Encoders and Servo Systems

Department of Mechanical Engineering
Lecturer: Jia-Yush Yen
3/16/2006

Measuring Angular Motion

- **Torque**
  - **Torque sensor**
- **Angular velocity**
  - **Tachometer**
  - **Encoder**
- **Angular displacement**
  - **Encoder**
  - **Synchro**
  - **Resolver**

Various Forms of Encoders

http://www.opticalencoder.com/summaries/cp-300_summary.html

Encoder Servo Systems 3

What is Coding

- **(Not coding theorem)**
- **Digitally representing angular positions**

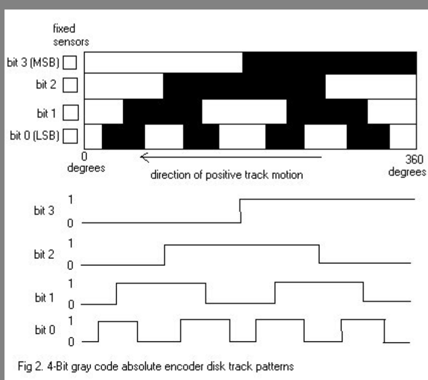


Fig 2. 4-Bit gray code absolute encoder disk track patterns

http://mechatronics.mech.northwestern.edu/design_ref/sensors/encoders.html

Encoder Servo Systems 4

Absolute Encoder Coding

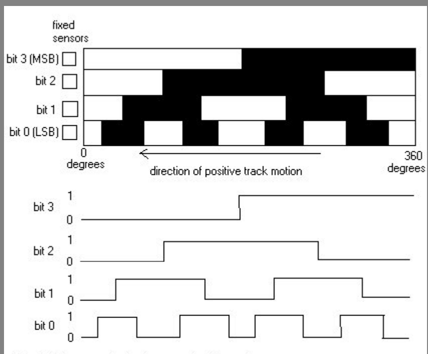


Fig 2. 4-Bit gray code absolute encoder disk track patterns

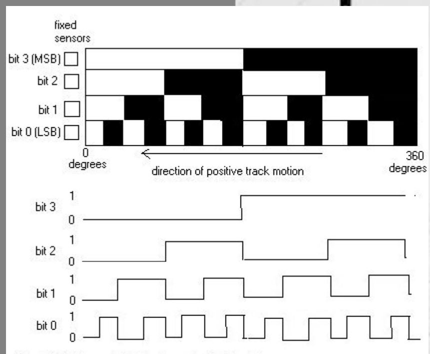
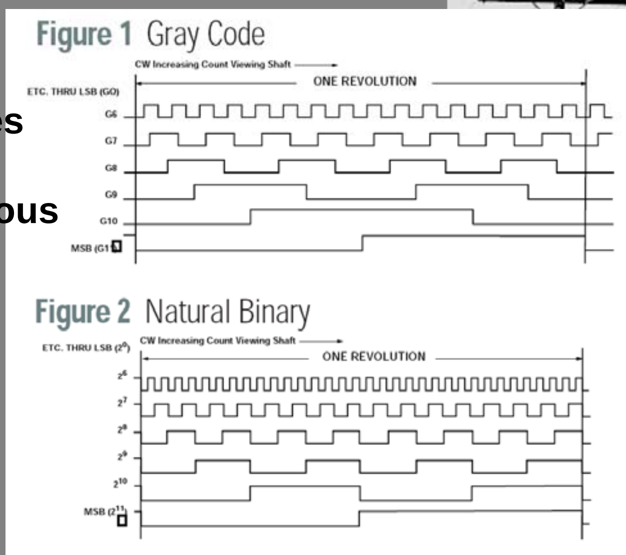


Fig 3. 4-Bit binary code absolute encoder disk track patterns

http://mechatronics.mech.northwestern.edu/design_ref/sensors/encoders.html

Absolute Encoder (Gray/binary)

- **Gray code:**
  - **Guarantees no simultaneous two bits change**



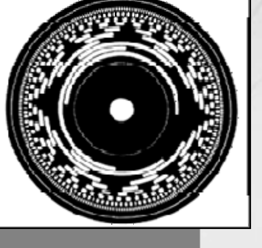
http://www.beiied.com/absolute_encoders.html

Gray vs. Binary Coding (4 bits)

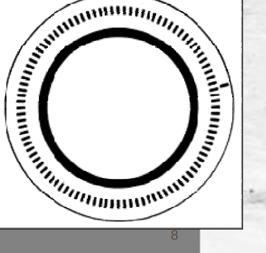
Decimal code	Rotation range (deg.)	Binary code	Gray code
0	0-22.5	0000	0000
1	22.5-45	0001	0001
2	45-67.5	0010	0011
3	67.5-90	0011	0010
4	90-112.5	0100	0110
5	112.5-135	0101	0111
6	135-157.5	0110	0101
7	15.75-180	0111	0100
8	180-202.5	1000	1100
9	202.5-225	1001	1101
10	225-247.5	1010	1111
11	247.5-270	1011	1110
12	270-292.5	1100	1010
13	292.5-315	1101	1011
14	315-337.5	1110	1001
15	337.5-360	1111	1000

