ECE 645: Lecture 1

Basic Adders and Counters

Implementation of Adders in FPGAs
Required Reading

Behrooz Parhami,
Computer Arithmetic: Algorithms and Hardware Design

Chapter 5, Basic Addition and Counting, Sections 5.1-5.5, pp. 75-85.
Required Reading

Spartan-3 Generation FPGA User Guide
http://www.xilinx.com/support/documentation/spartan-3_user_guides.htm

Chapter 9, Using Carry and Arithmetic Logic
### Half-adder

\[ x + y = (c \ s)_2 \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>c</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Half-adder**

**Alternative implementations (1)**

(a) $c = xy$

$s = x \oplus y$

![Diagram of AND/XOR half-adder](image)

(a) AND/XOR half-adder.

(b) $c = \overline{x} + y$

$s = xy + xy$

![Diagram of NOR-gate half-adder](image)

(b) NOR-gate half-adder.
Half-adder
Alternative implementations (2)

c)
\[ \bar{c} = \overline{xy} \]
\[ s = x\bar{c} + y\bar{c} = \bar{x}\bar{c} \cdot \bar{y}\bar{c} \]

(c) NAND-gate half-adder with complemented carry.
### Full-adder

The full-adder (FA) is a digital electronic circuit that performs the addition of two bits, along with a carry-in, to produce a sum and a carry-out.

The operation of the full-adder can be represented by the following equation:

\[ x + y + c_{in} = (c_{out} \quad s)_2 \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>c_{in}</th>
<th>c_{out}</th>
<th>s</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Full-adder**

**Alternative implementations (2)**

\[ c_{out} = xy + xc_{in} + yc_{in} \]

\[ s = x \oplus y \oplus c_{in} = xyc_{in} + \overline{xy}c_{in} + \overline{xy}c_{in} + xy\overline{c}_{in} \]

(b) Built as an AND-OR circuit.
Full-adder
Alternative implementations (1)

b) \[ s = (x \oplus y) \oplus c_{in} \]
\[ c_{out} = xy + c_{in} (x \oplus y) \]

(a) Built of half-adders.
**Full-adder**

**Alternative implementations (3)**

c)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>c(_{\text{out}})</th>
<th>s = c(_{\text{in}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c(_{\text{in}})</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>c(_{\text{in}})</td>
<td>c(_{\text{in}})</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>c(_{\text{in}})</td>
<td>c(_{\text{in}})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>c(_{\text{in}})</td>
</tr>
</tbody>
</table>
Full-adder

Alternative implementations (4)

Implementation used to generate fast carry logic in Xilinx FPGAs

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>c_{out}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>c_{in}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>c_{in}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>y</td>
</tr>
</tbody>
</table>

\[ p = x \oplus y \]
\[ g = y \]
\[ s = p \oplus c_{in} = x \oplus y \oplus c_{in} \]
CMOS transmission gate and its use in a 2-to-1 mux.
(b) Four-bit ripple-carry adder.
\[ T_{\text{ripple-add}} = T_{\text{FA}}(x, y \rightarrow c_{\text{out}}) + (k - 2)T_{\text{FA}}(c_{\text{in}} \rightarrow c_{\text{out}}) + T_{\text{FA}}(c_{\text{in}} \rightarrow s) \]

Fig. 5.5 Critical path in a \( k \)-bit ripple-carry adder.
Latency of a \( k \)-bit ripple-carry adder

\[
d_{\text{RCA}}(k) = d_{\text{FA}}(x, y \rightarrow c_{\text{out}}) + \\
+ (k-2) \cdot d_{\text{FA}}(c_{\text{in}} \rightarrow c_{\text{out}}) + \\
+ d_{\text{FA}}(c_{\text{in}} \rightarrow s)
\]

Latency = \( a \cdot k + b \)
Fig. 5.4  The layout of a 4-bit ripple-carry adder in CMOS implementation [Puck94].
### Unsigned addition vs. signed addition

#### Machine

<table>
<thead>
<tr>
<th>weight</th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>carry</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0 0 0 1 0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Y</td>
<td>1 0 0 0 0 0 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= S</td>
<td>1 0 0 1 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Programmer

