Pakorn Watanachaturaporn
pakorn.wa@KMITL.ac.th

01076244 Advanced Digital System Design

Bachelor Program in Computer Engineering (B.Eng.)
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Designing State Machines Using State Diagrams

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

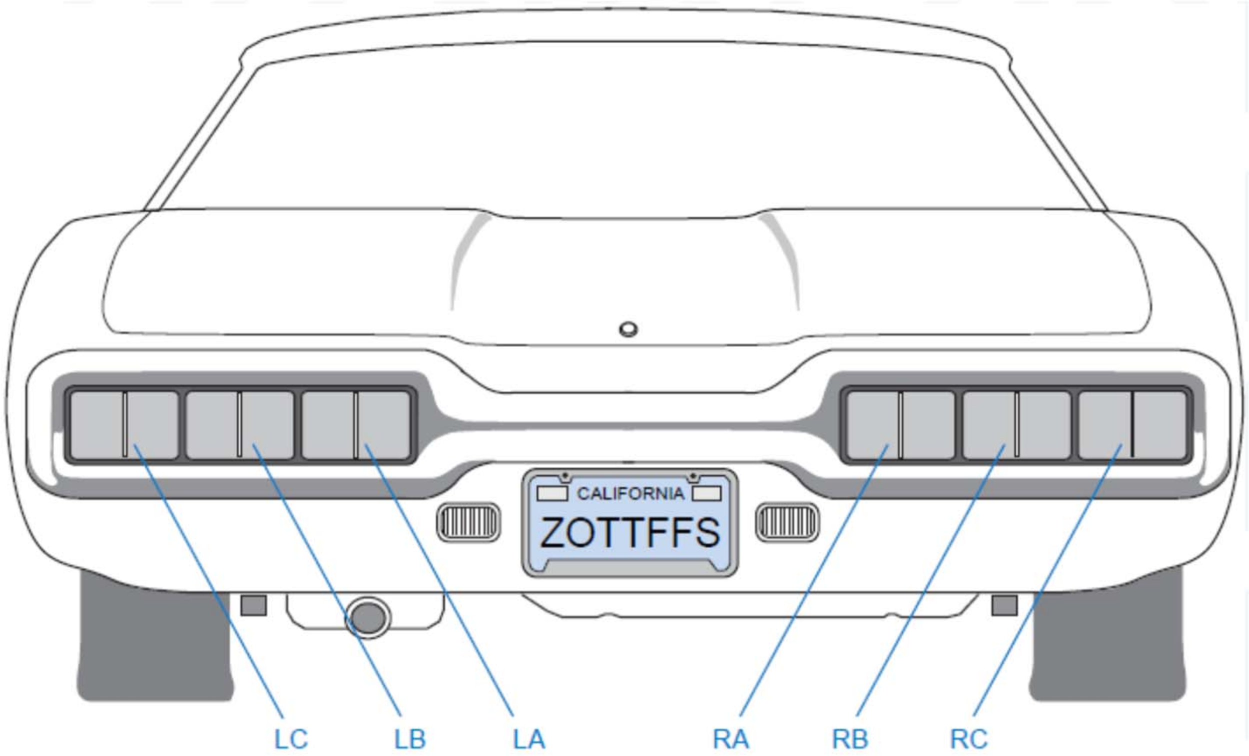
- Designing state machines is probably the most creative task of a digital designer.
- Most people like to take a graphical approach to design.
- State diagrams are often used to design small- to medium-sized state machines.

One fundamental difference between a state diagram and a state table; a difference that makes state-diagram design simpler but also more error prone:

- A state table is an exhaustive listing of the next states for each state/input combination.
  - No ambiguity is possible.
- A state diagram contains a set of arcs labeled with transition expressions.
  - Even when there are many inputs, only one transition expression is required per arc.
  - However, when a state diagram is constructed, there is no guarantee that the transition expression written on the arcs leaving a particular state cover all the input combinations exactly once.

