ECE 3401 Lecture 13

Sequential Circuits (III)

Datapath & Control Unit (Chpt. 5)

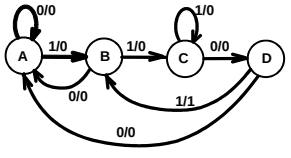
State Assignment

- Each of the *m* states must be assigned a unique code
- Minimum number of bits required is *n* such that
  $n \geq \lceil \log_2 m \rceil$
  where $\lceil x \rceil$ is the smallest integer $\geq x$
- There are 2^n - m unused states

State Assignment – Example

- How many assignments of codes with a minimum number of bits (2) for four states?
  - $4 \times 3 \times 2 \times 1 = 24$
- Does code assignment make a difference in cost?

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1



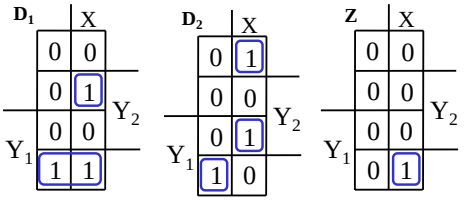
State Assignment 1– Example (Contd.)

- Assignment 1: A = 0 0, B = 0 1, C = 1 0, D = 1 1
- The resulting coded state table:
- Assume D flip-flops, state Y_1Y_2, next state D_1D_2

Present State (Y_1Y_2)	Next State (D_1D_2) x = 0	Next State (D_1D_2) x = 1	Output (Z) x = 0	Output (Z) x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1

Find Output and Flip-Flop Input Equations/ Optimization: Assignment 1

- **Assume D flip-flops, state Y_1Y_2, next state D_1D_2, output Z**



- **Performing two-level optimization:**

$D_1 = \overline{Y_1}Y_2 + X\overline{Y_1}\overline{Y_2}$
$D_2 = XY_1\overline{Y_2} + X\overline{Y_1}Y_2 + XY_1Y_2$
$Z = XY_1\overline{Y_2}$ Gate Input Cost = 22

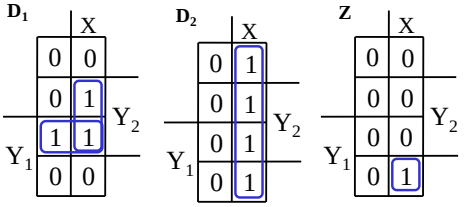
State Assignment 2 – Example (continued)

- Assignment 2: A = 0 0, B = 0 1, C = 1 1, D = 1 0
- The resulting coded state table:

Present State (Y_1Y_2)	Next State (D_1D_2) x = 0	Next State (D_1D_2) x = 1	Output (Z) x = 0	Output (Z) x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

Find Output and Flip-Flop Input Equations: Optimization: Assignment 2

- Assume D flip-flops



- Performing two-level optimization:

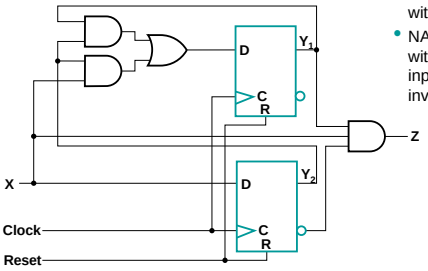
$D_1 = Y_1Y_2 + XY_2$ Gate Input Cost = 9

$D_2 = X$

$Z = XY_1\overline{Y}_2$

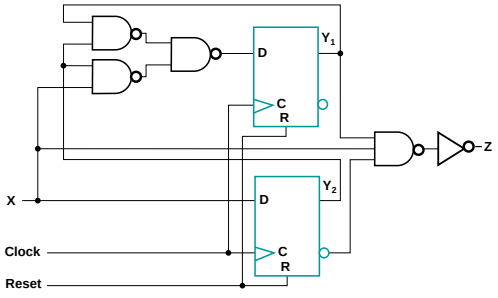
Map Technology

- **Initial Circuit:**



- Library:
  - D Flip-flops with Reset
  - NAND gates with up to 4 inputs and inverters

