Cairo University
Electronics and Communications Engineering

Computer Arithmetic, Lecture 6:
$1 + 1 = 10$

Hossam A. H. Fahmy

Addition is fundamental

- The subtraction, multiplication, and division are based on the addition.
- The addition is also fundamental in determining the processor cycle time and hence the overall performance.

Many people worked on addition producing algorithms that differ in minute details.

Types of adders

Time: variable time versus fixed (usually worst case) time.

Arrival of inputs: serial versus parallel adders.

Operands: two-operand versus multi-operand adders.

Two-operand parallel addition may use ripple carry, carry skip, carry select, conditional sum, carry lookahead, prefix, ...

Full adder

The sum and carry at a certain bit location are:

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus c_i && \text{(Odd)} \\ c_{i+1} &= a_i b_i + a_i c_i + b_i c_i && \text{(Majority)} \end{aligned}$$

- An incoming carry propagates to c_{i+1} if $p_i = a_i + b_i = 1$.
- A carry is generated (regardless of c_i) if $g_i = a_i b_i = 1$.
- An incoming carry is absorbed (killed) if $k_i = \bar{a}_i \bar{b}_i = 1$.

Note that $c_{i+1} = g_i + p_i c_i = g_i + t_i c_i$ where $t_i = a_i \oplus b_i$.

Rippling the carry

- The simplest parallel addition uses a *ripple carry adder*.
- Since $c_{i+1} = a_i b_i + a_i c_i + b_i c_i$, the generation of the carry takes 2 gate delays.
- The complete adder takes $2n$ gate delays.

Carry skip idea

We know that $c_{i+1} = g_i + p_i c_i$. Hence,

$$\begin{aligned} c_{i+1} &= g_i + p_i(g_{i-1} + p_{i-1}c_{i-1}) \\ &= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} c_{i-2} \end{aligned}$$

A low order carry propagates if all the propagate signals are active.

With the simple grouping $P_{i \leftarrow i-2} = p_i p_{i-1} p_{i-2}$, we have

$$c_{i+1} = c_{i+1}(\text{out of full adder } i+1) + P_i c_{i-2}$$

If the group propagation signal is ready, we *skip* over the group.

Carry skip analysis

- The worst case delay is to ripple through the first and last group and skip over the middle ones.
- Simple designs use a fixed group size of $r-1$.
- Hence the delay is $2 \times 2(r-1) + 2\left(\lceil \frac{n}{r-1} \rceil - 2\right)$.
- Better designs use multiple levels and variable block sizes.

Carry select and conditional sum idea

Instead of waiting for the carry then perform the summation, let us prepare two sums one with the carry assumed as zero and the other with the carry assumed as one.

We can break the long operand into smaller groups with two sums for each group. Once available, the *carry selects* the correct sum via a multiplexer.

The time delay is $5 + 2\left\lceil \log_{r-1}(\lceil n/r \rceil - 1) \right\rceil$

If the group size is reduced to just a pair of positions this is *conditional sum*.

A decimal conditional sum example

The operation is:

	2	6	7	7	4	1	0	0	2	6	9	2	4	3	5	8
+	5	6	0	4	9	7	9	4	1	5	1	7	1	6	4	5
	8	2	8	2	3	8	9	4	4	2	0	9	6	0	0	3

$i \rightarrow$	15	14	13	12	11	10	9	8	
X_i	2	6	7	7	4	1	0	0	
Y_i	5	6	0	4	9	7	9	4	
	08 | 07	13 | 12	08 | 07	12 | 11	14 | 13	09 | 08	10 | 09	05 | 04	t_0
	083	082	082	081	139	138	095	094	t_1
	08282		08281		13895		13894		t_2
	082823895				082823894				t_3
									t_4

$i \rightarrow$	7	6	5	4	3	2	1	0	
X_i	2	6	9	2	4	3	5	8	
Y_i	1	5	1	7	1	6	4	5	
	04 | 03	12 | 11	11 | 10	10 | 09	06 | 05	10 | 09	10 | 09	| 13	t_0
	042	041	110	109	060	059		103	t_1
	04210		04209				06003		t_2
					042096003				t_3
	08282389442096003								t_4

Carry lookahead ideas

Since

$$\begin{aligned} c_{i+1} &= g_i + p_i(g_{i-1} + p_{i-1}c_{i-1}) \\ &= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} c_{i-2} \\ &= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} g_{i-3} \\ &\quad + p_i p_{i-1} p_{i-2} p_{i-3} c_{i-3} \end{aligned}$$

We define two quantities,

a group generate

$$G_{i \leftarrow i-3} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} g_{i-3}$$

and

a group propagate $P_{i \leftarrow i-3} = p_i p_{i-1} p_{i-2} p_{i-3}$.

Carry lookahead, grouping the carries

With the group generate and propagate we get

$$\begin{aligned} c_4 &= G_{3\leftarrow 0} + P_{3\leftarrow 0} c_0 \\ c_8 &= G_{7\leftarrow 4} + P_{7\leftarrow 4} c_4 \\ c_{12} &= G_{11\leftarrow 8} + P_{11\leftarrow 8} c_8 \\ c_{16} &= G_{15\leftarrow 12} + P_{15\leftarrow 12} c_{12} \end{aligned}$$

Notice that each of $G_{3\leftarrow 0}$ and $P_{3\leftarrow 0}$ needs two gate delays after getting g_i and p_i. Then for each carry we need two more gate delays.

