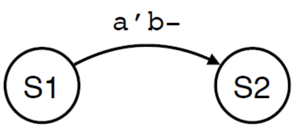
Systems analysis

1- **Draw state graph** – Graphical representation from verbal specs

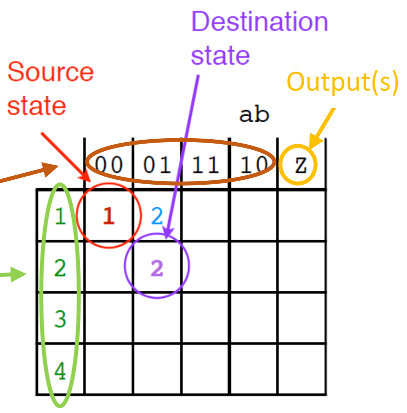
- System states = Circles
- State transitions = Oriented arcs
- Conditions = description on top of the arc
- Stable States = Self-arcs



2- **Primitive state table (Huffman Table)**

System evolution depending on present state and inputs

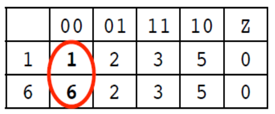
- **Bold** states are Stable States (if input doesn't change, state remains unchanged)
- Other columns indicate all input possibilities
- First column shows present states
- Non-existing transitions are marked with *don't cares*
- NB: inputs should not change during transitions.



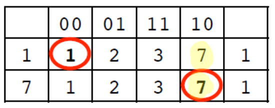
3- **Optimisation of primitive state table**

Reducing the number of states simplifies the corresponding logic circuit

- Notion of State Equivalence - States are equivalent iff:
  - Same lines (means that all future states and output are the same)
  - Stable states in the same column
- Notion of State Fusion
  - Same lines
  - Stable states in different columns

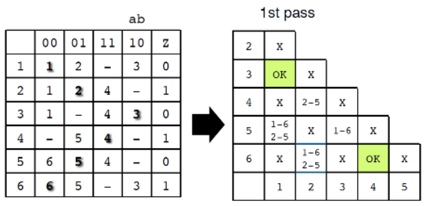


	00	01	11	10	Z
1	**1**	2	3	5	0
6	**6**	2	3	5	0



	00	01	11	10	
1	**1**	2	3	7	1
7	1	2	3	**7**	1

1. Systematic search of equivalences
   1. Build the equivalences conditions table [1->(n-1) x 2->n]
      - Moore Machine (output is function of state variables only)
        1. If two states equivalent -> put 'OK'
        2. Else if other pair of states (x,y) need to be equivalent & they <u>both</u> aren't stable states -> put 'x-y'
        3. Else - not equivalent -> put 'X'



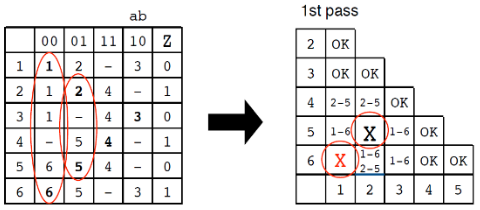
ab

	00	01	11	10	Z
1	**1**	2	–	3	0
2	1	**2**	4	–	1
3	1	–	4	**3**	0
4	–	5	**4**	–	1
5	6	**5**	4	–	0
6	**6**	5	–	3	1

1st pass

2	X				
3	OK	X			
4	X	2–5	X		
5	1–6 2–5	X	1–6	X	
6	X	1–6 2–5	X	OK	X
	1	2	3	4	5

      - Mealy machine (output is function of state variables AND system input)
        1. Can also merge states with different outputs iff future stable states in different columns.



ab

	00	01	11	10	Z
1	**1**	2	–	3	0
2	1	**2**	4	–	1
3	1	–	4	**3**	0
4	–	5	**4**	–	1
5	6	**5**	4	–	0
6	**6**	5	–	3	1

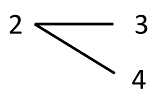
1st pass

2	OK				
3	OK	OK			
4	2–5	2–5	OK		
5	1–6	X	1–6	OK	
6	X	1–6 2–5	1–6	OK	OK
	1	2	3	4	5

   2. 2nd pass:
      1. Highlight in red all "conditional cells" if one condition couple has an 'X' in the precedent table.

2. Others are automatically green cells

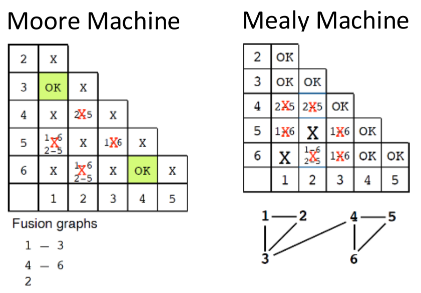
b. Draw state fusion graph



2-3

OR

2-4



c. Build the fused table

i. Moore machine:

1. Simply copy Z of merged states (since it's same for all merged states)

ii. Mealy machine (see T.P. 6):

1. Stable merged future states inherit their own previous output

4- **Solve Race Conditions (asynchronous logic circuits only)**

Race conditions are a sudden change of 2 (or more) variables during a transition. Solutions depends on circuit classes:

- <u>Asynchronous logic circuits</u> – Up to 1 internal variable change at most

  NB: It is mandatory to solve them in asynchronous logic circuits!

  i. Search of race conditions in state tables:

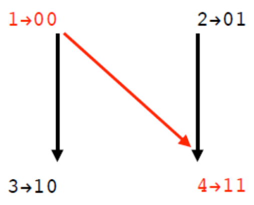
  There is a race condition if during transition from present state to unstable state, both bits change.