Absolute/Incremental Encoder

- **Absolute Encoder**



- **Incremental Encoder**



SW Analysis

- **Strength**
  - **Absolute position**
  - **Direct read out**
- **Weakness**
  - **Requires very high precision**
  - **Hard to manufacture**
  - **Easily corrupted data**
  - **Long parallel reading**

Encoder Servo Systems 9

Incremental Encoder

- **Often called: A B pulse**

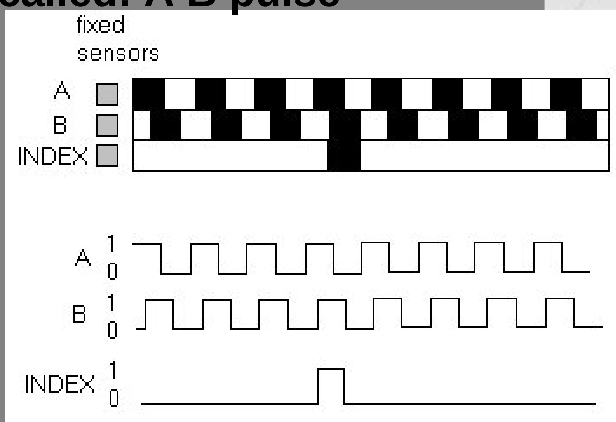


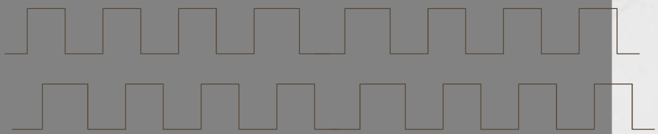
Fig 5. Incremental encoder disk track patterns

Encoder Servo Systems 10

To Detect Rotating Direction

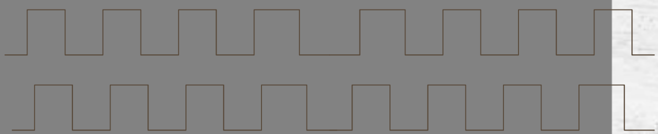
- 90° offset for direction
- Forward direction