Hence, the calculation of c_{16} takes $1+2+4\times 2 = 11$ gate delays.

Carry lookahead, second level

$$\begin{aligned} c_{16} &= G_{15\leftarrow 12} + P_{15\leftarrow 12} c_{12} \\ &= G_{15\leftarrow 12} + P_{15\leftarrow 12} G_{11\leftarrow 8} + P_{15\leftarrow 12} P_{11\leftarrow 8} c_8 \\ &= G_{15\leftarrow 12} + P_{15\leftarrow 12} G_{11\leftarrow 8} + P_{15\leftarrow 12} P_{11\leftarrow 8} G_{7\leftarrow 4} \\ &\quad + P_{15\leftarrow 12} P_{11\leftarrow 8} P_{7\leftarrow 4} c_4 \\ &= G_{15\leftarrow 12} + P_{15\leftarrow 12} G_{11\leftarrow 8} + P_{15\leftarrow 12} P_{11\leftarrow 8} G_{7\leftarrow 4} \\ &\quad + P_{15\leftarrow 12} P_{11\leftarrow 8} P_{7\leftarrow 4} G_{3\leftarrow 0} + P_{15\leftarrow 12} P_{11\leftarrow 8} P_{7\leftarrow 4} P_{3\leftarrow 0} c_0 \end{aligned}$$

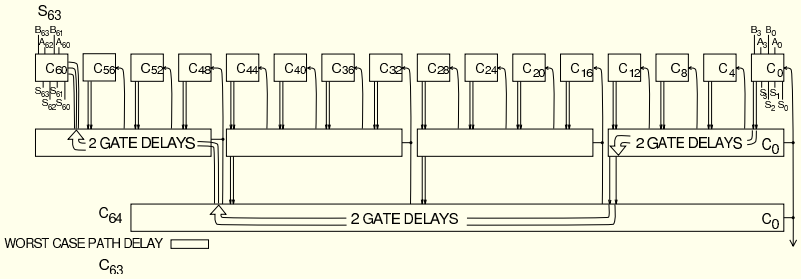
Once more we can define:

$$\begin{aligned} G_{15\leftarrow 0} &= G_{15\leftarrow 12} + P_{15\leftarrow 12} G_{11\leftarrow 8} + P_{15\leftarrow 12} P_{11\leftarrow 8} G_{7\leftarrow 4} \\ &\quad + P_{15\leftarrow 12} P_{11\leftarrow 8} P_{7\leftarrow 4} G_{3\leftarrow 0} \\ P_{15\leftarrow 0} &= P_{15\leftarrow 12} P_{11\leftarrow 8} P_{7\leftarrow 4} P_{3\leftarrow 0} \end{aligned}$$

Now, the calculation of c_{16} takes $1+2+2+2 = 7$ gate delays.

Note that we take the same time to get c_{12} and c_8.

Time delay in a 64 bit adder



$$
\begin{aligned}
c_{48} &= G_{47\leftarrow 0} + P_{47\leftarrow 0}c_0 \\
c_{60} &= G_{59\leftarrow 48} + P_{59\leftarrow 48}c_{48} \\
c_{63} &= G_{62\leftarrow 60} + P_{62\leftarrow 60}c_{60} \\
s_{63} &= t_{63} \oplus c_{63}
\end{aligned}
$$

Hence the delay is $2 \times (2\lceil \log_r n \rceil - 1) + 1 + 1 = 4 \times \lceil \log_r n \rceil$.

Canonic and prefix adders

- The *canonic adder* has a specific circuit to generate the carry into each bit location.
- c_{i+1} is due to a propagation from c_0 *or* a propagation from a generation at position 1 *or* a propagation from a generation at position 2 *or* ...
- Hence the delay is that of an *AND* tree to detect the propagation followed by an *OR* tree to combine the result.
- The total delay is $2\lceil \log_r n \rceil + 1 + 1$. (The two trees plus the formation of the initial p and g plus the final bit sum.)
- The *prefix adder* is similar assuming $r = 2$.

Ling adder

We notice that $g_i = p_i g_i$ and hence

$$
\begin{aligned}
G_{i\leftarrow i-3} &= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} g_{i-3} \\
&= p_i \left(g_i + g_{i-1} + p_{i-1} g_{i-2} + p_{i-1} p_{i-2} g_{i-3} \right)
\end{aligned}
$$

But,

$$
\begin{aligned}
c_{i+1} &= G_{i\leftarrow i-3} + P_{i\leftarrow i-3} c_{i-3} \\
&= p_i h_{i+1}
\end{aligned}
$$

which yields

$$
\begin{aligned}
s_{i+1} &= t_{i+1} \oplus (p_i h_{i+1}), \\
&= t_{i+1}(\bar{p}_i + \bar{h}_{i+1}) + \bar{t}_{i+1} p_i h_{i+1}, \\
&= \bar{h}_{i+1} t_{i+1} + h_{i+1}(t_{i+1} \oplus p_i).
\end{aligned}
$$

We moved one gate delay away from the critical path.

In his original work, Ling also used the wired logic capability of ECL to enhance the speed.

Hybrid adders

- Modern adders do not follow a "pure" strategy but use a combination of techniques.
- The "best" adder is not clearly defined. Those with lower gate delays usually have larger areas and complicated wiring.

Lookahead

- Multi-operand addition
- Carry save adders
- Multiplication