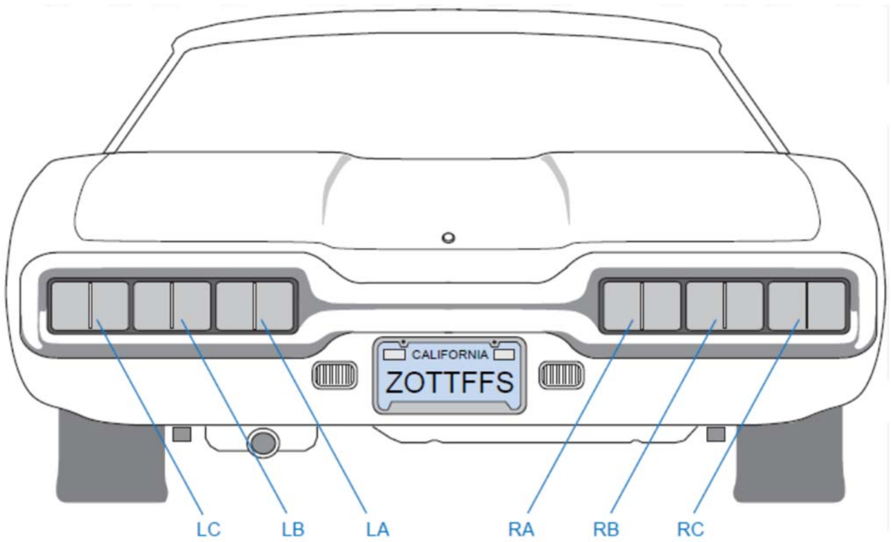
pakorn.wa@kmitl.ac.th

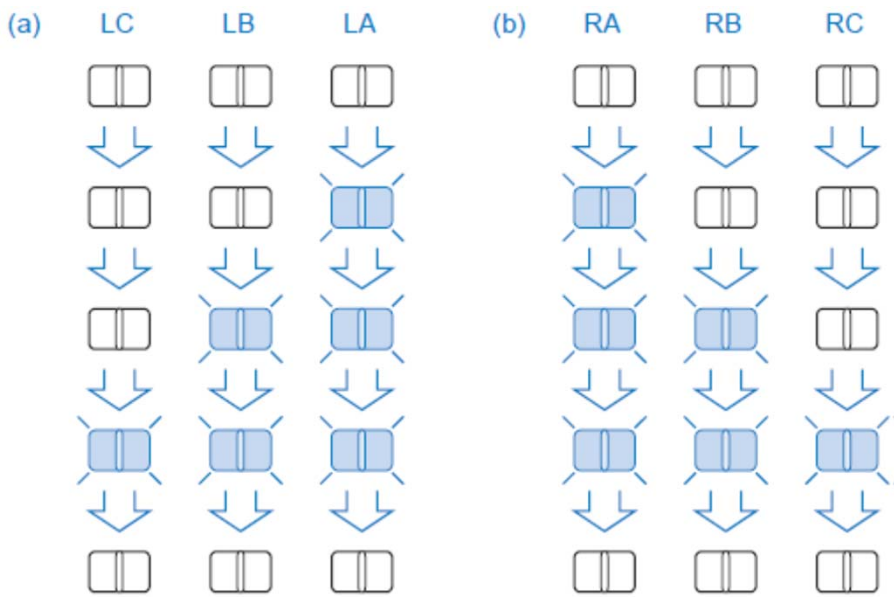
- Ambiguous state diagram
  - In an improperly constructed state diagram,
    - The next state for some input combinations may be unspecified, which is generally undesirable.
    - While multiple next states may be specified for others, which is just plain wrong.
  - Thus, considerable care must be taken in the design of state diagrams.



- The state machine has two input signals, LEFT and RIGHT.
- It also has an emergency flasher input, HAZ – all six lights flashing on and off in unison.
- Assume the existence of a free-running clock signal whose frequency equals the desired flashing rate for the lights.

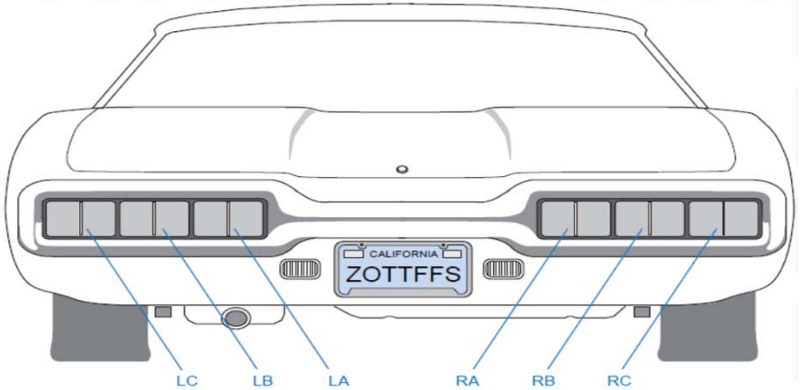
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

