Conditional Sum Adders

\[ s_i^0 = x_i \oplus y_i \]
\[ s_i^1 = x_i \oplus y_i \]
\[ c_i^0 = x_i \cdot y_i \]
\[ c_i^1 = x_i + y_i \]

Sklansky,
Trans on Comp
6/60
Conditional Sum Adders

TG Conditional Sum

Conditional Sum Adder

Conditional Cell

Rothermel, JSSC 89
**TG Conditional Sum**

- Serial connection of transmission gates
- Chain length = $1 + \log_2 n$

**DPL Conditional Sum**

CLA

“Conditional carry select”
DPL Conditional Sum

Block Conditional Sums

Carry-Select Adder

Setup

"0" Carry Propagation

"1" Carry Propagation

C_{0,k-1} → Multiplexer → C_{0,k+3}

Sum Generation
Carry Select Adder: Critical Path

Bit 0-3

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{0-3}

Bit 4-7

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{4-7}

Bit 8-11

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{8-11}

Bit 12-15

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{12-15}

Linear Carry Select

Bit 0-3

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{0-3}

Bit 4-7

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{4-7}

Bit 8-11

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{8-11}

Bit 12-15

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S_{12-15}

\[ t_{add} = t_{setup} + \left( \frac{N}{M} \right) t_{carry} + M t_{mux} + t_{sum} \]
Square Root Carry Select

Define 3 new variable which ONLY depend on A, B

- Generate \( G = AB \)
- Propagate \( P = A \oplus B \)
- Delete \( D = A \cdot B \)

\[
\begin{align*}
C_o(G, P) &= G + PC_i \\
S(G, P) &= P \oplus C_i
\end{align*}
\]

Can also derive expressions for \( S \) and \( C_o \) based on \( D \) and \( P \)
Carry Lookahead Adder

Lookahead Adder Equations

Position $i$: $c_i = g_i + p_i c_{i-1}$

Position $i + 1$: $c_{i+1} = g_{i+1} + p_{i+1} c_i$

$= g_{i+1} + p_{i+1} (g_i + p_i c_{i-1})$

$= g_{i+1} + p_{i+1} g_i + p_{i+1} p_i c_{i-1}$

Carry exists if:
- generated in stage $i + 1$
- generated in stage $i$ and propagated through $i + 1$
- propagated through both $i$ and $i + 1$
Lookahead Adder

- Unrolling of carry recurrence can be continued
- If unrolled to level $k$, resulting in two-level AND-OR structure
  - AND Fan-In = $k + 1$, OR Fan-In = $k + 1$
  - $k + 1$ transistors in the MOS stack
  - Limits $k$ to 3 -4
Block Lookahead

Fourth bit carry:
\[ c_{i+4} = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}g_{i+1}p_i + c_{i-1} \]

Block generate and block propagate:
\[ G_{i,i+3} = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_i \]
\[ P_{i,i+3} = p_{i+3}p_{i+2}p_i + 1 \]
\[ c_{i+4} = G_{i,i+3} + P_{i,i+3}c_{i-1} \]

Can create groups of groups, or ‘super-groups’:
\[ \hat{G}_j = G_{j+3} + P_{j+3}G_{j+2} + P_{j+3}P_{j+2}G_{j+1} + P_{j+3}P_{j+2}P_{j+1}G_j \]
\[ \hat{P}_j = P_{j+3}P_{j+2}P_{j+1}P_j \]

Delay is \( t_d = c_1 \log[N] \)
Block Lookahead

From Oklobdzija

Critical path delay = 1Δ (for g,p) + 2x2 Δ (for G,P) + 3x2 Δ (for Cin) + 1XOR- Δ (for Sum) = approx. 12 Δ of delay

