Conditional Sum Adders

\[ x = 1011101101101101101 \]
\[ y = 0001100110110110110. \]

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

Sklansky,
Trans on Comp
6/60
Conditional Sum Adders

UC Berkeley EE241

B. Nikolić

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

"Conditional carry select"
DPL Conditional Sum

Block Conditional Sums

Carry-Select Adder

Setup

"0" Carry Propagation

"1" Carry Propagation

Multiplexer

Sum Generation

UC Berkeley EE241

B. Nikolić
Carry Select Adder: Critical Path

Linear Carry Select

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

Propagate and Generate Signals

Define 3 new variables which ONLY depend on A, B

\[ \text{Generate } (G) = AB \]
\[ \text{Propagate } (P) = A \oplus B \]
\[ \text{Delete } = A \land B \]

\[ C_o(G, P) = G + PC_i \]
\[ S(G, P) = P \oplus C_i \]

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


Lookahead Adder

Lookahead 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:
\[ c_{i+4} = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}g_{i+1} \]

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

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

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

From Oklobdzija

Carry-lookahead super-blocks of 4-bits blocks generating: G^*, P^*, and C^*_n for the 4-bit blocks

Carry-lookahead blocks of 4-bits generating: G_n, P_n, and C_n for the adders

Group producing final carry C_{14H} and C_{16}

Critical path delay = 1Δ (for g1, p1) + 2x2 Δ (for G, P) + 3x2 Δ (for Cin) + 1xΔOR- Δ (for Sum) = appx. 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

\[ P = A \oplus B \text{ (propagate)} \]
\[ G = A \cdot B \text{ (generate)} \]
\[ C_0 \text{ (carry-in)} \]
\[ C_1 = G_0 + C_0 P_0 \]
\[ C_2 = G_1 + C_1 P_1 + G_0 P_1 + C_0 P_2 P_1 \]
\[ C_3 = G_2 + C_2 P_2 + G_1 P_2 + G_0 P_2 P_1 + C_0 P_2 P_1 P_2 P_1 \]
\[ C_4 = G_3 + C_3 P_3 + G_2 P_3 + G_1 P_2 P_2 P_1 + G_0 P_3 P_2 P_1 + C_0 P_3 P_2 P_1 P_2 P_1 \]
\[ C_5 = G_4 + C_4 P_4 + G_3 P_4 + G_2 P_3 P_3 P_2 P_1 + G_1 P_2 P_2 P_2 P_1 \]
\[ C_6 = G_5 + C_5 P_5 + G_4 P_5 + G_3 P_4 P_4 P_3 P_2 P_1 \]
\[ C_7 = G_6 + C_6 P_6 + G_5 P_6 + G_4 P_5 P_5 P_4 P_3 P_2 P_1 \]

Logarithmic Lookahead Adders

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

\[ t_p \sim N \]
Tree Adders

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

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

UC Berkeley EE241  
B. Nikolić
Brent-Kung Adder

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

Brent, Kung,
Trans on Comp, 3/82

---

Brent-Kung Tree

UC Berkeley EE241  B. Nikolić
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:

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

Also:

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

Ling’s equation shifts the index of pseudo carry

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

Propagates information on two bits

Doran, Trans on Comp 9/88
**Ling Adder**

**Conventional**

\[ G_3 = g_3 + t_3g_2 + t_3t_2g_1 + t_3t_2t_1g_0 \]

**Ling**

\[ H_3 = g_3 + t_2g_2 + t_2t_1g_1 + t_2t_1t_0g_0 \]

\[ = g_3 + g_2 + t_2g_1 + t_2t_1g_0 \]

---

**HP Adder**

\[ i_4 = p_3p_2p_1p_0 \]

Naffziger, ISSCC'96
**HP Adder**

Carry ripple

```
COUTBH  G3  CINBH
  1
COUT1H  G3  CINIH
```

Sum select

```
SUMH
```

**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