Sequential Logic : A Review

- State Minimization (Implication Chart vs Row Matching)
- DataPath vs Control Path
- Reverse Engineering FSM
- FSMs: Mealy vs Moore
- FFs converting one from another and using them to implement

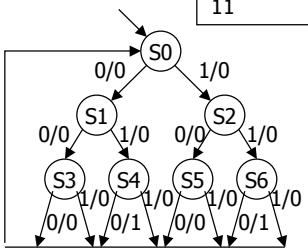
Algorithmic approach to state minimization

- Goal – identify and combine states that have equivalent behavior
- Equivalent states:
  - same output
  - for all input combinations, states transition to same or equivalent states
- Algorithm sketch
  - 1. place all states in one set
  - 2. initially partition set based on output behavior
  - 3. successively partition resulting subsets based on next state transitions
  - 4. repeat (3) until no further partitioning is required
    - states left in the same set are equivalent
  - polynomial time procedure

State minimization example

⌘ Sequence detector for 010 or 110

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0



Method of successive partitions

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0

(S0 S1 S2 S3 S4 S5 S6)

(S0 S1 S2 S3 S5) (S4 S6)

(S0 S3 S5) (S1 S2) (S4 S6)

(S0) (S3 S5) (S1 S2) (S4 S6)

S1 is equivalent to S2

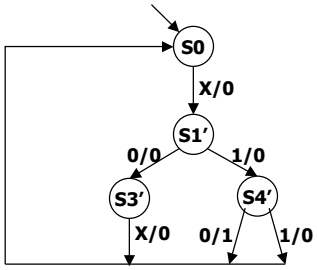
S3 is equivalent to S5

S4 is equivalent to S6

Minimized FSM

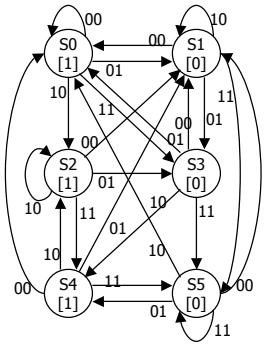
⌘ State minimized sequence detector for 010 or 110

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1'	S1'	0	0
0 + 1	S1'	S3'	S4'	0	0
X0	S3'	S0	S0	0	0
X1	S4'	S0	S0	1	0



More complex state minimization

⌘ Multiple input example



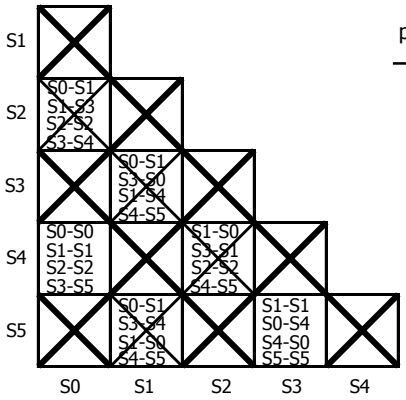
inputs here

present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

symbolic state transition table

Minimized FSM

- Implication chart method
  - cross out incompatible states based on outputs
  - then cross out more cells if indexed chart entries are already crossed out



present state	next state 00	01	10	11	output
S0'	S0'	S1	S2	S3'	1
S1	S0'	S3'	S1	S3'	0
S2	S1	S3'	S2	S0'	1
S3'	S1	S0'	S0'	S3'	0

minimized state table
(S0==S4) (S3==S5)

Minimizing incompletely specified FSMs

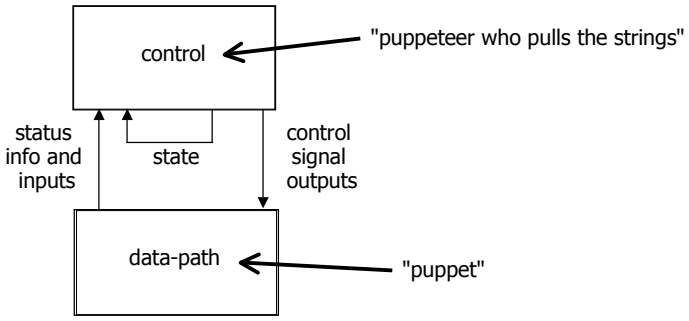
- Equivalence of states is transitive when machine is fully specified
- But its not transitive when don't cares are present

e.g.,	state	output	
	S0	– 0	S1 is compatible with both S0 and S2
	S1	1 –	but S0 and S2 are incompatible
	S2	– 1	

- No polynomial time algorithm exists for determining best grouping of states into equivalent sets that will yield the smallest number of final states

Data-path and control

- Digital hardware systems = data-path + control
  - datapath: registers, counters, combinational functional units (e.g., ALU), communication (e.g., busses)
  - control: FSM generating sequences of control signals that instructs datapath what to do next



Comparison of Mealy and Moore machines

- Mealy machines tend to have less states
  - different outputs on arcs (n^2) rather than states (n)
- Moore machines are safer to use
  - outputs change at clock edge (always one cycle later)
  - in Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected – asynchronous feedback
- Mealy machines react faster to inputs
  - react in same cycle – don't need to wait for clock
  - in Moore machines, more logic may be necessary to decode state into outputs – more gate delays after