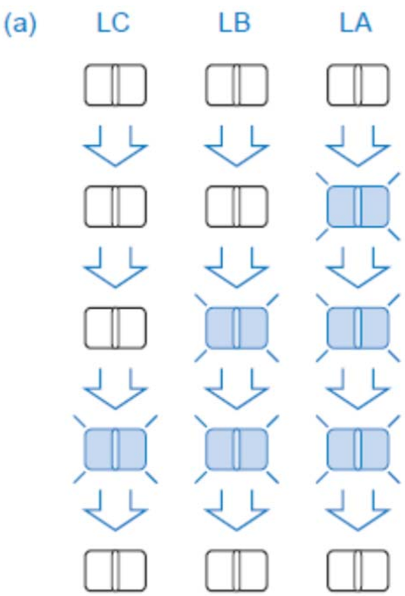




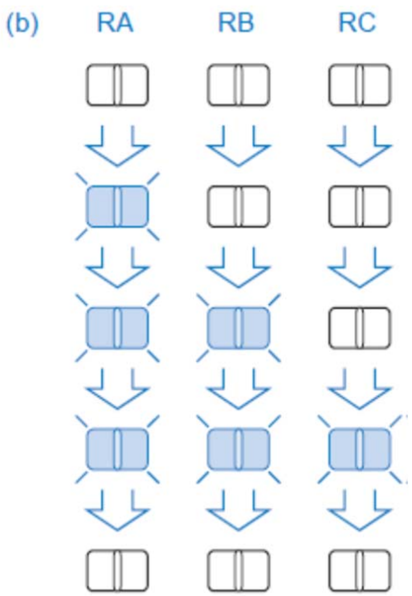
(a) Left turn (b) Right turn

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

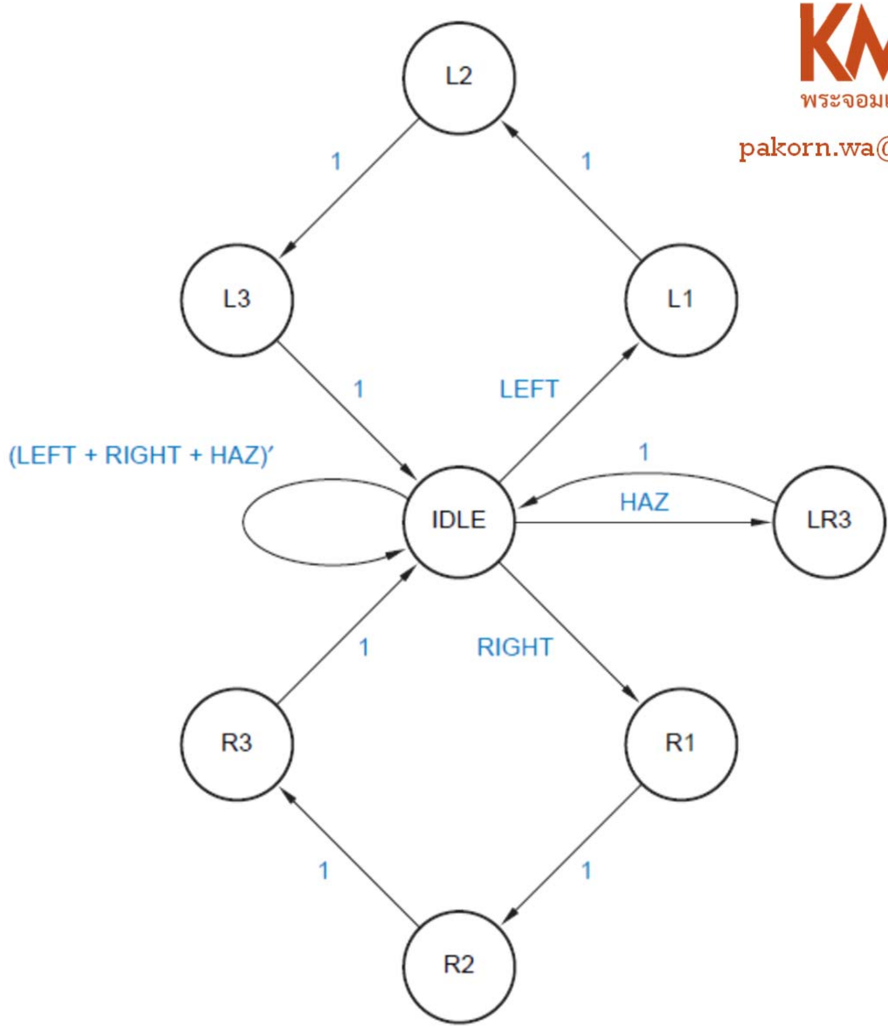




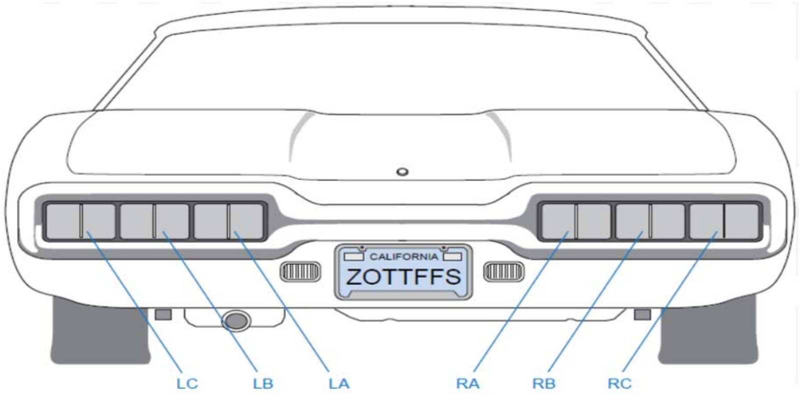
(a) Left turn

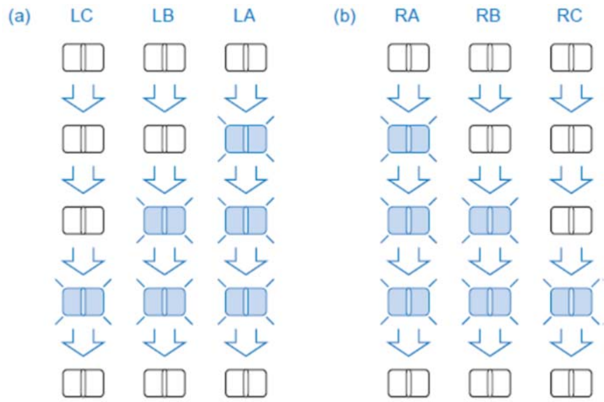


(b) Right turn

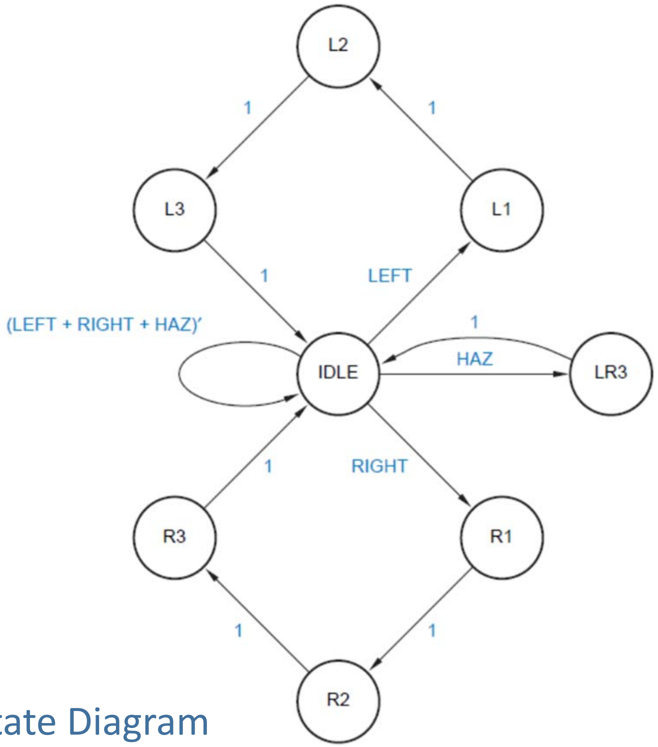


State Diagram





(a) Left turn (b) Right turn



State Diagram

State	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