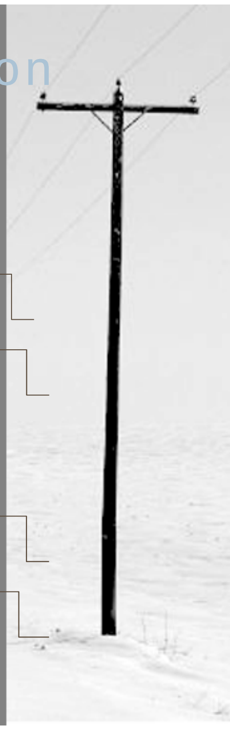
A
B



- When there is a reverse

A
B

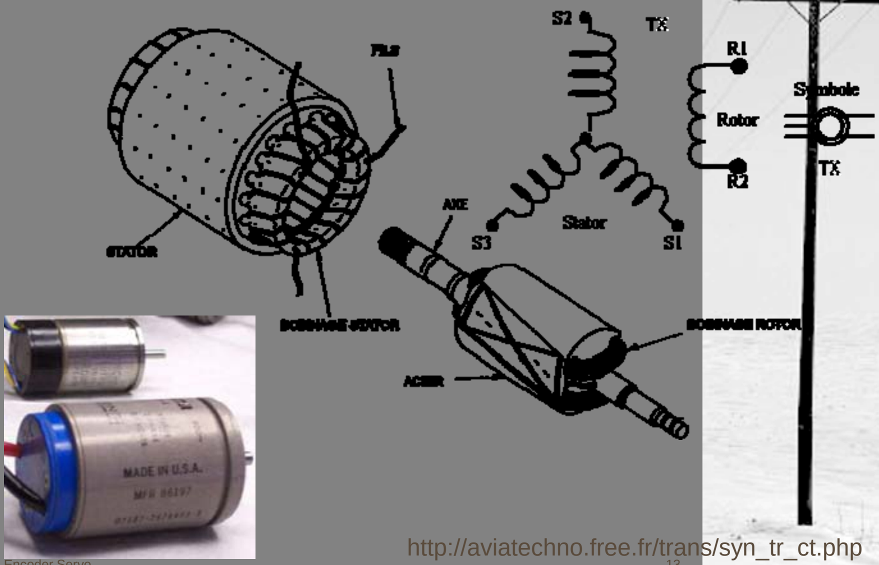




SW Analysis

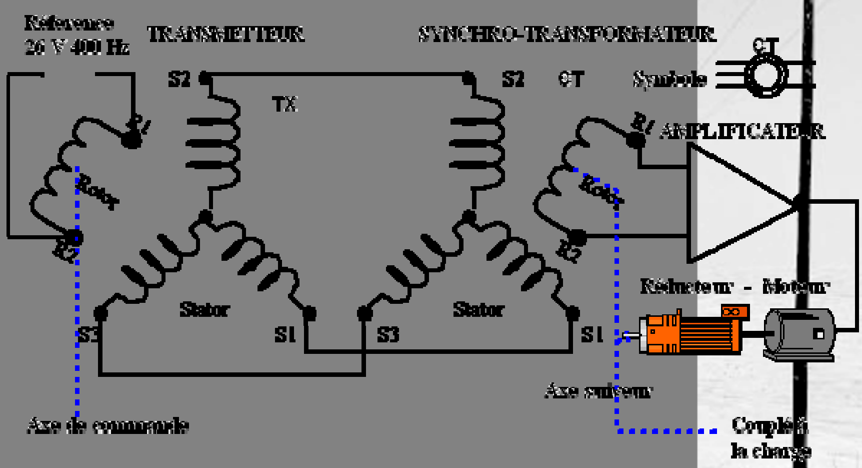
- Strength
  - Easy to manufacture
  - No limit on the rotation angle
  - High resolution
  - Sub-division for enhanced resolution
- Weakness
  - Require decoder
  - Serial (incremental) measurement

Synchro



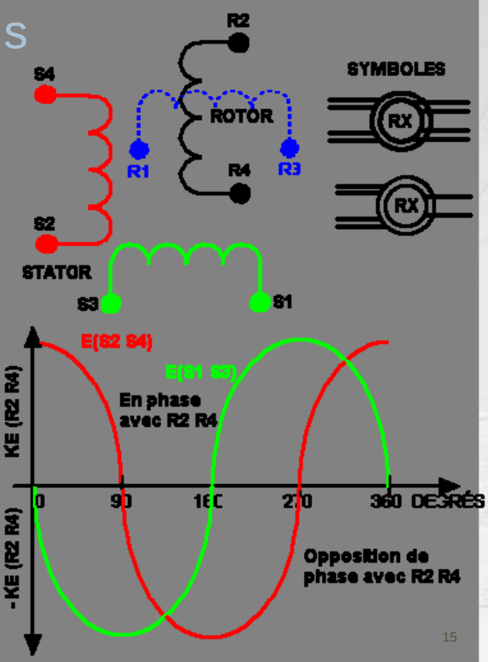
http://aviatechno.free.fr/trans/syn_tr_ct.php

Transmission Synchro



http://aviatechno.free.fr/trans/syn_tr_ct.php

Resolvers



Resolvers

Single speed brushless resolvers

Low Cost Brushless Pancake Resolvers

COMBINATIONAL LOGIC

Encoder Servo Systems 17

Basic Combinational Logic

- **Based on Boolean algebra and Karnaugh maps**
- **Use logic variables to represent signals**

Encoder Servo Systems 18

- **Example:**
  - **Take the air-conditioning control as an example:**
    A = temp too high,
    B = humidity too high,
    C = air-conditioning on,
  - **One can then say**

    C = A + B

  - **For air-conditioner control logic**

Boolean Logic

- **There is a complete logic**
- **Upon the field of 0 and 1**
  (all the elements are 0 and 1)

- **“+” stands for “or”**
  - **A + 0 = A**
  - **A + A = A**
  - **A + 1 = 1**
  - **A + (-A) = 1**
- **“.” stands for “and”**
  - **A • 1 = A**
  - **A • A = A**
  - **A • 0 = 0**
  - **A • (-A) = 0**

Algebraic Laws

- **Communitative**
  - **A + B = B + A**
  - **AB = BA**
- **Associative**
  - **A + (B + C) = (A + B) + C**
  - **A(BC) = (AB)C**
- **Distributive**
  - **A (B + C) = AB + AC**

Encoder Servo Systems 21

Basic Laws

- Negative of negative becomes positive
  - $A = \overline{\overline{A}}$
  - $\overline{AB} = \overline{A}\,\overline{B}$
  - $\overline{A+B} = \overline{A} + \overline{B}$
- Conjugate
  - For all Boolean equation
    - Change “or” to “and”
    - Change “and” to “or”
    - Switch “0” and “I”
    - We get the conjugate equation

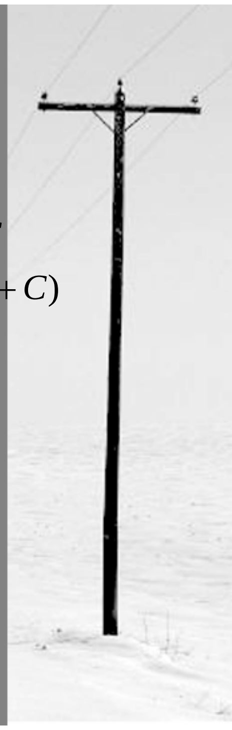
Encoder Servo Systems 22

- **Example**

$$A + 0 = A$$
$$A \cdot 1 = A$$

- **Example**

$$A(B + C) = AB + AC$$
$$A + BC = (A + B)(A + C)$$



Controller Design

- **"Sum of products" method**
- **Product of two variables: Y=AB**
  - **Possible combinations**