Mapped Circuit - Final Result

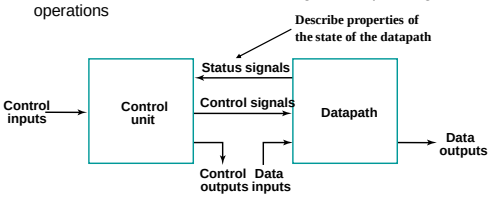


Microprogramming Overview (Chpt. 5)

- Datapath and control
- Microoperations
- Sequencing and control

Datapath and Control

- Datapath - performs data transfer and processing operations
- Control Unit - Determines the enabling and sequencing of the operations



- The control unit receives:
  - External control inputs
  - Status signals
- The control unit sends:
  - Control signals
  - Control outputs

Overview

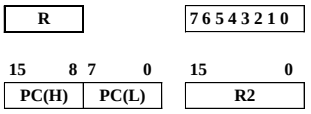
- Datapath and control
- Microoperations
  - Register transfer operations
  - Microoperations - arithmetic, logic, and shift
  - Register cell design
  - Serial transfers and microoperations
- Sequencing and control

Register Transfer Operations

- Register Transfer Operations – the movement and processing of data stored in registers
- Three basic components:
  - A set of registers (operands)
  - Transfer operations
  - Control of operations
- Elementary operations -- called *microoperations*
  - load, count, shift, add, bitwise "OR", etc.

Register Notation

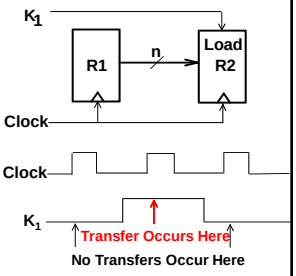
- Letters and numbers – register (e.g. R2, PC, IR)
- Parentheses ( ) – range of register bits (e.g. R1(1), PC(7:0), AR(L))



- Arrow (←) – data transfer (ex. R1 ← R2, PC(L) ← R0)
- Brackets [ ] – Specifies a memory address (ex. R0 ← M[AR], R3 ← M[PC])
- Comma – separates parallel operations

Conditional Transfer

- If (K_1 =1) then (R2 ← R1) ⇔ K_1: (R2 ← R1)
  where K_1 is a control expression specifying a conditional execution of the microoperation.



Microoperations

- Logical groupings:
  - Transfer - move data from one set of registers to another
  - Arithmetic - perform arithmetic on data in registers
  - Logic - manipulate data or use bitwise logical operations
  - Shift - shift data in registers

Arithmetic operations
+ Addition
– Subtraction
* Multiplication
/ Division

Logical operations
∨ Logical OR
∧ Logical AND
⊕ Logical Exclusive OR
¯ Not

Example Microoperations

- R1← R1 + R2
  - Add the content of R1 to the content of R2 and place the result in R1.
- PC ← R1 * R6
- R1 ← R1 ⊕ R2
- (K1 + K2): R1 ← R1 ∨ R3
  - On condition K1 OR K2, the content of R1 is Logic bitwise Ored with the content of R3 and the result placed in R1.
  - NOTE: "+" (as in K_1 + K_2) means "OR." In R1 ← R1 + R2, + means "plus."

Arithmetic Microoperations

Symbolic Designation	Description
R0 ← R1 + R2	Addition
R0 ← $\overline{R1}$	Ones Complement
R0 ← $\overline{R1}$ + 1	Two's Complement
R0 ← R2 + $\overline{R1}$ + 1	R2 minus R1 (2's Comp)
R1 ← R1 + 1	Increment (count up)
R1 ← R1 – 1	Decrement (count down)

- Any register may be specified for source 1, source 2, or destination.
- These simple microoperations operate on the whole word

Logical Microoperations

Symbolic Designation	Description
R0 ← $\overline{R1}$	Bitwise NOT
R0 ← R1 ∨ R2	Bitwise OR (sets bits)
R0 ← R1 ∧ R2	Bitwise AND (clears bits)
R0 ← R1 ⊕ R2	Bitwise EXOR (complements bits)

Shift Microoperations

- Let R2 = 11001001

Symbolic Designation	Description	R1 content
R1 ← sl R2	Shift Left	10010010
R1 ← sr R2	Shift Right	01100100

- **Note: These shifts "zero fill". Sometimes a separate flip-flop is used to provide the data shifted in, or to "catch" the data shifted out.**
- **Other shifts are possible (rotates, arithmetic)**