Output Table

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

pakorn.wa@kmitl.ac.th

State	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

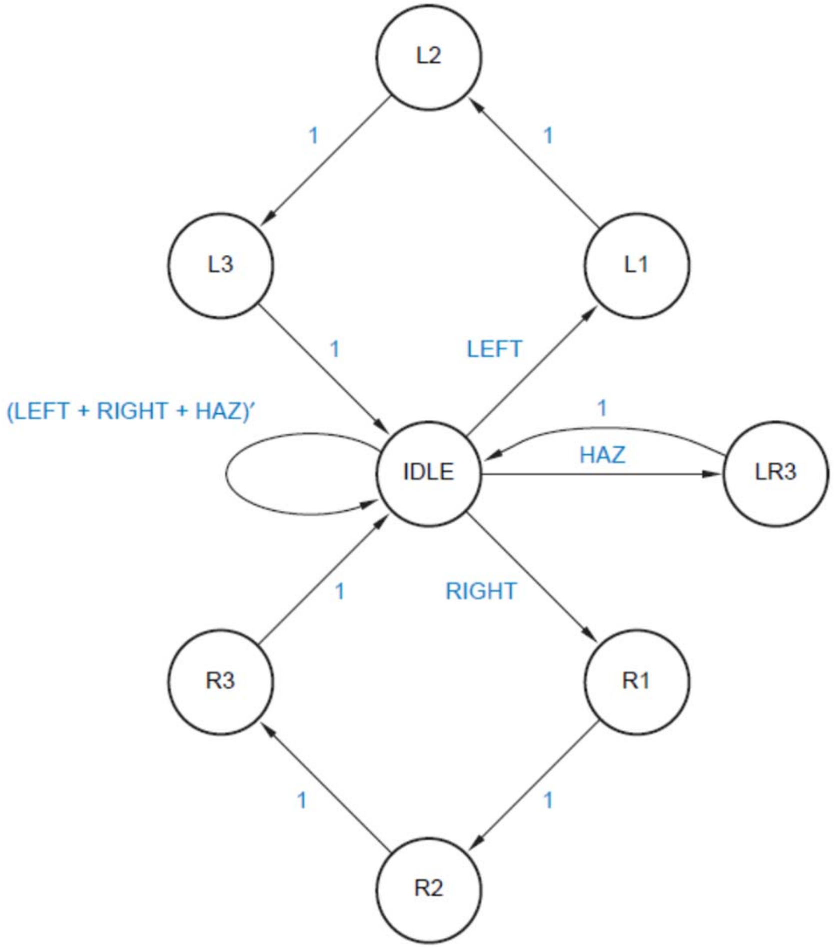
Output Table

LA = L1 + L2 + L3 + LR3
LB = L2 + L3 + LR3
LC = L3 + LR3

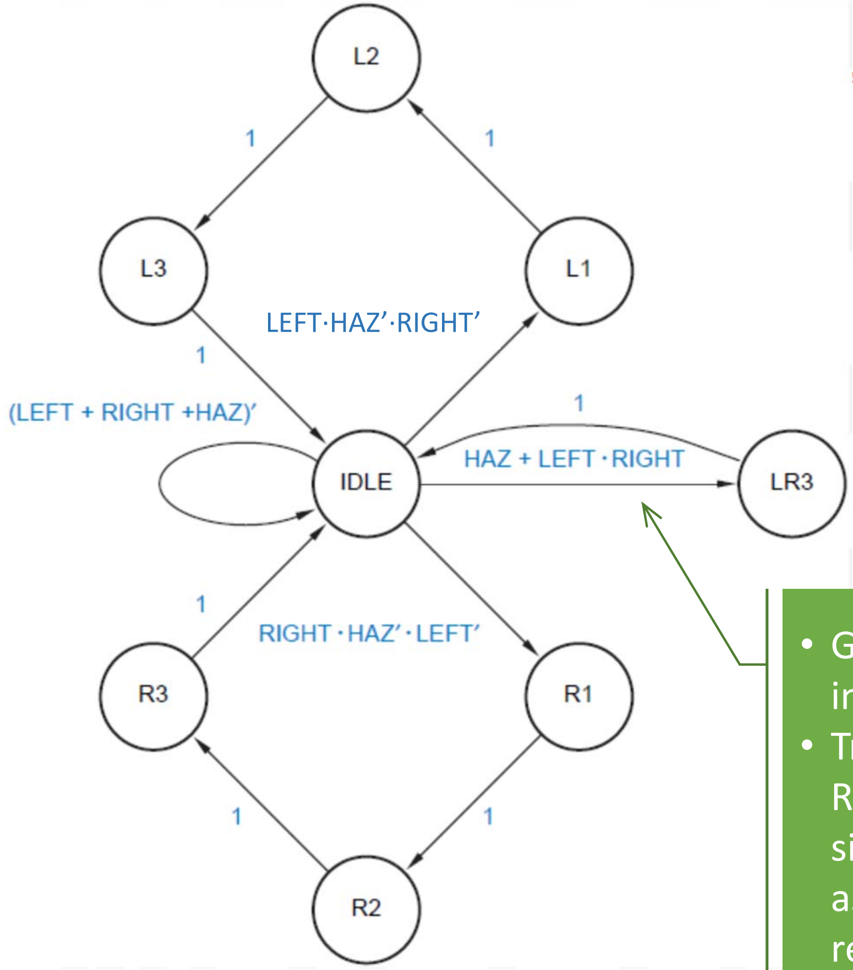
RA = R1 + R2 + R3 + LR3
RB = R2 + R3 + LR3
RC = R3 + LR3

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- What if ???
  - Both LEFT and HAZ are asserted simultaneously?