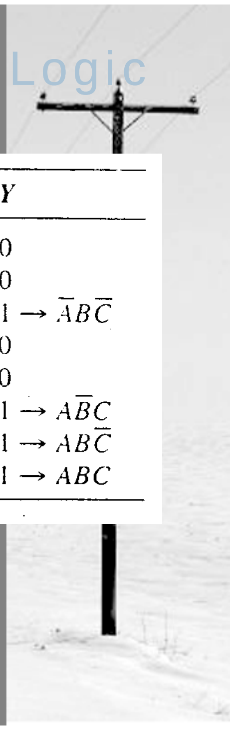
A	B	Fundamental product
0	0	$\bar{A}\bar{B}$
0	1	$\bar{A}B$
1	0	$A\bar{B}$
1	1	AB

- **Product of three variables: Y = ABC**
  - **Possible combinations are**

A	B	C	Fundamental product
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	$\overline{A}\,\overline{B}C$
0	1	0	$\overline{A}B\overline{C}$
0	1	1	$\overline{A}BC$
1	0	0	$A\overline{B}\,\overline{C}$
1	0	1	$A\overline{B}C$
1	1	0	$AB\overline{C}$
1	1	1	ABC

- **And product of four variables, …**
  - **Possible combinations are**

To Establish The Required Logic



- **Example**
  - **The required "truth table"**

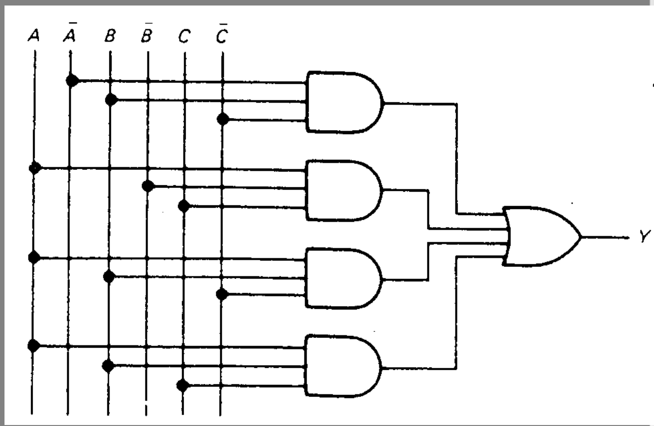
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	$1 \rightarrow \overline{A}B\overline{C}$
0	1	1	0
1	0	0	0
1	0	1	$1 \rightarrow A\overline{B}C$
1	1	0	$1 \rightarrow AB\overline{C}$
1	1	1	$1 \rightarrow ABC$

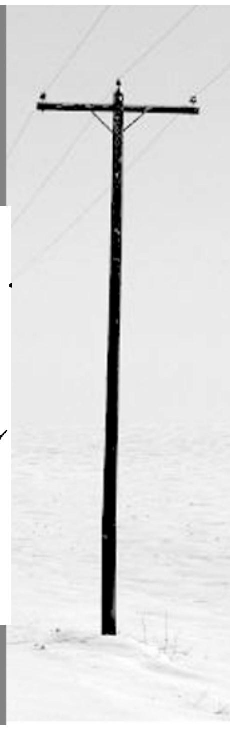
  - **The required actions becomes**

$$Y = \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C} + ABC$$

  - **Sum up all the actions!!**

- **The circuit becomes**





Simplifying

- **“Gate count” is a very important parameter defining logic circuit efficiency**
- **Simplifying reduces “Gate Count”**

Simplify Methods

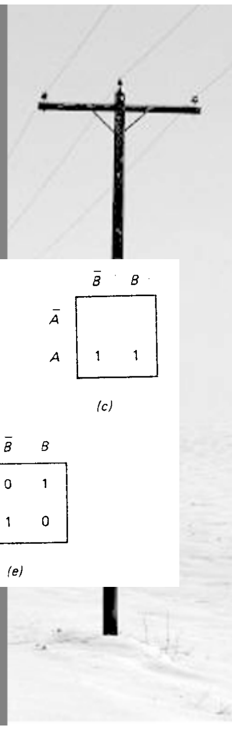
- **Factoring**
- **Example**

$$Y = A\overline{B} + AB$$
$$= A(\overline{B} + B)$$
$$= A \cdot 1 = A$$

- **Example**

$$Y = AB + AC + BD + CD$$
$$= A(B + C) + D(B + C)$$
$$= (A + D)(B + C)$$

Karnaugh Map

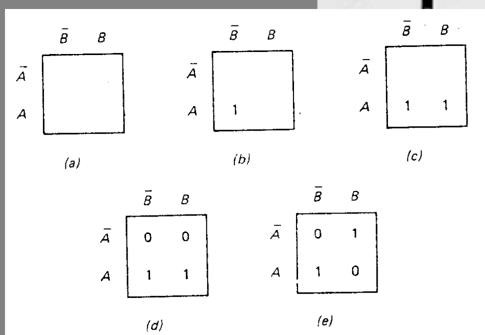


- **A effective tool to simplify a Boolean logic**
- **Construct a Karnaugh Map**
- **Two variables**

