Outline

❍ Last time:
- ➔ Where to from here?
- ➔ The responsibilities of a technologist
- ➔ CS150: once over lightly
- ➔ Relationship to other courses in EECS

❍ This lecture:
- ➔ Error Detection & Correction
- ➔ State Assignment in Asynchronous

Error Detection & Correction

- ❍ As transistors & wires become smaller, the number of electrons used to represent a stored 1 or 0 decreases
- ❍ The probability that a cosmic ray, alpha particle, local electric field (crosstalk) or some other disturbance will change a stored value (or even the value on a wire) increases
- ❍ Two aspects:
  - ➔ Error detection (e.g. parity)
  - ➔ Error correction (e.g. Hamming Codes)
    - ✷ 2-bit detect, 1-bit correct

Hamming Codes

- Most common types of error-correcting codes used in RAM
- Based on work of R. W. Hamming
- k parity bits are added to an n-bit word, forming a new n+k bit word
- The positions numbered with powers of two are reserved for the parity bits
- Can be used with words of any length
- Example: 8-bit data word 11000100

1	2	3	4	5	6	7	8	9	10	11	12
P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

CS150 Newton 14.2.3

Hamming Codes

- Calculate parity bits as follows:

P_1= xor(3,5,7,9,11) = $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$
P_2= xor(3,6,7,10,11) = 0
P_4= xor(5,6,7,12) = 1
P_8= xor(9,10,11,12) = 1

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	1	0	0	1	0	1	0	0

- When bits are read from memory, compute check bits:

C_1= xor(1,3,5,7,9,11)
C_2= xor(2,3,6,7,10,11)
C_4= xor(4,5,6,7,12)
C_8= xor(8,9,10,11,12)

CS150 Newton 14.2.4

Hamming Codes

- $C=C_8C_4C_2C_1$=0000 indicates no error has occurred
- Examples:

1	2	3	4	5	6	7	8	9	10	11	12	
0	0	1	1	1	0	0	1	0	1	0	0	no error
1	0	1	1	1	0	0	1	0	1	0	0	error in bit 1
0	0	1	1	0	0	0	1	0	1	0	0	error in bit 5

C_8	C_4	C_2	C_1	
0	0	0	0	no error
0	0	0	1	error bit 1
0	1	0	1	error bit 5

CS150 Newton 14.2.5

Hamming Codes

- For n data bits and k check bits, $n+k \leq 2^k-1$
- Grouping of bits for parity generation can be observed from listing of binary numbers:

	B_1	B_2	B_3
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

B_1=1 for (1,3,5,7)

B_2=1 for (2,3,6,7)

B_3=1 for (4,5,6,7)

CS150 Newton 14.2.6

Hamming Codes

- ❍ Add P_{13} as an overall parity bit: 001110010100P_{13}
- ❍ Detect/Correct Rules as follows:

If C=0, P=0	no error
If C≠ 0, P=1	single error, correctable
If C ≠ 0, P=0	double error, uncorrectable
If C=0, P=1	error occurred in P_{13}

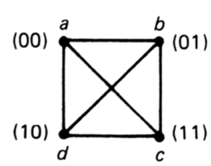
- ❍ Can also have far more sophisticated schemes
- ❍ Covered in detail in communication & information theory courses

CS150 Newton 14.2.7

Race-Free State Assignment

- ❍ A race-free state assignment for any 4-row table can be found using three state variables
- ❍ Example:

	00	01	11	10
a	a	c	d	a
b	a	d	b	b
c	d	c	b	a
d	d	d	d	b

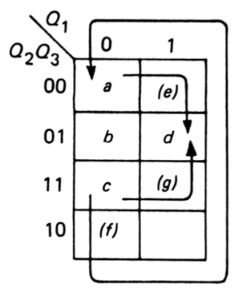


column 00	$b \to a,\ c \to d$
column 01	$a \to c,\ b \to d$
column 11	$a \to d,\ c \to b$
column 10	$c \to a,\ d \to b$

CS150 Newton 14.2.8

Race-Free State Assignment

- The assignment below is a universal state-assignment map that will work for any 4-row table
- The example can then be expanded as shown, with the additional rows added



$Q_1Q_2Q_3$		00	01	11	10
000	a	a	f	e	a
001	b	a	d	b	b
011	c	g	c	b	f
101	d	d	d	d	b
100	e	–	–	d	–
010	f	–	c	–	a
111	g	d	–	–	–

Race-Free State Assignment

- Another universal assignment for 4-row tables is shown below (M. Mano, *Logic and Computer Design*, Prentice Hall)
- Using this approach, the expanded example from before becomes as shown below right
- This approach faster than previous general approach above. Why?