    - According to the state diagram, the machines goes to two states, L1 and LR3, which is impossible.
    - In reality, the machine would have only one next state, which could be L1, LR3 or a totally unrelated (and possible unused) third state, depending on details of the state machine's realization.



State Diagram



Corrected State Diagram

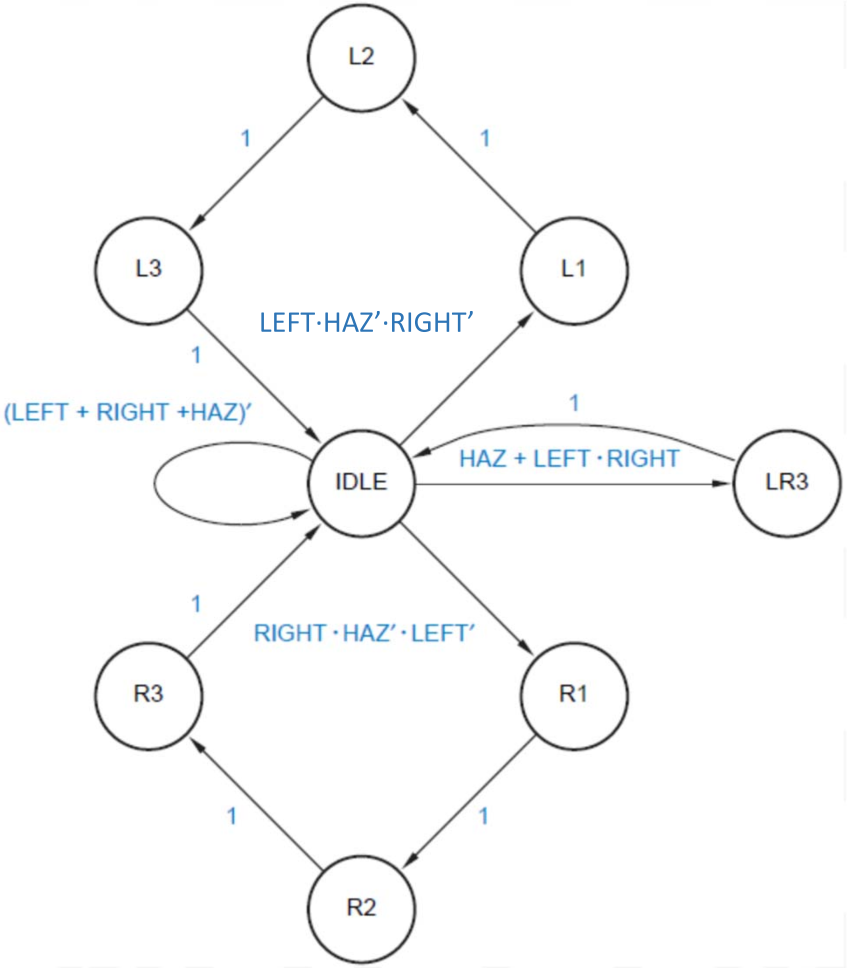
- Given the HAZ input priority
- Treat LEFT and RIGHT asserted simultaneously as a hazard request