A	B	Y
0	0	0
0	1	0
1	0	1
1	1	1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- **Mark all the "1's"**



- **Three variables**
- **Truth table**

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

	$\bar{C}$	C
$\bar{A}\bar{B}$		
$\bar{A}B$		
AB		
$A\bar{B}$		

(a)

	$\bar{C}$	C
$\bar{A}\bar{B}$		
$\bar{A}B$	1	
AB	1	1
$A\bar{B}$		

(b)

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	1	0
AB	1	1
$A\bar{B}$	0	0

(c)

- **Four variables**
- **Truth table**

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB				
$A\bar{B}$				

(a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$			1	1
AB				1
$A\bar{B}$				

(b)

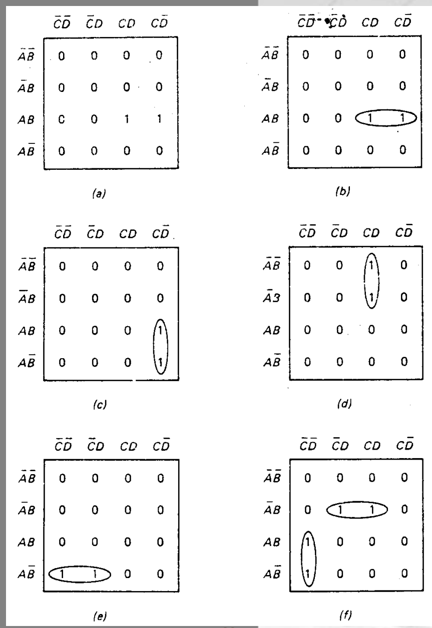
	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	0	1	1
AB	0	0	0	1
$A\bar{B}$	0	0	0	0

(c)

- **Find features**
- **Pair**

$$Y = ABCD + ABC\overline{D}$$
$$= ABC(D + \overline{D})$$
$$= ABC$$

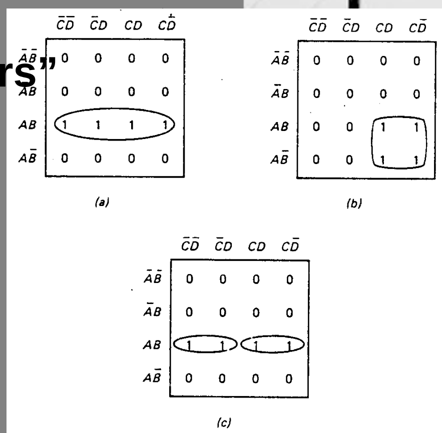
- **Pairs are eliminated**



Encoder Servo Systems 33

- **Quads can be eliminated**
- **“Quads” are the same as two “pairs”**

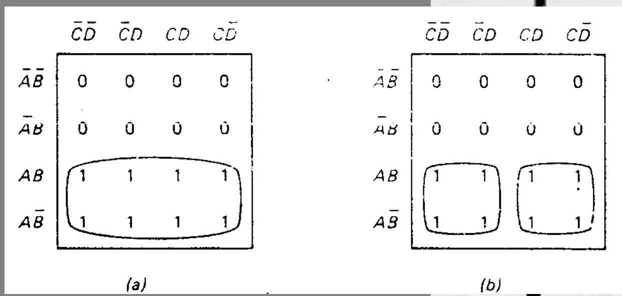
$$Y = AB\overline{C} + ABC$$
$$= AB(\overline{C} + C)$$
$$= AB$$