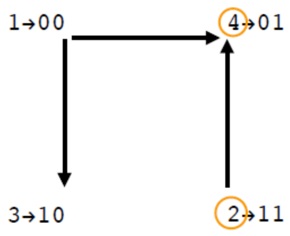
  ii. Method 1 – State encoding

  Try to change encoding so that there are no race conditions.

  1. Build state encoding graphs (like a state graph without conditions)
  2. Diagonals indicate Race conditions
  3. If race, swap codes (1,2,3,4), so that adjacent vertices have max H.D. = 1
  4. NB: DON'T FORGET to re-swap lines for K-Map optimisation!

Race condition since we have one diagonal





  iii. Method 2 – Transition modifications

  Since only source and destination are important, we could:

  - Edit transitions
  - Add extra transitions (Ex: **00**->11->**11** becomes **00**->01->11->**11**)
  - Use *don't care* instead of a state transition

  iv. Method 3 – Adding an extra state variable to (re-)enable method 2

  Add an extra state variable (so extra 2^n transitions == extra n lines, where n=state variables at beginning)

- <u>Synchronous logic circuits</u> – Different approach: Use of memory to synchronise state values. *Principle: State variable update in regular time intervals, driven by external control signal (Clock). The period is computed so that the slowest signal (critical path) has enough time to travel. We use one of the FFs, where their excitation functions steer the content of these memories.*

  i. Encoded state table (any encoding & leaving race conditions as they are)

  ii. Excitation table: Rewrite encoded state table following these rules (where $Q=y_i$ and $Q+=Y_i$)

Q	Q+	Operation	Code
0	0	Maintain 0	μ_0
0	1	Enable	ε
1	0	Disable	δ
1	1	Maintain 1	μ_1

	00	01	11	10
00	**00/1**	**00/0**	11	01
01	**01/0**	00	10	**01/1**
11	00	**11/1**	**11/0**	10
10	01	11	**10/1**	**10/0**

➡

	00	01	11	10
00	$\mu_0\mu_0$	$\mu_0\mu_0$	$\varepsilon\varepsilon$	$\mu_0\varepsilon$
01	$\mu_0\mu_1$	$\mu_0\delta$	$\varepsilon\delta$	$\mu_0\mu_1$
11	$\delta\delta$	$\mu_1\mu_1$	$\mu_1\mu_1$	$\mu_1\delta$
10	$\delta\varepsilon$	$\mu_1\varepsilon$	$\mu_1\mu_0$	$\mu_1\mu_0$

iii. Memory excitation tables
   1. Split the excitation table into one per memory element.
   2. Rewrite them using the ‘little’ table of the memory element

ab

M_1	00	01	11	10
00	μ_0	μ_0	ε	μ_0
01	μ_0	μ_0	ε	μ_0
11	δ	μ_1	μ_1	μ_1
10	δ	μ_1	μ_1	μ_1

	J	K
μ_0	0	–
ε	1	–
δ	–	1
μ_1	–	0

ab

M_2	00	01	11	10
00	μ_0	μ_0	ε	ε
01	μ_1	δ	δ	μ_1
11	δ	μ_1	μ_1	δ
10	ε	ε	μ_0	μ_0

iv. K-Maps & optimized excitation functions

ab

J_1K_1	00	01	11	10
00	0–	0–	1–	0–
01	0–	0–	1–	0–
11	–1	0	0	0
10	–1	0	0	0

ab

J_2K_2	00	01	11	10
00	0–	0–	1–	1–
01	0	–1	–1	0
11	–1	0	0	–1
10	1–	1–	0–	0–

   1. Split each table to obtain functions J_1, K_1, J_2, K_2

5- Feedback logic functions

6- Output of transitions

- Moore Machine (output is function of state variables only)
  NB: Only one transition allowed between two stable states to avoid glitches!
  i. Stable states: take the output value of the same line
  ii. Transitions from present states (to another column):
     ▪ If Z of departure state = Z of arrival state -> PUT Z
     ▪ If Z of departure state ≠ Z of arrival state -> PUT *don’t care*
  iii. K-Map and extract Z logic function

means 00 → 01

What are Z for 00 & 01?

0 0 → Same so we put a 0

ab

	00	01	11	10	Z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

ab

Z	00	01	11	10
00	**0**	0	**0**	–
01	0	**0**	–	–
11	–	–	**1**	1
10	–	–	–	**1**

- Mealy machine (output is function of state variables AND system input)
  Two present states with different outputs can be fused, if stable states in different columns.
  ▪ Case 1: Single transition

ab

	00	01	11	10
1	**1/1**		2	**1/0**
2			**2/0**	
3				
4				

➡

ab

Z	00	01	11	10
1	**1**		0	**0**
2			**0**	
3				
4				

  i. Check if a single transition is used by two same stable states with different output.
  ii. Check the destination stable state reached through this transition state.
  iii. Assign to the transition state the same output as destination and one of the departure stable states.

  ▪ Case 2: Multiple (eventually) shared transitions

ab

	00	01	11	10
1	**1/1**		2	**1/1**
2		**2/0**	3	
3			**3/0**	
4				

➡

ab

Z	00	01	11	10
1	**1**		–	**1**
2		**0**	**0**	
3			**0**	
4				

  i. Same rules as case 1 but give priority to the set having the same result as destination.

7- Enumerate all logic functions

8- Draw circuit diagram