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The new state diagram is unambiguous because the transition expressions on the arcs leaving each state are mutually exclusive and all-inclusive.
- That is, for each state,
  - No two expression are 1 for the same input combination, and
  - Some expression is 1 for every input combination

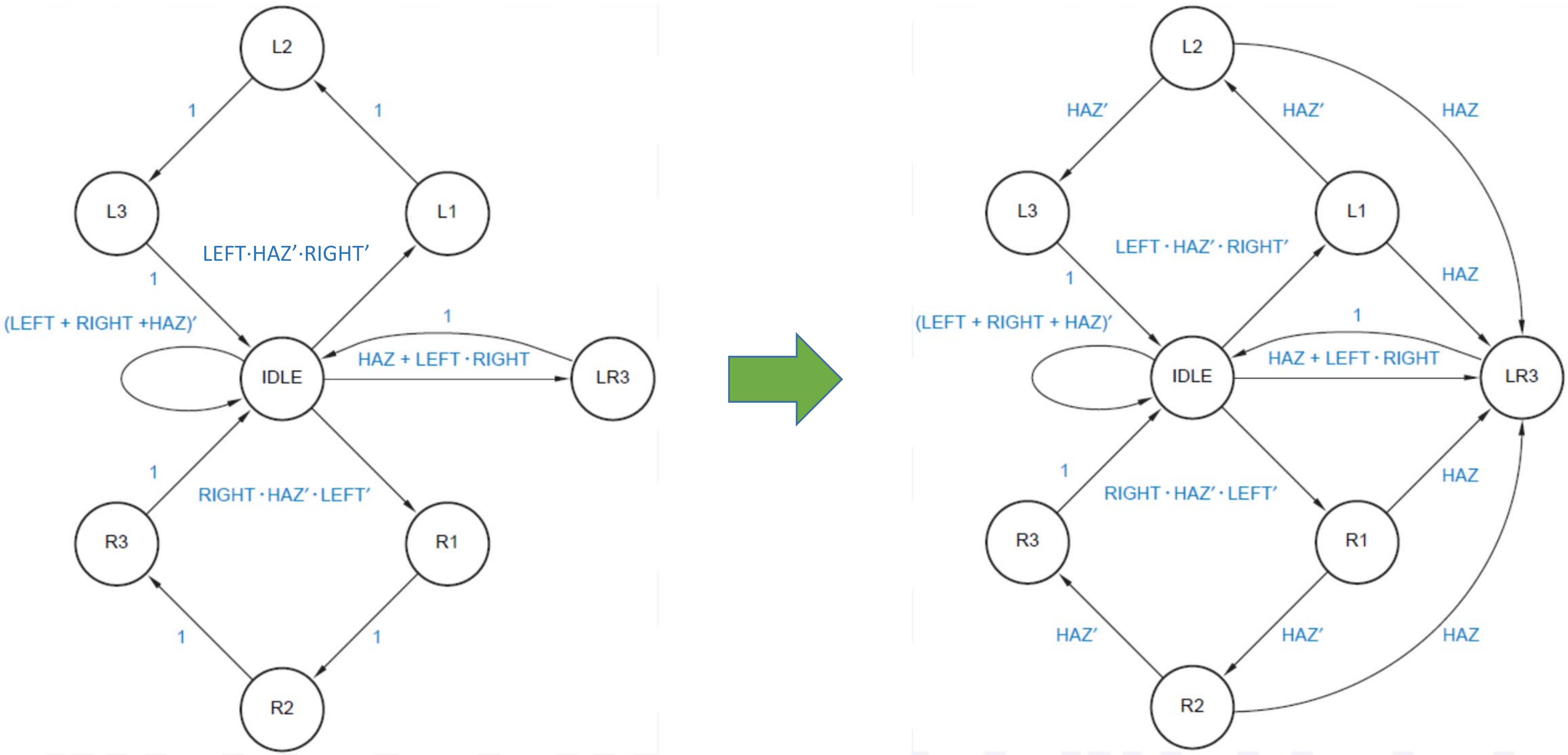
KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- This can be confirmed algebraically for this or any other state diagram by performing two steps:
  1. Mutual exclusion.
     - For each state, show that the logical product of each possible pair of transition expression on arcs leaving that state is 0.
     - If there are n arcs, then there are $n(n-1)/2$ logical products to evaluate.
  2. All inclusion.
     - For each state, show that the logical sum of the transition expressions on all arcs leaving that state is 1.

