ECE 274 – Digital Logic
Optimization: Carry-Lookahead Adders

Digital Design (Vahid): Ch. 6.4

Faster Adder

- Built carry-ripple adder in Ch 4
  - Conv: Slow
    - Output is not correct until the carries have rippled to the left
  - Pro: Small
  - 4-bit carry-ripple adder has 4*5 = 20 gates

Faster Adder – (Bad) Attempt at “Lookahead”

- Idea
  - Modify carry-ripple adder – For a stage’s carry-in, don’t wait for carry to ripple, but rather directly compute from inputs of earlier stages
  - Called “lookahead” because current stage “looks ahead” at previous stages rather than waiting for carry to ripple to current stage

Notice – no rippling of carry

Optimization: Faster Adder

- Faster adder – Use two-level combinational logic design process
  - Recall that 4-bit two-level adder was big
    - Pro: Fast
      - 2 gate delays
    - Con: Large
      - Truth table would have $2^{n+2} = 256$ rows
      - Plot shows 4-bit adder would use about 500 gates

Is there a compromise design?
- Between 2 and 8 gate delays
- Between 20 and 500 gates

Faster Adder – (Bad) Attempt at “Lookahead”

- Want each stage’s carry-in bit to be function of external inputs only ($a$, $b$, or $c$)
  - Recall full-adder equations:
    - $c = bc + ac + ab$
    - $s = a \oplus b$

Stage 1: Carry-in is already an external input, $c_0$

Stage 2: $c_2 = c_1 c_0 + a_1 b_0 + a_0 b_1$

Stage 3: $c_3 = c_2 c_1 + a_2 b_0 + a_0 b_2$

Stage 4: $c_4 = c_3 c_2 + a_3 b_0 + a_0 b_3$

Continue for $c_5$

ECE 274 – Digital Logic
Optimization: Carry-Lookahead Adders

Digital Design (Vahid): Ch. 6.4

Faster Adder

- Faster – Use two-level combinational logic design process
  - Recall that 4-bit two-level adder was big
    - Pro: Fast
      - 2 gate delays
    - Con: Large
      - Truth table would have $2^{n+2} = 256$ rows
      - Plot shows 4-bit adder would use about 500 gates

Is there a compromise design?
- Between 2 and 8 gate delays
- Between 20 and 500 gates

Faster Adder – (Bad) Attempt at “Lookahead”

- Want each stage’s carry-in bit to be function of external inputs only ($a$, $b$, or $c$)
  - Recall full-adder equations:
    - $c = bc + ac + ab$
    - $s = a \oplus b$

Stage 1: Carry-in is already an external input, $c_0$

Stage 2: $c_2 = c_1 c_0 + a_1 b_0 + a_0 b_1$

Stage 3: $c_3 = c_2 c_1 + a_2 b_0 + a_0 b_2$

Stage 4: $c_4 = c_3 c_2 + a_3 b_0 + a_0 b_3$

Continue for $c_5$
**Faster Adder – (Bad) Attempt at "Lookahead"**

- Carry lookahead logic function of external inputs
  - No waiting for ripple
- Problem
  - Equations get too big
  - Not efficient
  - Need a better form of lookahead

\[
\begin{align*}
S_2 &= b_0\text{and}\ b_1 + b_0\text{or}\ a_0 + a_1\text{or}\ a_0 + a_1 + a_0b_1 + a_1b_0 + a_0b_1 + a_1b_0
\end{align*}
\]

- Equations before plugging in
- Carry-lookahead logic quickly computes the carry
- Each stage has SPG block with 2 gate levels
- Need a better form of
to get carries: 
\[
\begin{align*}
\text{cout} &= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0 \\
c_3 &= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0c_0 \\
c_2 &= G_1 + P_1G_0 + P_1P_0c_0 \\
c_1 &= G_0 + P_0c_0
\end{align*}
\]

- Carry-lookahead equations
- Equations get too big
- No waiting for ripple
- Why those names?
  - \( G \): if \( a \) and \( b \) are 1, carry-out will be \( 1 \) — generate a carry-out of \( 1 \) in this case
  - \( P \): if only one of \( a \) or \( b \) is \( 1 \), then carry-out will equal the carry-in — propagate the carry-in to the carry-out in this case

