REALIZATION OF BINARY OPERATIONS

Lecture 6.
ALU

• Binary operations are realized in the ALU, that can be structured in:
  • serial form:
    • the operations are performed bitwise, from the lowest local value toward to the higher local value,
  • parallel form:
    • all the operations are performed in one step in every local values,
  • mixed form:
    • generally used.

• Every type of arithmetic operations can be realized by addition:
  • substraction,
  • multiplication,
  • division.
ALU

- **ALU:**
Due to the importance of the addition, the time required to add numbers plays an important role in determining the speed of the ALU.

Main types of the 1-bit adders:

- full-adder:

- half-adder:
Adder

• If we add three \((2+CY)\) bits:

\[
\begin{array}{cccccc}
\text{CY} & \text{CY} & \text{CY} & \text{CY} & \text{CY} & \text{CY} \\
0_2 & 0_2 & 0_2 & 1_2 & 1_2 & 1_2 \\
+0_2 & +1_2 & +0_2 & +0_2 & +1_2 & +1_2 \\
\hline
0_2 & +1_2 & +0_2 & +1_0 & +1_1 & +1_1 \\
\end{array}
\]

• full adder: adds binary numbers and accounts for values carried in as well as out,
  • a one-bit full adder adds three one-bit numbers, where \(A_i\) and \(B_i\) are the operands and \(C_{i-1}\) is a bit carried in form the previous less-significant stage,
  • a full adder is usually a component of adders, which adds 8, 16, 32, 64, etc bits binary numbers.
• Truth table of a one-bit full-adder:

<table>
<thead>
<tr>
<th>A_i</th>
<th>B_i</th>
<th>C_i-1</th>
<th>S_i</th>
<th>C_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Adder

- Karnaugh maps:

\[
\begin{array}{c|cc}
S_i & B_i & A_i \\
--- & --- & --- \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
• Logical functions:

• $S_i = A_i \oplus B_i \oplus C_{i-1}$ (XOR)

• $C_i = A_iB_i + C_{i-1}(A_i + B_i)$
• Realization:
  
  \( S_i = A_i \oplus B_i \oplus C_{i-1} \) (XOR)
  
  \( C_i = A_iB_i + C_{i-1}(A_i + B_i) \)
Adder

• If we add two bits:

\[
\begin{array}{cccc}
\text{CY} & 0_2 & \text{CY} & 0_2 \\
1_2 & +1_2 & 2 & +1_2 \\
& 1_2 & 10_2
\end{array}
\]

• full adder: adds two single binary digits,
  • a one-bit half-full adder adds two one-bit numbers, where \( A_i \) and \( B_i \) are the operands,
  • it has two outputs, where the carry represents an overflow in to the next digit.
Adder

- Truth table of a one-bit half-adder:
Adder

• Logical functions and realization:

  • $S_i = A_i \oplus B_i$ (XOR)

  • $C_i = A_iB_i$
Adder

- Complex Adders:
  - to add more digits,

- Serial Adder:
  - the result will be in the operand A,
  - it is slow.

\[ A + B = C \]

\[ \text{storage} \text{ (CU, shifter)} \]
Adder

• Complex Adders:
  • to add more digits,

• Riple-Carry Adder (paralell adder):
  • to add N-bits,
  • in this case, the adder is simple, that allows fast design time,
  • the ripple-carry adder is relative slow, because each full-adder must wait for the carry bit, calculated from the previous adder, this is called the gate-daley (\(\Delta t\)),
  • if the gate delay of a full-adder is \(3\Delta t\), the result will be correct in time: \(n\Delta t\), e.g the total gate delay in a case of addition of two 32 bits number: \(31\times3\Delta t\) (full adders) + \(1\times\Delta t\) (half-adder), the total delay=\(94\Delta t\)
  • if this time is not acceptable, it must to accelerate the addition
• Riple-Carry Adder (parallel adder):
Adder

- Riple-Carry Adder (parallel adder):
  - it is possible to build smaller units – 4-bit Riple-Carry Adder (or carry-propagated adder, CPA), because the carry is propagated serially through each full adder.

• Ripple-Carry Adder (parallel adder):
  • 16-bits CPA:

• Carry-lookahead adder:

• Idea: it is needed to determine the carry before the addition
  • carry-look ahead logic uses the concept of generating and propagating carries,
  • in the case of binary addition, \( A + B \) generates carry, if, and only if both \( A \) and \( B \) are 1.
    • \( G(A,B) = AB \)
  • the addition of two 1-digit inputs \( A \) and \( B \) is said to be propagate, if the addition will carry whenever there is an input carry,
  • in the case of binary addition, \( A + B \) propagates a carry, if and only if at least one of \( A \) or \( B \) is 1
    • \( P(A,B) = A + B \)
Recursive Transfer Training method:

\[ C_i = G_i + (P_i \cdot C_{i-1}) \]

\[ C_0 = G_0 = A_0B_0 \]
\[ C_1 = G_1 + P_1C_0 = A_1B_1 + (A_1 + B_1)A_0B_0 \]
\[ C_2 = G_2 + P_2C_1 = A_2B_2 + (A_2 + B_2)[A_1B_1 + (A_1 + B_1)A_0B_0] = \]
\[ = A_2B_2 + (A_2 + B_2)(A_1B_1 + A_0A_1B_0 + A_0B_0B_1) = \]
\[ = A_2B_2 + A_1A_2B_1 + A_0A_1A_2B_0 + A_0A_2B_0B_1 + A_1B_1B_2 + A_0A_1B_0B_2 + A_0B_0B_1B_2 \]
\[ \ldots \]

It means a complicated 2-levels combinational logical network, and contains the gate delays
• e.g. a standard 16 bit adder would take 46 gate delays, with this method it is just 5 (2+3) gate delays
• Carry-lookahead adder:
• Carry-lookahead adder:
  • 4-bits CLA, e.g.
  • BA: Basic-Adder

Adder

- BCD Adder:
  - Truth Table (see earlier):

  - we have to create an addition if the result is between 9<Res<16,

  - if the result is bigger then 15, it is needed to create the decimal adjust

- \( C=Z_4*Z_8+Z_2*Z_8+C_h \)
• BCD adder for two tetrades:
Shifting

- Multiplication by 2:
  - or with $2^x$

- Division by 2:
  - or with $2^x$
Multiplication

• Multiplication by leftforward shifting, 4x4 Array Multiplier:

• https://www.youtube.com/watch?v=K79wfflmLNo

• https://www.youtube.com/watch?v=I15ZMmrOfnA

• https://www.youtube.com/watch?v=fpnE6UAbtU

• https://www.youtube.com/watch?v=FZGugFqdr60
End of Lecture 6.

Thank you for your attention!