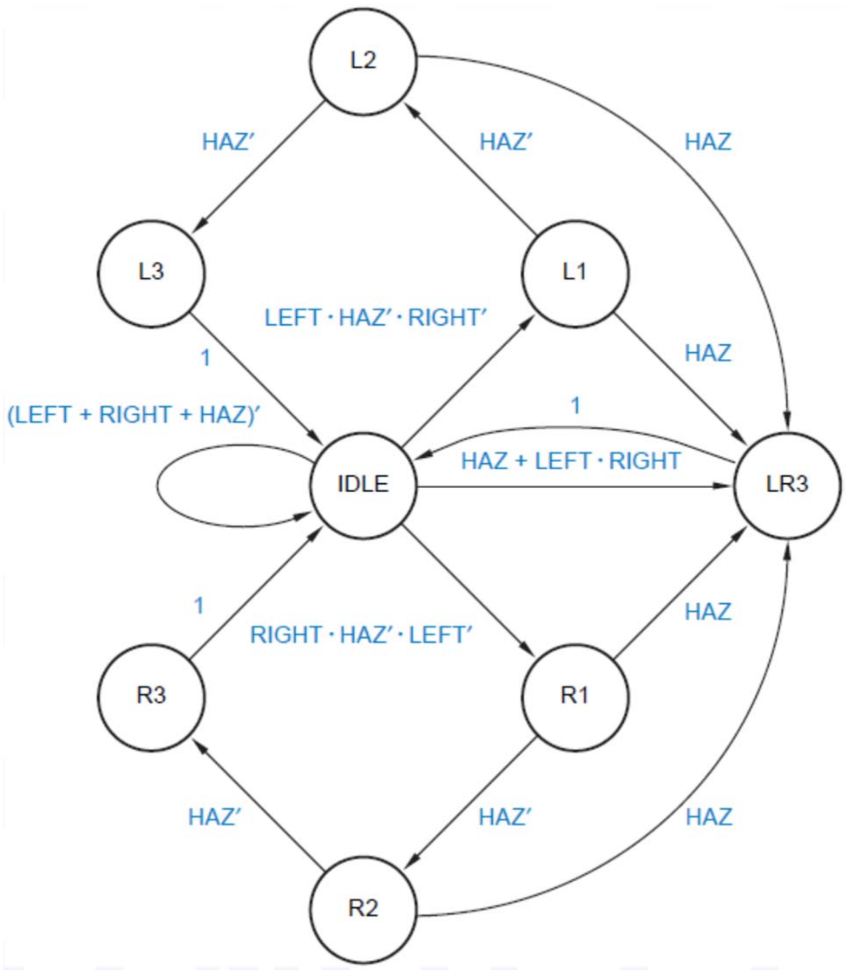


KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- What if ???
  - HAZ is asserted after a left- or right-turn cycles has begun.
  - While this may have a certain aesthetic appeal, it would be safer for the car’s occupants to have the machine go into hazard mode as soon as possible.


L2
1
1
L3
L1
LEFT·HAZ’·RIGHT’
1
(LEFT + RIGHT +HAZ)’
1
IDLE
HAZ + LEFT · RIGHT
LR3
1
RIGHT · HAZ’ · LEFT’
R3
R1
1
1
R2
L2
HAZ’
HAZ’
HAZ
L3
L1
LEFT · HAZ’ · RIGHT’
HAZ
1
(LEFT + RIGHT + HAZ)’
1
IDLE
HAZ + LEFT · RIGHT
LR3
1
RIGHT · HAZ’ · LEFT’
HAZ
R3
R1
HAZ’
HAZ’
HAZ
R2

pakorn.wa@kmitl.ac.th



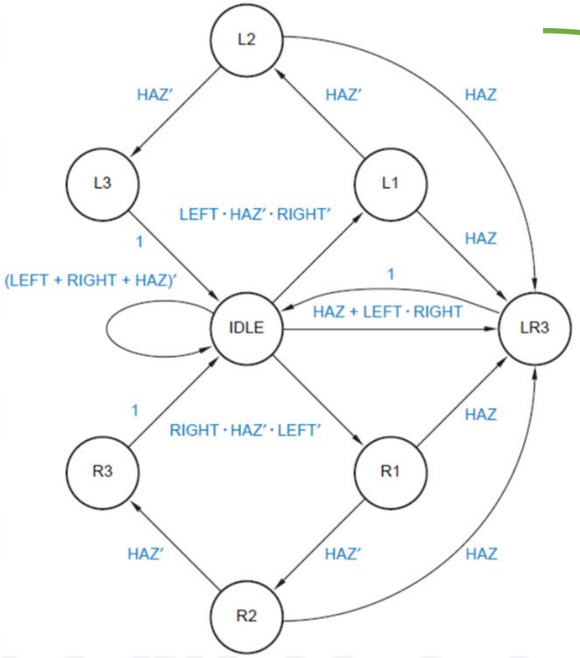
State	Q2	Q1	Q0
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0

State assignment

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

State	Q2	Q1	Q0
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0

- There are many state assignment possible.
- The table is used for the following reasons:
  1. An initial (idle) state of 000 is compatible with most flip-flops and registers, which are easily initialized to the 0 state.
  2. Two state variables, Q1 and Q0, are used to "count" in Gray-code sequence for the left-turn cycle (IDLE → L1 → L2 → L3 → IDLE). This minimizes the number of state-variable changes per state transition, which can **often** simplify the excitation logic.
  3. Because of the symmetry in the state diagram, the same sequence on Q1 and Q0 is used to "count" during a right-turn cycle, which Q2 is used to distinguish between left and right.
  4. The remaining state-variable combination is used for the LR3 state.



State	Q2	Q1	Q0
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

S	Q2	Q1	Q0	Transition expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	(LEFT + RIGHT + HAZ)'	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ' · RIGHT'	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ' · LEFT'	R1	1	0	1
L1	0	0	1	HAZ'	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ'	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ'	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ'	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

Transition list

State-Machine Synthesis Using Transition Lists

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The main purpose of this section is to help you understand the internal operation and the external quirks of CAD programs and languages that deal with state machines.