**Better Form of Lookahead**

- Have each stage compute two terms
  - Propagate: \( P = a \oplus b \)
  - Generate: \( G = ab \)
- Compute lookahead from \( P \) and \( G \) terms, not from external inputs
  - Why \( P \& G \)? Because the logic comes out much simpler
    - Very clever finding; not particularly obvious though
    - Why these names?
      - \( G \) if \( a \) and \( b \) are 1, carry-out will be 1 — generate a carry-out of 1 in this case
      - \( P \) if only one of \( a \) or \( b \) is 1, then carry-out will equal the carry-in — propagate the carry-in to the carry-out in this case

**Carry-Lookahead Adder – High-Level View**

- Fast — only 4 gate delays
  - Each stage has SPG block with 2 gate levels
  - Carry-lookahead logic quickly computes the carry from the propagate and generate bits using 2 gate levels inside
- Reasonable number of gates — 4-bit adder has only 26 gates

**Carry-Lookahead Adder – 32-bit?**

- Problem: Gates get bigger in each stage
  - 4th stage has 5-input gates
  - 32-bit stage would have 33-input gates
    - Too many inputs for one gate
    - Would require building from smaller gates, meaning more levels (slower), more gates (bigger)
- One solution: Connect 4-bit CLA adders in ripple manner
  - But slow (4 + 4 + 4 + 4 gate delays)

- 4-bit adder comparison
  - Carry-ripple: \((B, 20)\)
  - Two-level: \((2, 500)\)
  - CLA: \((A, 26)\)
  - Nice compromise
Hierarchical Carry-Lookahead Adders

- Better solution — Rather than rippling the carries, just repeat the carry-lookahead concept
  - Requires minor modification of 4-bit CLA adder to output P and G

\[
\begin{array}{cccc}
\text{a3} & \text{a2} & \text{a1} & \text{a0} \\
\text{b3} & \text{s3} & \text{s2} & \text{s1} & \text{s0} \\
\text{cout} & \text{P} & \text{G} & \text{cin} & \text{b2} & \text{b1} & \text{b0}
\end{array}
\]

4-bit adder

\[
\begin{array}{cccc}
\text{a15} & \text{a12} & \text{a8} & \text{a7} \\
\text{b15} & \text{b12} & \text{b8} & \text{b7} \\
\text{cin} & \text{b2} & \text{b1} & \text{b0}
\end{array}
\]

4-bit carry-lookahead logic

\[
\begin{array}{cccc}
\text{a3} & \text{a2} & \text{a1} & \text{a0} \\
\text{b3} & \text{s3} & \text{s2} & \text{s0} & \text{cout} & \text{PG} & \text{cin} & \text{b2} & \text{b1} & \text{b0}
\end{array}
\]

4-bit adder

These use carry-lookahead internally

Second level of carry-lookahead

Same lookahead logic as inside the 4-bit adders

Hierarchical CLA concept can be applied for larger adders

- 32-bit hierarchical CLA
  - Only about 8 gate delays (2 for SPG block, then 2 per CLA level)
  - Only about 14 gates in each 4-bit CLA logic block

Hierarchical Carry-Lookahead Adders

Q: How many gate delays for 64-bit hierarchical CLA, using 4-bit CLA logic?
A: 16 CLA-logic blocks in 1st level, 4 in 2nd, 1 in 3rd — so still just 8 gate delays (2 for SPG, and 2+2+2 for CLA logic), CLA is a very efficient method.

Carry Select Adder

- Another way to compose adders
  - High-order stage — Compute result for carry in of 1 and of 0
  - Select based on carry-out of low-order stage
  - Faster than pure rippling

\[
\begin{array}{cccc}
\text{a3} & \text{a2} & \text{a1} & \text{a0} \\
\text{a7} & \text{a6} & \text{a5} & \text{a4} \\
\text{b3} & \text{b2} & \text{b1} & \text{b0}
\end{array}
\]

4-bit adder

\[
\begin{array}{cccc}
\text{ci} & \text{HI4_1} & \text{HI4_0} & \text{S}\text{Q} \\
\text{s5} & \text{s4} & \text{s3} & \text{s0} \\
\text{cin} & \text{LO4} & \text{I1} & \text{I0}
\end{array}
\]

Design Challenge

- Design Challenge
  - Design a 24-bit carry-lookahead adder using 4-bit carry-lookahead adders
    - What is the total delay through the 24-bit adder?
    - How much faster is the carry-lookahead adder compared to a 24-bit carry-ripple adder?

Adder Tradeoffs

- Designer picks the adder that satisfies particular delay and size requirements
  - May use different adder types in different parts of same design
    - Faster adders on critical path, smaller adders on non-critical path

Due Next Lecture (as announced in class)