How to translate binary numbers to BCD

The old fashioned way

Pascal Pieper

Pascal.Pieper@dfki.de

Deutsches Forschungsinstitut
für Künstliche Intelligenz

Cyber-Physical Systems

Requirements

These slides should help with...

- Sheet 1, Exercise 3: ALU Seven Segment Display
- Understanding the Paper DEVELOPING LARGE BINARY TO BCD CONVERSION STRUCTURES by *M. Benedek*

BCD

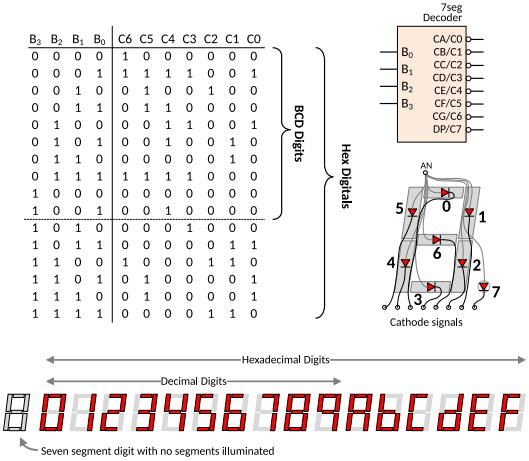
Problem

- Imagine, you are in the 70s
  - Flared hip trousers are 'in'
  - No processing power or memory to spare
  - This new thing called "LED" comes out
- How to iterface with humans?

Seven Segment Displays - Binary Coded Digit

- 4 bit per decimal digit: More space than can be displayed
  - Up to 9_{10} (1001_2)
- \+ Easy to interface with decimal displays
- – How to calculate with wider (binary) integers?

Sevensegment Displays



B_3	B_2	B_1	B_0	C6	C5	C4	C3	C2	C1	C0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0	0	0	0
0	1	0	0	0	0	1	1	0	0	1
0	1	0	1	0	0	1	0	0	1	0
0	1	1	0	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	1	1
1	1	0	0	1	0	0	0	1	1	0
1	1	0	1	0	1	0	0	0	0	1
1	1	1	0	0	1	0	0	0	0	1
1	1	1	1	0	0	0	0	1	1	0

https://www.realdigital.org/doc/586fb4c3326dcd493a5774b2a6050f41

Static Conversion

First Idea

Show $1100_2 = \mathsf{C}_{16} = 12_{10}$

- If number $> 9_{10}$ (1001_2) it can't be displayed with one digit
- Perhaps add something: Trigger "carry over" by adding difference of bases?

→ Max number 1001_2 is 4 bit wide, can be represented by one hexadecimal digit (`base16`)

- Difference `base16` - `base10` $= 6 \ldots$
- So:

$$C_{16(=1100_2)} + 6_{16(=0110_2)} = 12_{16(=0001\,0010_2)} \overset{\text{Interp.}}{=} 1\,2_{\mathsf{BCD}} \tag{1}$$

→ Nice! Why? And does this work for bigger digits?

Static Conversion

First Idea: Why?

$$X \pmod{16} \equiv \begin{cases} (X+6) \pmod{10} & \text{if } X > 9 \\ X \pmod{10} & \text{else} \end{cases} \tag{2}$$

- Maths!
- Does that work simply for every digit?

Static Conversion

Naive Approach

$$
\begin{aligned}
175_{10} = AF_{16} = (A+6)\cdot 16^1 + (F+6) &= (1\,0000_2)\cdot 16^1 + (1\,0101_2) \\
&= (1\,0000\,0000) + (1\,0101_2) \\
&= (1\,\underline{0001}\,0101_2) \\
&\overset{\text{Interp.}}{=} 1\,1\,5_{10}\ ??? \qquad (3)
\end{aligned}
$$

- What is missing? $1\,0\underline{11}1\,0101_2$ would be correct

Static Conversion - Main Idea

Recursive Approach

- Use the fact (from MSB to LSB), that with $b = [b_n b_{n-1} \ldots]$

$$2 \cdot \left[2 \cdot \left[2 \cdot \left[2 \cdot [b_n]\, b_{n-1} \right] + b_{n-2} \right] + b_{n-3} \right] + b_{n-\ldots} \Bigg] = b \tag{4}$$

- And *if* $x > 9 \iff \frac{x}{2} \geq 5$ (and $6 = 3 \cdot 2$)
- Idea: For every step (binary digit), translate to BCD!

Static Conversion - Example

AF_{16}	1010 1111

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\ 0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\ 0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000		

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{\text{BCD}} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{\text{BCD}} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!
+11 0011		add 3s in both quads

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!
+11 0011		add 3s in both quads
1011 1010	1	

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!
+11 0011		add 3s in both quads
1011 1010	1	... and shift left

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{\text{BCD}} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!
+11 0011		add 3s in both quads
1011 1010	1	... and shift left
1 0111 0101		

[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 1\,0_{10}$

Static Conversion - Example

AF_{16}	1010 1111	directly align first three
101	0 1111	$101_2 \geq 5$!
+11		Add 3
1000	0 1111	... and shift
1 0000	1111	no overflow, just shift left[1]
10 0001	111	no overflow, just shift left
100 0011	11	no overflow, just shift left
1000 0111	1	two overflows!
+11 0011		add 3s in both quads
1011 1010	1	... and shift left
1 0111 0101		Done! $1\,0111\,1010_{BCD} = 175_{10}$

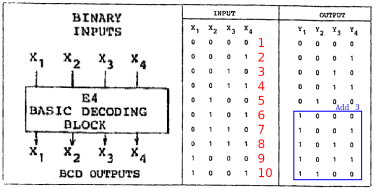
[1] left side already (in fact, always) correct for left bits! $1010_2 = 0001\,0000_{BCD} = 10_{10}$

Conclusion

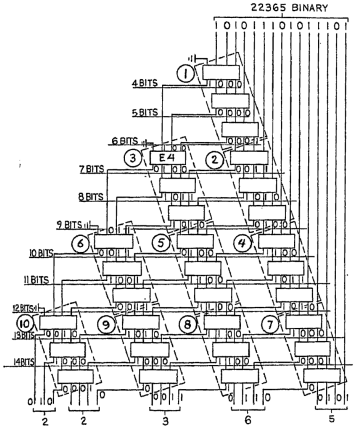
So..

- We can translate any width binary to BCD now
- Implementation in combinatorial logic can be done with lookup tables (LUTs)
  - Quad-aligned overflow-check-and-add can be indirectly implemented
  - Shift left is just a connection to the next bit for all items

TABLE 2. LOOK-UP TABLE CONTENT



INPUT					OUTPUT				
X_1	X_2	X_3	X_4		Y_1	Y_2	Y_3	Y_4	
0	0	0	0	1	0	0	0	0	
0	0	0	1	2	0	0	0	1	
0	0	1	0	3	0	0	1	0	
0	0	1	1	4	0	0	1	1	
0	1	0	0	5	0	1	0	0	
0	1	0	1	6	1	0	0	0	Add 3
0	1	1	0	7	1	0	0	1	
0	1	1	1	8	1	0	1	0	
1	0	0	0	9	1	0	1	1	
1	0	0	1	10	1	1	0	0	



BINARY TO BCD DECODING TREE WITH E4 DECODERS.
THE DOTTED OUTLINES REPRESENT E4 DECODERS.

INTERCONNECTION OF BASIC BUILDING BLOCKS
FIGURE 1