Encoder Servo Systems 34

- **Octal:**
- **Like two "quads"**

$$Y = A\overline{C} + AC$$
$$= A(\overline{C} + C)$$
$$= A$$



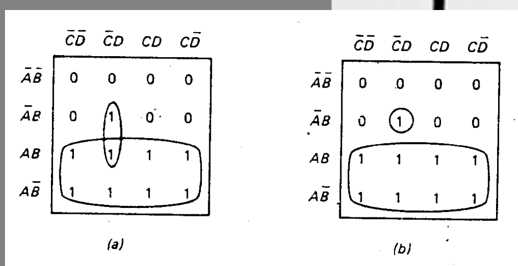
- **Circle all the "1's"**

- **Note:**
- **Overlapping features**
  - **The equation in (b)**

$$Y = A + \overline{A}B\overline{C}D$$

  - **But for (a)**

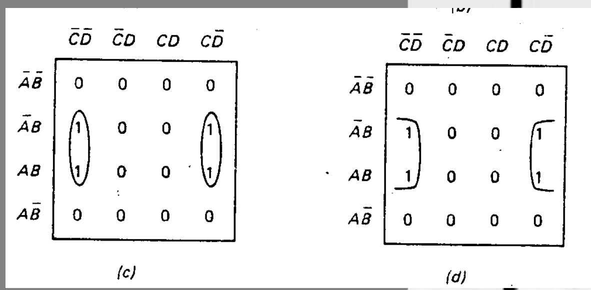
$$Y = A + B\overline{C}D$$



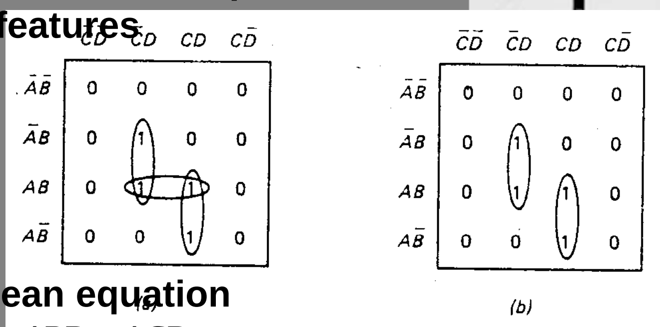
- **Roll over:**
  - **The equation in (C)**

    $Y = B\overline{C}\,\overline{D} + BC\overline{D}$
  - **But for (d)**

    $Y = B\overline{D}$



- **Redundant features**
  - **There is no need to repeat the covered features**



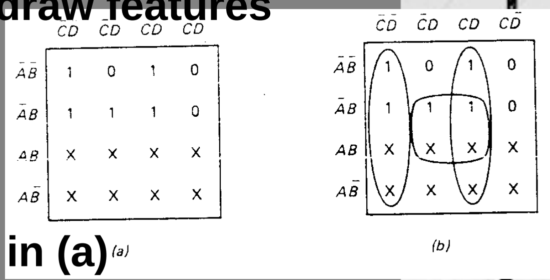
  - **The Boolean equation**

$$Y = B\overline{C}D + \overline{A}BD + ACD$$
$$= B\overline{C}D + ACD$$

- **Don’t care states**
  - **Help further simplification**

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	X
1	0	0	1	X
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

- **Two ways to draw features**



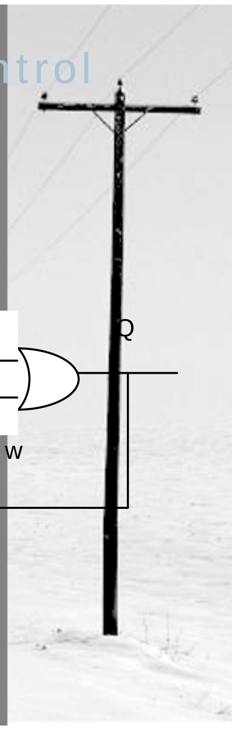
- **The equation in (a)**

$$Y = A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + A\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD$$

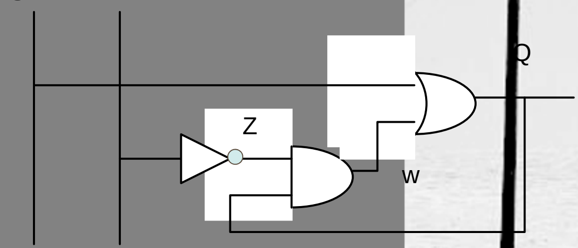
- **The equation in (b)**

$$Y = BD + \overline{C}\overline{D} + CD$$

Dynamic Sequential Logic Control



- **Memory in Logic Circuits**
- **S-R Flip-flop:**



- **$Q^+ = S+\overline{R}Q$**
  - **「or in "C",」**
  - **q = s || (!r $$ q)**

- **The truth table**

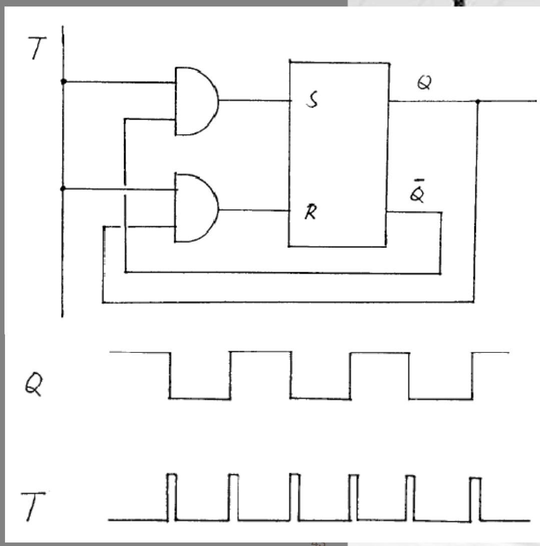
S	R	$\overline{R}$	Q	Q^+
0	0	1	0	0
1	0	1	0	1
0	0	1	1	1
0	1	0	1	0
0	0	1	0	0