Transition Equations

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The first step is to develop a set of equations that define each next-state variable V* in terms of the current state and input.
- The transition list can be viewed as a sort of hybrid truth table in which the state-variable combinations for current-state are listed explicitly and input combinations are listed algebraically.
- Reading down a V* column in a transition list, we find a sequence of 0s and 1s, indicating the value of V* for various (if we've done it right, all) state/input combinations.

pakorn.wa@kmitl.ac.th

A transition equation for a next-state variable V* can be written using a sort of hybrid canonical sum:

$$V* = \sum_{\text{transition-list rows where } V*=1} (\text{transition p-term})$$

That is, the transition equation has one "transition p-term" for each row of the transition list that contains a 1 in the V* column. A row's transition p-term is the product of the current state's minterm and the transition expression.

S	Q2	Q1	Q0	Transition expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	(LEFT + RIGHT + HAZ)’	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ’ · RIGHT’	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ’ · LEFT’	R1	1	0	1
L1	0	0	1	HAZ’	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ’	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ’	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ’	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

$$
\begin{aligned}
Q2^* = {} & Q2' \cdot Q1' \cdot Q0' \cdot (HAZ + LEFT \cdot RIGHT) \\
& + Q2' \cdot Q1' \cdot Q0' \cdot (RIGHT \cdot HAZ' \cdot LEFT') \\
& + Q2' \cdot Q1' \cdot Q0 \cdot (HAZ) \\
& + Q2' \cdot Q1 \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q1 \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1 \cdot Q0 \cdot (HAZ)
\end{aligned}
$$

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

S	Q2	Q1	Q0	Transition expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	(LEFT + RIGHT + HAZ)’	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ’ · RIGHT’	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ’ · LEFT’	R1	1	0	1
L1	0	0	1	HAZ’	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ’	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ’	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ’	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

$$
\begin{aligned}
Q2^* = {} & Q2' \cdot Q1' \cdot Q0' \cdot (HAZ + LEFT \cdot RIGHT) \\
& + Q2' \cdot Q1' \cdot Q0' \cdot (RIGHT \cdot HAZ' \cdot LEFT') \\
& + Q2' \cdot Q1' \cdot Q0 \cdot (HAZ) \\
& + Q2' \cdot Q1 \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q1 \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1 \cdot Q0 \cdot (HAZ)
\end{aligned}
$$

$$
\begin{aligned}
Q2^* = {} & Q2' \cdot Q1' \cdot Q0' \cdot (HAZ + RIGHT) \\
& + Q2' \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q0
\end{aligned}
$$

pakorn.wa@kmitl.ac.th

S	Q2	Q1	Q0	Transition expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	(LEFT + RIGHT + HAZ)’	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ’ · RIGHT’	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ’ · LEFT’	R1	1	0	1
L1	0	0	1	HAZ’	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ’	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ’	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ’	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

$$
\begin{aligned}
Q2^* = {} & Q2' \cdot Q1' \cdot Q0' \cdot (HAZ + RIGHT) \\
& + Q2' \cdot Q0 \cdot (HAZ) \\
& + Q2 \cdot Q0
\end{aligned}
$$

$$
\begin{aligned}
Q1^* = {} & Q2' \cdot Q1' \cdot Q0 \cdot (HAZ') \\
& + Q2' \cdot Q1 \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1 \cdot Q0 \cdot (HAZ') \\
= {} & Q0 \cdot HAZ'
\end{aligned}
$$

$$
\begin{aligned}
Q0^* = {} & Q2' \cdot Q1' \cdot Q0' \cdot (LEFT \cdot HAZ' \cdot RIGHT') \\
& + Q2' \cdot Q1' \cdot Q0' \cdot (RIGHT \cdot HAZ' \cdot LEFT') \\
& + Q2' \cdot Q1' \cdot Q0 \cdot (HAZ') \\
& + Q2 \cdot Q1' \cdot Q0 \cdot (HAZ') \\
= {} & Q2' \cdot Q1' \cdot Q0' \cdot HAZ' \cdot (LEFT \oplus RIGHT) \\
& + Q1' \cdot Q0 \cdot HAZ'
\end{aligned}
$$

pakorn.wa@kmitl.ac.th

Excitation Equations

- So far we have derived only transition equations, not excitation equations.
- If we use D flip-flops as the memory elements in our state machines, then the excitation equations are trivial to derive from the transition equations, since the characteristic equation of a D flip-flop is $Q^* = D$.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Therefore, if the transition equation for a state variable Q* is

$$Qi^* = \text{expression}$$

then the excitation equation for the corresponding D flip-flop input is

$$Di = \text{expression}$$

Variations on the Scheme

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

If the column for a particular next-state variable contains fewer 0s than 1s, it may be advantages to write that variable's transition equation in terms of the 0s in its column.

$$V* = \prod_{\text{transition-list rows where } V^*=0} (\text{transition s-term})$$

Here, a row's transition s-term is the sum of the maxterm for the current state and the complement of the transition expression. If the transition expression is a simple product term, then its compliment is a sum, and the transition equation express V* in product-of-sums form.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Q & A