Consider addition of the following numbers:

\[
\begin{align*}
\cdots & a_{k+2} a_{k+1} a_k 010101 \ a_{l+2} a_{l+1} a_l \cdots \\
\cdots & b_{k+2} b_{k+1} b_k 101010 \ b_{l+2} b_{l+1} b_l \cdots
\end{align*}
\]

- If \( a_{k+3} = 1 \) \( \rightarrow \) carry will propagate to position \( k \)
- To speed-up operation, propagation is skipped to position \( i \) without waiting for rippling.
- Operation time varies according to operands as in carry-complete addition.
- To implement carry-skip adder, stages are divided into blocks.

The carry-skip adder diagram shows the flow of propagation and skipping. Carry-skip logic is added to each block to detect when carry-in the block can be passed directly to the next block.

Define the carry transfer:

\[
T_i = a_i + b_i
\]

Carry skipping can be detected for a block size of \( m \) as follows (carry propagates through all stages):

\[
T_j \cdot T_{j+1} \cdots T_{j+m-1} = 1 \quad (= (a_j + b_j) \cdot (a_{j+1} + b_{j+1}) \cdots)
\]

- Note: This takes into account both propagated and generated carries.
- Carry out from the block (\( m \)-bits in a block) is:

\[
\frac{T_j \cdot T_{j+1} \cdots T_{j+m-1} \cdot c_j + c_{j+m}}{skipped \quad generated}
\]
• block size in carry-skip adder is very important

• worst case operation time takes place when
  – carry is generated in the first block
  – carry skips intermediate stages
  – carry is killed in the last block

• worst case addition time is \( \left( \frac{2n}{m} + 4m - 4 \right) \tau \) (\( n=\)adder width, \( m=\)block size)

• for optimal block size, minimize delay:
  \[
  \frac{d}{dm} \left( \frac{2n}{m} + 4m - 4 \right) = -2 \left( \frac{n}{m^2} - 2 \right) \\
  \Rightarrow m = \sqrt{\frac{n}{2}}
  \]

• in practise, non-uniform block sizes gives the best performance

• in general, outer blocks should be smaller than middle blocks