  - **Q goes to 1 when S = 1, and remain 1 even if S goes to 0,**
  - **Q goes to zero as R = 1, and remain 0 even if R goes to 0.**

- **T Flip-flop (Triggered flip-flop):**
  - **From S-R flip-flop and take**

$$S = T\overline{Q} \quad R = TQ$$

  - **The output**

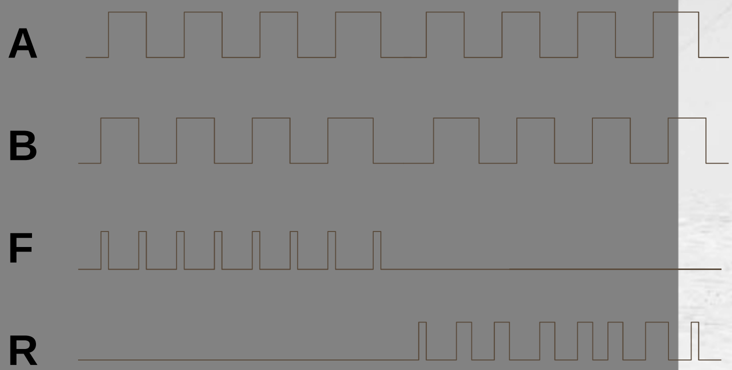


Design A Decoder

- **Two modes of operation**
  - **Clocked circuit**
  - **Asynchronous circuit**
    - **Pulsed mode**
      - **Systems in which the input is removed before any feedback signals have had a chance to change**
    - **Fundamental (Level) mode**
      - **Systems in which the input must remain without change until all of the changes in internal and feedback variables are complete.**

Quadrature Decoder

- **The desired output**



Primitive Flow Table

- **A special form of next state table**
  - **Assume enough time must elapse after an input variable change for the system to reach a stable state**
  - **Because the input remain applied to the system, the calculation of the next state must take both the current state and the input into account.**
  - **The combination of the current and the input6 is called the "total state"**
  - **A state is <u>stable</u> if the next state is equal to the current state**

- The primitive flow table is started by establishing a column of reach possible input combination and a column for each of the outputs

- **Define states**
  - For forward motion, from S_1 proceed to the second quadrant will cause B: $0 \rightarrow 1$. The immediate change in input will cause a transition in column $1 \rightarrow 2$ and reach the unstable state S_2.
  - The outputs associated with the states are given in FR column
  - For forward motion $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4$, Reverse $S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_8$

The quadrant

B \ A	0	1
0	1	4
1	2	3

Quadrant	1	2	3	4	FR
AB	00	01	11	10	
	(S_1)	S_2		S_5	00
	S_8	(S_2)	S_3		10
		S_7	(S_3)	S_4	00
	S1		S_9	(S_4)	10
	S_8		S_6	(S_5)	01
		S_7	(S_6)	S_4	00
	S_1	(S_7)	S_9		01
	(S_8)	S_7		S_4	11
		S_2	(S_9)	S_5	11

Encoder Servo Systems 47

- **Now go back and fill in the empty states.**
  - Moving in the reverse direction from S_2 requires a stable state with an output of 11, which is satisfied by state S_8.
  - Existing stable states can be used for these transitions until S_4.
  - Then there is no stable state already in the column for quadrant 3 that has the proper output, so the new state S_9 must be created.
- **The primitive flow table is complete when all "legal" transitions have been filled in. The remaining transitions will become don't-cares.**

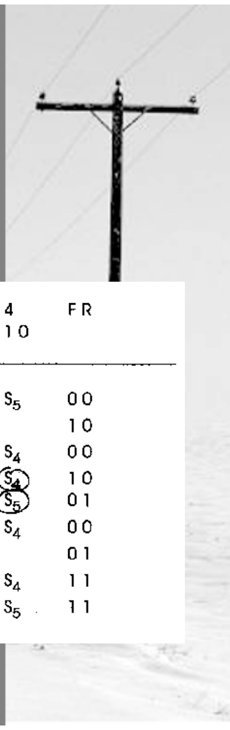
Quadrant	1	2	3	4	FR
AB	00	01	11	10	
	(S_1)	S_2		S_5	00
	S_8	(S_2)	S_3		10
		S_7	(S_3)	S_4	00
	S1		S_9	(S_4)	10
	S_8		S_6	(S_5)	01
		S_7	(S_6)	S_4	00
	S_1	(S_7)	S_9		01
	(S_8)	S_7		S_4	11
		S_2	(S_9)	S_5	11

Encoder Servo Systems 48

Flow Table Reduction and Merger

- **The primitive table needs reduction**
- **Stage 1:**
  - **Combine rows for which the stable states appear in the same column, and the outputs associated with each of the candidate rows must be the same and their next states are the same or equivalent.**
    - **Rows 3 and 6**

