \( b \) is found in row \( i_1 \).
  
- In row \( i_1 \), search for \( a \) - let this position be column \( j_2 \).
  
- This procedure continues until symbol \( a \) or \( b \) cannot be found in the column or row visited.

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
1 & 2 & a & b \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
1 & 2 & a & b \\
\end{array}
\]

Connection rearrangement by Paull's matrix (cont.)

- At this point connections identified can be rearranged by replacing symbol \( a \) (in rows \( x, i_1, i_2, \ldots \)) by \( b \) and symbol \( b \) (in columns \( y, j_1, j_2, \ldots \)) by \( a \).

- \( a \) and \( b \) still appear at most once in any row or column.

- 2nd stage SB \( a \) can be used to connect \( x \) and \( y \).
Example of connection rearrangement by Paull’s matrix

- Let’s take a three-stage network 24x25 with \( r_1=4 \) and \( r_3=5 \)
- Rearrangeability condition requires that \( r_2 \geq 6 \)
  - let these SBs be marked by \( a, b, c, d, e \) and \( f \)
  
  \[ \Rightarrow m_1 = 6, \; n_1 = 6, \; m_2 = 4, \; n_2 = 5, \; m_3 = 6, \; n_3 = 5 \]

Example of connection rearrangement by Paull’s matrix (cont.)

- In the network state shown below, a new connection is to be established between SB1 of stage 1 and SB1 of stage 3
- No SBs available in stage 2 to allow a new connection
- Slepian-Duguid theorem \(
\Rightarrow \) a three stage network is rearrangeable if and only if \( r_2 \geq \max(m_1, n_3) \)
  - \( m_1 = 6, \; n_3 = 5, \; r_2 = 6 \) \(
\Rightarrow \) condition fulfilled
- SBs \( c \) and \( d \) are selected to operate rearrangement

<table>
<thead>
<tr>
<th>Stage SBs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st stage SBs</td>
<td>1</td>
<td>f</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>a,b</td>
<td>d</td>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>c</td>
<td>e,f</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>d</td>
<td>c</td>
<td>a</td>
<td>b,f</td>
</tr>
</tbody>
</table>

Occupancy vectors of SB1/stage 1 and SB1/stage 3

- \( U_{1,t} = 1 \; 1 \; 0 \; 1 \; 1 \)
- \( a \; b \; c \; d \; e \; f \)
- \( U_{2,t} = 1 \; 1 \; 0 \; 1 \; 0 \; 0 \)
- \( a \; b \; c \; d \; e \; f \)
Example of connection rearrangement by Paull’s matrix (cont.)

- Start rearrangement procedure from symbol c in row 1 and column 5
- 5 connection rearrangements are needed to set up the required connection - Paull’s theorem !!!

Example of connection rearrangement by Paull’s matrix (cont.)

- Paull's theorem states that the number of circuits that need to be rearranged is at most \( \min(r_1, r_3) - 1 = 3 \)
  => there must be another solution
- Start rearrangement procedure from d in row 4 and column 1
  => only one connection rearrangement is needed
Recursive construction of switching networks

- To reduce cross-point complexity of three stage switches individual stages can be factored further
- Suppose we want to construct an $NxN$ switching network and let $N = pxq$
- A rearrangeably non-blocking Clos network is constructed recursively by connecting a $pxp$, $qxq$ and $pxp$ rearrangeably non-blocking switch together in respective order
  => under certain conditions result may be a strict-sense non-blocking network
- A strict-sense non-blocking network is constructed recursively by connecting a $p(2p - 1)$, $qxq$ and $p(2p - 1)$ strict-sense non-blocking switch together in respective order
  => result may be a rearrangeable non-blocking network

3-dimensional construction of a rearrangeably non-blocking network

Number of cross-points for the rearrangeable construction is

$$p^2q + q^2p + p^2q = 2p^2q + q^2p$$
3-dimensional construction of a strict-sense non-blocking network

Number of cross-points for the strictly non-blocking construction is

\[ p(2p - 1)q + q^2(2p - 1) + p(2p - 1)q = 2p(2p - 1)q + q^2(2p - 1) \]

Recursive factoring of switching networks

- \( N \) can be factored into \( p \) and \( q \) in many ways and these can be factored further
- Which \( p \) to choose and how should the sub-networks be factored further?
- Doubling in the 1st and 3rd stages suggests to start with the smallest factor and recursively factor \( q = N/p \) using the next smallest factor
  => this strategy works well for rearrangeable networks
  => for strict-sense non-blocking networks width of the network is doubled
  => not the best strategy for minimizing cross-point count
- Ideal solution: low complexity, minimum number of cross-points and easy to construct => quite often conflicting goals
Recursive factoring of a rearrangeably non-blocking network

- Special case $N = 2^n$, $n$ being a positive integer
  => a rearrangeable network can be constructed by factoring $N$ into $p = 2$ and $q = N/2$
  => resulting network is a Benes network
  => each stage consists of $N/2$ switch blocks of size $2 \times 2$
- Factor $q$ relates to the multiplexing factor (number of time-slots on inputs)
  => recursion continued until speed of signals low enough for real implementations

---

Benes network

Number of stages in a Benes network

$$K = 2\log_2 N - 1$$
Benes network (cont.)

- Benes network is recursively constructed of 2x2 switch blocks and it is rearrangeably non-blocking (see Clos theorem)
- First half of Benes network is called baseline network
- Second half of Benes network is a mirror image (inverse) of the first half and is called inverse baseline network
- Number of switch stages is \( K = 2 \log_2 N - 1 \)
- Each stage includes \( N/2 \) 2x2 switching blocks (SBs) and thus number of SBs of a Benes network is \( N \log_2 N - (N/2) = N(\log_2 N - 1/2) \)
- Each 2x2 SB has 4 cross-points and number of cross-points in a Benes network is \( 4(N/2)(2\log_2 N - 1) = 4N\log_2 N - 2N \approx 4N\log_2 N \)

Illustration of recursively factored Benes network