<table>
<thead>
<tr>
<th></th>
<th>Unsigned mind</th>
<th>Signed mind</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Diagram

```
 FA  FA  FA  FA  FA  FA  FA  FA
    |    |    |    |    |    |    |
    s7  s6  s5  s4  s3  s2  s1  s0
```

```
 FA  FA  FA  FA  FA  FA  FA  FA
    |    |    |    |    |    |    |
    x7  x6  x5  x4  x3  x2  x1  x0
```

```
 FA  FA  FA  FA  FA  FA  FA  FA
    |    |    |    |    |    |    |
    c8  c7  c6  c5  c4  c3  c2  c1
```
Out of range flags

**Carry flag - C**

out-of-range for unsigned numbers

\[
C = 1 \text{ if } \text{result} > \text{MAX}_\text{UNSIGNED} \text{ or result} < 0 \\
0 \text{ otherwise}
\]

where \( \text{MAX}_\text{UNSIGNED} = 2^8-1 \) for 8-bit operands
\( 2^{16}-1 \) for 16-bit operands

**Overflow flag - V**

out-of-range for signed numbers

\[
V = 1 \text{ if } \text{result} > \text{MAX}_\text{SIGNED} \text{ or result} < \text{MIN}_\text{SIGNED} \\
0 \text{ otherwise}
\]

where \( \text{MAX}_\text{SIGNED} = 2^7-1 \) for 8-bit operands
\( 2^{15}-1 \) for 16-bit operands
\( \text{MIN}_\text{SIGNED} = -2^7 \) for 8-bit operands
\( -2^{15} \) for 16-bit operands
Overflow for signed numbers

Indication of overflow

Positive + Positive = Negative

Negative + Negative = Positive

Formulas

Overflow\textsubscript{2\text{'} s complement} = x_{k-1} y_{k-1} s_{k-1} + x_{k-1} y_{k-1} s_{k-1}
Addition of Signed and Unsigned Numbers

C=1 and V=0

\[
\begin{array}{c|ccc}
8 & 4 & 2 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\hline
1 & 1 & 0 & 0 & 0 \\
\end{array}
\quad
\begin{array}{c|ccc}
-8 & 4 & 2 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\hline
1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

=8

C=0 and V=1

\[
\begin{array}{c|ccc}
8 & 4 & 2 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 \\
\end{array}
\quad
\begin{array}{c|ccc}
-8 & 4 & 2 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 \\
\end{array}
\]

=9

Addition of Signed and Unsigned Numbers

C=0 and V=0

<table>
<thead>
<tr>
<th>8 4 2 1</th>
<th>-8 4 2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1 1 0</td>
</tr>
</tbody>
</table>

C=1 and V=1

<table>
<thead>
<tr>
<th>8 4 2 1</th>
<th>-8 4 2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>1 0 1 1 1</td>
<td>1 0 1 1 1</td>
</tr>
</tbody>
</table>

Two’s complement representation of signed integers
Two’s-complement adder with provisions for detecting conditions and exceptions.
Overflow for signed numbers (1)

Indication of overflow

Positive
+ Positive
= Negative

Negative
+ Negative
= Positive

Formulas

\[
\text{Overflow}_{2\text{'s complement}} = x_{k-1} y_{k-1} s_{k-1} + x_{k-1} y_{k-1} s_{k-1} =
\]

\[
= c_k \oplus c_{k-1}
\]
Overflow for signed numbers (2)

<table>
<thead>
<tr>
<th>$x_{k-1}$</th>
<th>$y_{k-1}$</th>
<th>$c_{k-1}$</th>
<th>$c_k$</th>
<th>$s_{k-1}$</th>
<th>overflow</th>
<th>$c_k \oplus c_{k-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<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>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| 1         | 1         | 0         | 1     | 1         | 0        | 0                    |
Implementation of Adders in FPGAs
# Xilinx FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>220 nm</td>
<td>Spartan II</td>
<td>Virtex</td>
</tr>
<tr>
<td>120/150 nm</td>
<td></td>
<td>Virtex II, II Pro</td>
</tr>
<tr>
<td>90 nm</td>
<td>Spartan 3</td>
<td>Virtex 4</td>
</tr>
<tr>
<td>65 nm</td>
<td></td>
<td>Virtex 5</td>
</tr>
<tr>
<td>45 nm</td>
<td>Spartan 6</td>
<td></td>
</tr>
<tr>
<td>40 nm</td>
<td></td>
<td>Virtex 6</td>
</tr>
<tr>
<td>28 nm</td>
<td>Artix 7</td>
<td>Virtex 7</td>
</tr>
</tbody>
</table>
## Altera FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>Mid-range</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>130 nm</td>
<td>Cyclone</td>
<td></td>
<td>Stratix</td>
</tr>
<tr>
<td>90 nm</td>
<td>Cyclone II</td>
<td></td>
<td>Stratix II</td>
</tr>
<tr>
<td>65 nm</td>
<td>Cyclone III</td>
<td>Arria I</td>
<td>Stratix III</td>
</tr>
<tr>
<td>40 nm</td>
<td>Cyclone IV</td>
<td>Arria II</td>
<td>Stratix IV</td>
</tr>
<tr>
<td>28 nm</td>
<td>Cyclone V</td>
<td>Arria V</td>
<td>Stratix V</td>
</tr>
</tbody>
</table>
General structure of an FPGA