Lookahead Example

Multiple Output Domino (MODL)
Lookahead Example

4-bit group generate

4-bit group propagate

64-b Lookahead Example
**Lookahead Example**

\[ P = A \oplus B \text{ (propagate)} \]
\[ G = A \cdot B \text{ (generate)} \]
\[ C_0 \text{ (carry-in)} \]
\[ C_1 = G_0 + C_0 P_0 \]
\[ C_2 = G_1 + C_1 P_1 = G_0 P_0 + C_0 P_0 P_1 \]
\[ C_3 = G_2 + C_2 P_2 = G_1 P_1 + G_0 P_0 P_1 + C_0 P_0 P_1 P_2 \]
\[ C_4 = G_3 + C_3 P_3 = G_2 P_2 + G_1 P_1 P_2 + G_0 P_0 P_1 P_2 + C_0 P_0 P_1 P_2 P_3 \]
\[ \vdots \]
\[ C_8 = G_{7,k} + C_7 P_{7,k} = G_{6,k} P_{6,k} + G_{5,k} P_{5,k} P_{6,k} + C_6 P_{6,k} P_{7,k} + \ldots \]
\[ C_{10} = G_{9,k} + C_9 P_{9,k} \]
\[ C_{16} = G_{15,k} + C_{15} P_{15,k} \]

\[ S_n = P_n \oplus C_n \]

**Logarithmic Lookahead Adders**

\[ t_p \sim \log_2(N) \]

\[ t_p \sim N \]
Tree Adders

\[ P_G = p_m \cdot p_l \quad \text{\(m\) – more significant} \]
\[ G_G = g_m + p_m \cdot g_l \]

Start from the input \(P, G\), and continue up the tree
2-bit groups, then 4-bit groups, …

\[(g, p) = (g_m, p_m) \cdot (g_l, p_l) = (g_m + p_m \cdot g_l, p_m \cdot p_l)\]

Kogge, Stone, Trans on Comp,’73

Kogge-Stone Tree
Brent-Kung Adder

\[ t_{\text{add}} \sim \log_2(N) \]

Brent, Kung, Trans on Comp, 3/82

Brent-Kung Tree
**Ling Adder**

Variation of CLA

\[
p_i = a_i \oplus b_i
\]

\[
g_i = a_i \cdot b_i
\]

\[
G_i = g_i + p_i \cdot G_{i-1}
\]

\[
S_i = p_i \oplus G_{i-1}
\]

Ling’s equations

\[
t_i = a_i + b_i
\]

\[
g_i = a_i \cdot b_i
\]

\[
H_i = g_i + t_{i-1} \cdot H_{i-1}
\]

\[
S_i = t_i \oplus H_i + g_i t_{i-1} H_{i-1}
\]

Ling, IBM J. Res. Dev, 5/81

**Ling Adder**

Conventional CLA:

\[
G_i = g_i + p_i \cdot G_{i-1}
\]

Ling’s equation shifts the index of pseudo carry

Also:

\[
G_i = g_i + t_i \cdot G_{i-1}
\]

\[
H_i = g_i + t_{i-1} \cdot G_{i-1}
\]

Propagates information on two bits

Doran, Trans on Comp 9/88
Ling Adder

Conventional

\[ G_3 = g_3 + t_3 g_2 + t_3 t_2 g_1 + t_3 t_2 t_1 g_0 \]

Ling

\[ H_3 = g_3 + t_2 g_2 + t_2 t_1 g_1 + t_2 t_1 t_0 g_0 \]

= \[ g_3 + g_2 + t_2 g_1 + t_2 t_1 g_0 \]

HP Adder

Naffziger, ISSCC’96

\[ i_4 = p_3 p_2 p_1 p_0 \]
HP Adder

Carry ripple

SUMH

COUTBH

C3

G8

CINBH

COUT1H

C3

G8

CINIH

Sum select

LONGC_H

LONGC_L

CIL

CIH

CBL

CBH

P

G

Hybrid Adders

Dobberpuhl, JSSC 11/92

DEC Alpha 21064
DEC Adder

- Combination:
  - 8-bit tapered pre-discharged Manchester carry chains, with $C_{in} = 0$ and $C_{in} = 1$
  - 32-bit LSB carry-lookahead
  - 32-bit MSB conditional sum adder
  - Carry-select on most significant bits
  - Latch-based timing