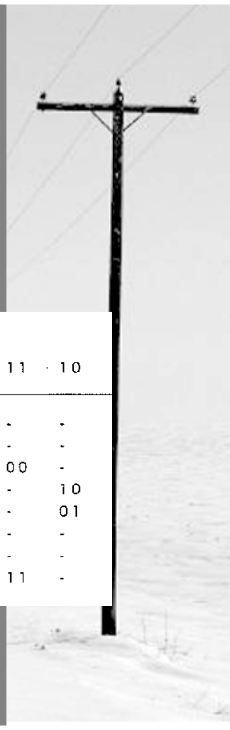
Quadrant AB	1 00	2 01	3 11	4 10	FR
	(S_1)	S_2		S_5	00
	S_8	(S_2)	S_3		10
		S_7	(S_3)	S_4	00
	S1		S_9	(S_4)	10
	S_8		S_6	(S_5)	01
		S_7	(S_6)	S_4	00
	S_1	(S_7)	S_9		01
	(S_8)	S_7		S_4	11
		S_2	(S_9)	S_5	11



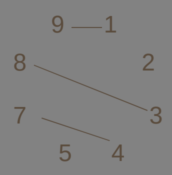
- **Stage 2:**
  - **Merge rows and allow several stable states to coexist on a single row.**

	Next State AB 00	01	11	10	FR AB 00	01	11	10
1	(S_1)	S_2	-	S_5	00	-	-	-
2	S_8	(S_2)	S_3	-	-	10	-	-
3	-	S_7	(S_3)	S_4	-	-	00	-
4	S_1	-	S_9	(S_4)	-	-	-	10
5	S_8	-	S_3	(S_5)	-	-	-	01
7	S_1	(S_7)	S_9	-	-	01	-	-
8	(S_8)	S_7	-	S_4	11	-	-	-
9	-	S_2	(S_9)	S_5	-	-	11	-

  - **The ones filled in are the ones associated with stable states; the rest have been left as don't cares for the moment.**



- **Merging rows:**
  - **Look for rows that do not have stable states in the same column, where the transitions in the candidate rows must either be to the same state, or one of the transitions must be a don't care**



	Next State AB 00	01	11	10	FR AB 00	01	11	10
1	(S_1)	S_2	-	S_5	00	-	-	-
2	S_8	(S_2)	S_3	-	-	10	-	-
3	-	S_7	(S_3)	S_4	-	-	00	-
4	S_1	-	S_9	(S_4)	-	-	-	10
5	S_8	-	S_3	(S_5)	-	-	-	01
7	S_1	(S_7)	S_9	-	-	01	-	-
8	(S_8)	S_7	-	S_4	11	-	-	-
9	-	S_2	(S_9)	S_5	-	-	11	-

- **There is not unique choice of mergers**

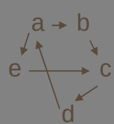
	Next state AB 00	01	11	10	FR AB 00	01	11	10
a	(1)	2	(9)	5	00	--	11	--
b	8	(2)	3	-	--	10	--	--
c	(8)	2	(3)	-	--	10	--	--
d	1	(7)	9	(4)	11	--	00	--
e	8	-	3	(5)	--	--	--	01

- **State assignment:**
  - **State assignment is not arbitrary:**
    - **Example: 5 states ⇒ $\lceil \log_2 5 \rceil = 3$**
    - **We need three bits to represent 5 states**

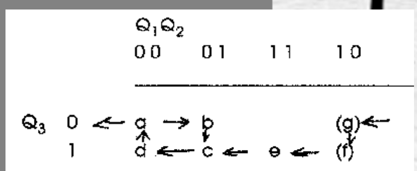
$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ AB 00	01	11	10
a 000	000	001	000	100
b 001	010	001	010	-
c 010	010	011	010	011
d 011	000	011	000	011
e 100	010	-	010	100

    - **There will be trouble going from 011 to 000, might end up in 010 (stable).**
      - **This is called “race”.**

- **To avoid “race”**
  - **Use all transitions are made with only one state change**
- **Diagram of all possible transitions**



- **Use Karnaugh map**
  - **Make all transitions adjacent to each other. (Notice, augment states can be used to aid in the selection**



- **New assignment of next state table**

$Q_1Q_2Q_3$	Next state AB 00	01	11	10	FR AB 00	01	11	10
a 000	(000)	010	(000)	100	(00)	10	(11)	01
b 010	011	(010)	011	-	1-	(10)	-0	--
c 011	(011)	001	(011)	001	(11)	01	(00)	10
d 001	000	(001)	000	(001)	00	(01)	11	(10)
e 111	011	-	011	(111)	-1	--	0-	(01)
(f) 101	-	-	-	111	--	--	--	01
(g) 100	-	-	-	101	--	--	--	01

- **Output Assignment**
  - **The output assignment are only associated with stable states.**
    - **Assign outputs in such a way that a transition through that unstable state will not cause a spurious pulse**

Exercise

- **Find the register that holds the up/down count in the interface card in our system.**
- **Determine what kind of servo motor is used in our lab.**
- **Hook up the servo motor and try to read the up/down pulse count from the interface card.**