Programmable interconnect

Programmable logic blocks
Xilinx Spartan 3 FPGAs

Configurable logic block (CLB)

<table>
<thead>
<tr>
<th>Slice</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic cell</td>
<td>Logic cell</td>
</tr>
<tr>
<td>Logic cell</td>
<td>Logic cell</td>
</tr>
<tr>
<td>Logic cell</td>
<td>Logic cell</td>
</tr>
</tbody>
</table>
CLB Slice Structure

- Each slice contains two sets of the following:
  - Four-input LUT
    - Any 4-input logic function,
    - or 16-bit x 1 sync RAM (SLICEM only)
    - or 16-bit shift register (SLICEM only)
  - Carry & Control
    - Fast arithmetic logic
    - Multiplier logic
    - Multiplexer logic
  - Storage element
    - Latch or flip-flop
    - Set and reset
    - True or inverted inputs
    - Sync. or async. control
Carry & Control Logic
### Carry & Control Logic in Xilinx FPGAs

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>COUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CIN</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>CIN</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>y</td>
</tr>
</tbody>
</table>

\[
\text{Propagate} = x \oplus y \\
\text{Generate} = y \\
\text{Sum} = \text{Propagate} \oplus \text{CIN} = x \oplus y \oplus \text{CIN}
\]
Carry & Control Logic in Spartan 3 FPGAs

Hardwired (fast) logic
Simplified View of Spartan-3 FPGA Carry and Arithmetic Logic in One Logic Cell
Simplified View of Carry Logic in One Spartan 3 Slice
\[ T_{\text{ripple-add}} = T_{\text{FA}(x,y \rightarrow c_{\text{out}})} + (k - 2) \times T_{\text{FA}(c_{\text{in}} \rightarrow c_{\text{out}})} + T_{\text{FA}(c_{\text{in}} \rightarrow s)} \]

**Fig. 5.5** Critical path in a \( k \)-bit ripple-carry adder.
Critical Path for an Adder Implemented Using Xilinx Spartan 3 FPGAs
### Number and Length of Carry Chains for Spartan 3 FPGAs

<table>
<thead>
<tr>
<th>Device</th>
<th>Number of Carry Chains</th>
<th>Bits per Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50</td>
<td>24</td>
<td>64</td>
</tr>
<tr>
<td>XC3S200</td>
<td>40</td>
<td>96</td>
</tr>
<tr>
<td>XC3S400</td>
<td>56</td>
<td>128</td>
</tr>
<tr>
<td>XC3S1000</td>
<td>80</td>
<td>192</td>
</tr>
<tr>
<td>XC3S1500</td>
<td>104</td>
<td>256</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>128</td>
<td>320</td>
</tr>
<tr>
<td>XC3S4000</td>
<td>144</td>
<td>384</td>
</tr>
<tr>
<td>XC3S5000</td>
<td>160</td>
<td>416</td>
</tr>
</tbody>
</table>
Bottom Operand Input to Carry Out Delay $T_{OPCYF}$

0.9 ns for Spartan 3
Carry Propagation Delay

$t_{\text{BYP}}$

0.2 ns for Spartan 3
Carry Input to Top Sum Combinational Output Delay \( T_{\text{CINY}} \)

1.2 ns for Spartan 3
Critical Path Delays and Maximum Clock Frequencies (taking into account surrounding registers)

- 8 bits: 3.0 ns or 333 MHz
- 16 bits: 3.8 ns or 263 MHz
- 32 bits: 5.4 ns or 185 MHz
- 64 bits: 8.6 ns or 116 MHz
# Major Differences between Xilinx Families

