Lecture 5: Adders

1. Carry Lookahead Adders (CLA)

1.1. The object of lookahead carry is to provide all of the carry bits for an adder at the same time instead of waiting for them to ripple through the adders.

1.2. The addition of two 1-digit inputs \( A \) and \( B \) is said to generate if the addition will always carry, regardless of whether there is an input carry. Thus \( G(A, B) = A \cdot B \)

1.3. The addition of two 1-digit inputs \( A \) and \( B \) is said to propagate if the addition will carry whenever there is an input carry. Thus \( P(A, B) = A + B \), and also \( P(A, B) = A \oplus B \). The XOR version is used normally within a basic full adder circuit; the OR is an alternate option (for a carry lookahead only) which is far simpler in transistor-count terms.

1.4. \( C_{i+1} = G_i + (P_i \cdot C_i) \), carry of adder \( i+1 \)

1.4.1. \( C_1 = G_0 + P_0 \cdot C_0 \)

1.4.2. \( C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0) = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0 \)

1.4.3. \( C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot (G_1 + P_1 \cdot C_1) = G_2 + P_2 \cdot (G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0)) \)

1.4.4. \( C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0))) \)

1.4.5. Note that \( C_0 \) is the only carry that must be known for all of these calculations.

1.4.6. Note that all of these expression can be implemented with two levels of gates, so the longest path could always be two gates for each of the expressions!

1.4.6.1. However, the more bits involved in an AND gate, the longer it will take to propagate the signal through the gate’s circuit because there will be more transistors in series. For more than 7 inputs, a binary tree of AND gates could be faster (and allow uniform 2-bit AND gates) though the path would be longer.

1.5. Three step process of lookahead carry adders

1.5.1. Each bit-adder \( i \), calculates \( P_i \) and \( G_i \), and sends them to the Carry Look Ahead Unit. This will take only one gate delay.

1.5.2. The CLA Unit simultaneously calculates the \( C_i \) for all of its adders. This will take two gate delays.

1.5.2.1. The carry for the next group of adder units is also calculated (\( C_i \) in the 4-bit Carry Lock Ahead in the above image).

1.5.3. Now that the bit-adders know their carries, all of the bit-adders in the group simultaneously calculate their sums. This will take only one gate delay because the carry is no longer calculated and \( A_i \oplus B_i \) was already calculated for \( P_i \).

1.6. “The Carry Look Ahead 4-bit adder can also be used in a higher-level circuit by having each CLA Logic circuit produce a propagate and generate signal to a higher-level CLA Logic circuit. The group propagate (PG) and group generate (GG) for a 4-bit CLA are:

1.6.1. \( PG = P_0 \cdot P_1 \cdot P_2 \cdot P_3 \)

1.6.2. \( GG = G_3 + G_2 \cdot P_3 + G_1 \cdot P_3 \cdot P_2 + G_0 \cdot P_3 \cdot P_2 \cdot P_1 \)

1.6.3. Putting 4 4-bit CLAs together yields four group propagates and four group generates. A Lookahead Carry Unit (LCU) takes these 8 values and uses identical logic to calculate \( C_i \) in the CLAs. The LCU then generates the carry input for each of the 4 CLAs and a fifth equal to \( C_{16} \).

1.6.4. The calculation of the gate delay of a 16-bit adder (using 4 CLAs and 1 LCU) is not as straightforward as the ripple carry adder. Starting at time of zero:

1.6.4.1. calculation of \( P_i \) and \( G_i \) is done at time 1

1.6.4.2. calculation of \( C_i \) is done at time 3

1.6.4.3. calculation of the \( PG \) is done at time 2

1.6.4.4. calculation of the \( GG \) is done at time 3
1.6.4.5. calculation of the inputs for the CLAs from the LCU are done at
   1.6.4.5.1. time 0 for the first CLA
   1.6.4.5.2. time 5 for the second CLA
      1.6.4.5.2.1. Sean notes that this could be time 3 using the $C_e$ from the first LCA, but nothing is gained overall from this, and it disrupts the uniformity of the system.
   1.6.4.5.3. time 5 for the third & fourth CLA
1.6.4.6. calculation of the $S_i$ are done at
   1.6.4.6.1. time 4 for the first CLA
   1.6.4.6.2. time 8 (Sean notes that time 6 is possible) for the second CLA
   1.6.4.6.3. time 8 for the third & fourth CLA
1.6.4.7. calculation of the final carry bit $C_{16}$ is done at time 5
1.6.5. The maximum time is 8 gate delays (for $S_{4,15}$). A standard 16-bit ripple carry adder would take 31 gate delays.”
   (Wikipedia)