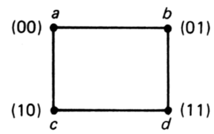
Q_2Q_3 \ Q_1	0	1
00	a_1	b_2
01	d_1	c_2
11	b_1	a_2
10	c_1	d_2

$Q_1Q_2Q_3$		00	01	11	10
000	a_1	a_1	c_1	d_1	a_1
111	a_2	a_2	c_2	d_2	a_2
011	b_1	a_2	d_1	b_1	b_1
100	b_2	a_1	d_2	b_2	b_2
010	c_1	d_2	c_1	b_1	a_1
101	c_2	d_1	c_2	b_2	a_2
001	d_1	d_1	d_1	d_1	b_1
110	d_2	d_2	d_2	d_2	b_2

Race-Free State Assignment

- Don't cares can be used to make race-free assignment
- In example below-left, we need:

  col_{00} d→a; col_{01} a→b, c→d; col_{11} b→c; col_{10} a→c, b→d



- Looks like we need extra rows, but use the don't cares:

	00	01	11	10
a	a	b	-	c
b	b	b	c	d
c	-	d	c	c
d	a	d	d	d

		00	01	11	10
00	a	a	b	c	c
01	b	b	b	a	d
10	c	a	d	c	c
11	d	c	d	d	d

CS150 Newton 14.2.11

Shared-Row Assignments

	00	01	11	10
a	1	2	5	4
b	7	2	3	10
c	1	8	3	4
d	7	8	5	4
e	1	9	5	6
f	11	8	3	6

column 00: $e, c \to a$ $\quad d \to b$
column 01: $a \to b$ $\quad c, f \to d$
column 11: $b, f \to c$ $\quad a, d \to e$
column 10: $a, c \to d$ $\quad e \to f$

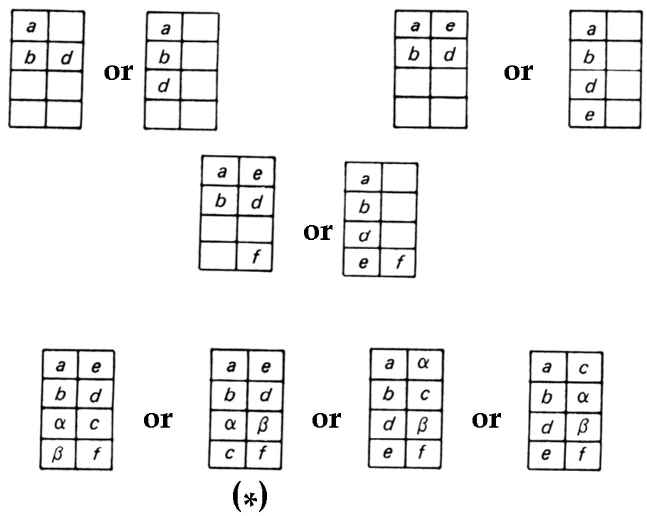
Consider the example above. We need at least three state variables and maybe even four.

e→a, c→a; e→c→a; c→e→a
or any of these going through an intermediate state

CS150 Newton 14.2.12

Shared-Row Assignment

❍ How do we get there? Trial and error...



Shared-Row Assignment

❍ Result of using second option from previous slide:

		00	01	11	10
000	a	a	b	e	c
001	b	b	b	α	b
010	c	a	f	c	α
101	d	b	d	e	d
100	e	a	e	e	f
110	f	f	β	c	f
011	α	-	-	c	β
111	β	-	d	-	d