<table>
<thead>
<tr>
<th>Feature</th>
<th>Spartan 3</th>
<th>Virtex 4</th>
<th>Virtex 5, Virtex 6, Spartan 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look-Up Tables</td>
<td>4-input</td>
<td></td>
<td>6-input</td>
</tr>
<tr>
<td>Number of CLB slices per CLB</td>
<td>4</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Number of LUTs per CLB slice</td>
<td>2</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Number of adder stages per CLB slice</td>
<td>2</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
Altera Cyclone III
Logic Element (LE) – Normal Mode
Altera Cyclone III
Logic Element (LE) – Arithmetic Mode
Altera Stratix III, Stratix IV
Adaptive Logic Modules (ALM) – Normal Mode
Altera Stratix III, Stratix IV
Adaptive Logic Modules (ALM) – Arithmetic Mode
Addition of a Constant
Addition of a constant (1)

\[ x_{k-1} x_{k-2} \cdots x_1 x_0 \quad \text{variable} \]
\[ + \quad y_{k-1} y_{k-2} \cdots y_1 y_0 \quad \text{constant} \]
\[ \quad \overline{s_{k-1} s_{k-2} \cdots s_1 s_0} \]

\[ x_{k-1} x_{k-2} \cdots x_{h+1} x_h x_{h-1} \cdots x_0 \quad \text{variable} \]
\[ + \quad y_{k-1} y_{k-2} \cdots y_{h+1} 1 0 \cdots 0 \quad \text{constant} \]
\[ \quad \overline{s_{k-1} s_{k-2} \cdots s_{h+1} x_h x_{h-1} \cdots x_0} \]
Addition of a constant (2)

If

\[ y_i = 0 \quad \text{Half-adder (HA)} \]
\[ y_i = 1 \quad \text{Modified half-adder (MHA)} \]
Modified half-adder

\[ x + y + 1 = (c \ s)_{2} \]
Incrementer

Decrementer
Fast three-stage up counter.
Bit-Serial & Digit-Serial Adders
(a) Bit-serial adder.
Bit-serial adder
Digit-serial adder

Inputs:
- $x_i$
- $y_i$
- $d$

Outputs:
- $c_{i+1}$
- $s_i$
- $d$
- $c_0$
- start
- clk
Asynchronous Adders
Possible solutions to the carry propagate problem

1. Detect the end of propagation rather than wait for the worst-case time

2. Speed-up propagation via
   • look-ahead
   • carry skip
   • carry select, etc

3. Limit carry propagation to within a small number of bits

4. Eliminate carry propagation through the redundant number representation
Bit #: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

\[
\begin{array}{cccccccccccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
\hline
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

Carry chains and their lengths:
- Carry chain 4: length 6
- Carry chain 3: length 2
- Carry chain 2: length 1
- Carry chain 1: length 1
- Carry chain 0: length 0
- Carry chain In: length 0
Analysis of carry propagation

Probability of carry generation = \( \frac{1}{4} \) \( (x_i y_i = 11) \)

Probability of carry propagation = \( \frac{1}{2} \) \( (x_i y_i = 01 \text{ or } 10) \)

Probability of carry anihilation = \( \frac{1}{2} \) \( (x_i y_i = 00 \text{ or } 11) \)

\[
\begin{array}{ccccccccccc}
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
& & & & & & & & & & \\
\end{array}
\]

Probability of carry propagating from position \( i \) to position \( j \)

= \( \frac{1}{2} \frac{j - i - 1}{2} \cdot \frac{1}{2} \) = \( \frac{1}{2} \frac{j - i}{2} \)

\( \frac{1}{2} \) probability of propagation \( \frac{1}{2} \) probability of anihilation
Expected length of the carry chain that starts at position $i$ (1)

Expected length$(i, k) =$

$$
\sum_{j = i + 1}^{k - 1} (j - i) \left( \frac{1}{2} \right)^{j - i} + (k - i) \left( \frac{1}{2} \right)^{k - 1 - i}
$$

Length of the carry chain

Probability of the given length

Distance till the end of adder

Probability of propagation till the end of adder
Expected length of the carry chain that starts at position $i$ (2)

Expected length($i$, $k$) =

$$2 - 2^{-(k-i-1)}$$

For $i << k$

Expected length of the carry propagation is $\approx 2$