5.3 Realization of SM Chart

- state reduction:
  - difficult to eliminate redundant states in an SM chart as the chart is usually incompletely specified.
  - combining states may make the SM chart more difficult to interpret.

- state assignment
  - The best way of making the assignment depends on how the SM chart is realized.
    - gates & FFs (or equivalent PLD realization): follow the guidelines for state assignment given in 1-7.
    - programmable gate array (e.g. FPGA): one-hot encoding may be best
      - FPGAs are rich in FFs but poor in combinational paths

- output and next-state equations
  - after the state assignment has been made, output & state equations can be read directly from the SM chart.

  - procedure for deriving the next state equations for a FF Q from the SM chart
    1. identify all states in which Q = 1.
    2. For each of these states, find all link paths that lead to this state.
    3. For each of these link paths, find a term that is 1 when the link path is followed.
       - for a link path from S_i to S_j, the term will be 1 if the machine is in state S_i and the conditions for exiting to S_j are met.
    4. The expression for Q^+ is formed by

ORing together the terms in step 3.

Example

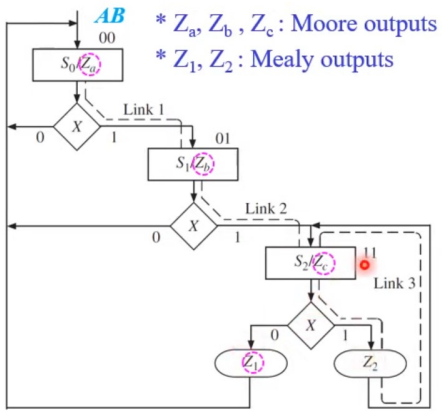
- Derive the *output & next state equations* from the SM chart.
- Output equations:

$Z_a = A'B'$

$Z_b = A'B$

$Z_c = AB$

$Z_1 = ABX'$



- Nest state equations:

link 2 link 3

$A^+ = A'BX + ABX$

$= \ldots$

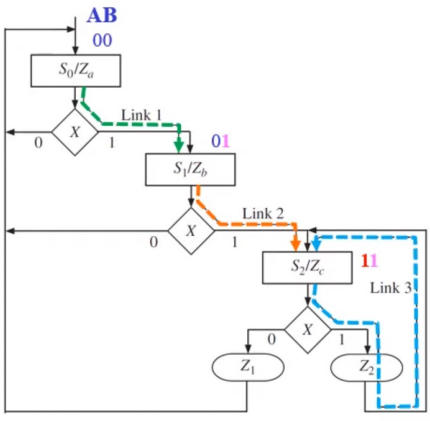
link 1 link 2

$B^+ = A'B'X + A'BX$

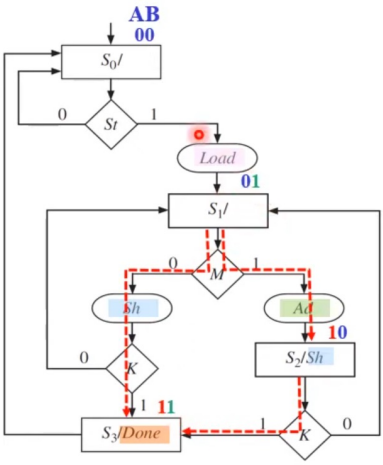
$+ ABX$

link 3

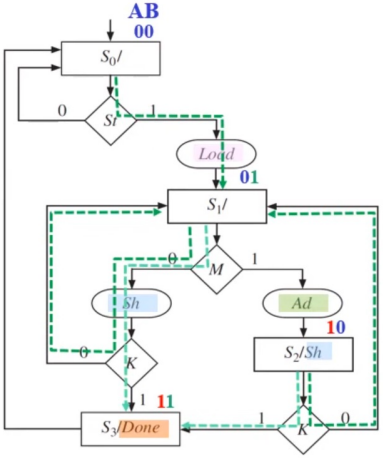
$= \ldots$



Implementation 1

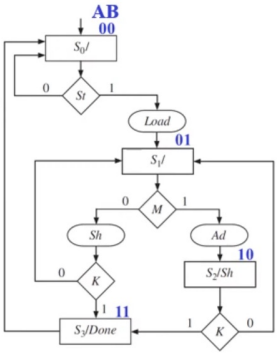


- **SM chart:**
  - trace ***link paths*** on the SM chart and then simplify the resulting equations
  - Output equations:
    - $Load = A'B'St$
    - $Sh = A'BM' + AB'$
    - $Ad = A'BM$
    - $Done = AB$
  - Next-state equations:
    - $A^+ = A'BM + A'BM'K + AB'K = A'B\,(M + K) + AB'K$
    - $B^+ =$
    - $=$



- **SM chart:**
  - trace ***link paths*** on the SM chart and then simplify the resulting equations
  - Output equations:
    - $Load = A'B'St$
    - $Sh = A'BM' + AB'$
    - $Ad = A'BM$
    - $Done = AB$
  - Next-state equations:
    - $A^+ = A'BM'K + A'BM + AB'K = A'B\,(M + K) + AB'K$
    - $B^+ = A'B'St + A'BM'K' + AB'K' + A'BM'K + AB'K$
      $= A'B'St + A'BM' + AB'$

Implementation 2



- **State transition table:**
  - Each **row** in the table corresponds to one ***link path*** in the SM chart.
  - The outputs for each row can be filled in by tracing the corresponding link paths on the SM chart.

	Present state		Inputs			Next state		Outputs			
	A	B	St	M	K	A^+	B^+	Load	Sh	Ad	Done
S_0	0	0	0	–	–	0	0	0	0	0	0
	0	0	1	–	–	0	1	1	0	0	0
S_1	0	1	–	0	0	0	1	0	1	0	0
	0	1	–	0	1	1	1	0	1	0	0
	0	1	–	1	–	1	0	0	0	1	0
S_2	1	0	–	–	0	0	1	0	1	0	0
	1	0	–	–	1	1	1	0	1	0	0
S_3	1	1	–	–	–	0	0	0	0	0	1

- Implement the multiplier controller w/ ROM:
  - Determine the size of the ROM: 32×6
    - 5 different inputs to the comb ckt (A, B, St, M, and K) $\Rightarrow$ 32 entries.
    - Comb ckt should generate six signals
      $\Rightarrow$ Each entry has to be 6 bits wide.
  - $\Rightarrow$ This design can be implemented using a 32×6 ROM and two D flip-flops.

- If a ROM is used, the table must be expanded to $2^5 = 32$ rows as there are 5 inputs.
  - Dashes must be replaced with all combinations of 0s and 1s. ⇒ If a row has n dashes, it must be replaced by 2^n rows.

A	B	St	M	K	A^+	B^+	Ld	Sh	Ad	Done
…										
0	1	–	1	–	1	0	0	0	1	0
…										

0	1	**0**	1	**0**	1	0	0	0	1	0
0	1	**0**	1	**1**	1	0	0	0	1	0
0	1	**1**	1	**0**	1	0	0	0	1	0
0	1	**1**	1	**1**	1	0	0	0	1	0

- Standard ROM (LUT) implementation of the multiplier: original SM chart
  - 4 states ⇒ 2 flip-flops, 2 next state equations
  - 3 inputs: St, M, K
  - 4 outputs: *Load*, *Sh*, *Ad*, *Done*
  - ROM size: 32 × 6

A	B	St	M	K	A^+	B^+	Ld	Sh	Ad	Done
…