Outline

- ❍ Last time:
  - ➔ Asynchronous Circuits
  - ➔ Moore and Mealy Standard Forms
  - ➔ Design Example: Word Problem
- ❍ This lecture:
  - ➔ Race-free State Assignment
  - ➔ Excitation Equations
  - ➔ Design Example: Implementation
  - ➔ Summary of Asynchronous Design Process

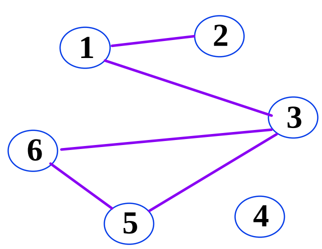
Steps to Asynchronous FSM Design

- Construct a *Primitive Flow Table* from the word statement of the problem.
- ✓ Derive a minimum-row primitive flow table or *Reduced Primitive Flow Table* by eliminating redundant, stable total-states.
- ❍ Convert the resulting table to Mealy form, if necessary, so that the output value is associated with the total state rather than the internal state.
- ❍ Derive a minimum-row flow table, or *Merged Flow Table*, by merging compatible rows of the reduced primitive flow table using a merger diagram. (Note: solution not necessarily unique)
- ❍ Perform race-free, or critical-race-free, state assignment, adding additional states if necessary.
- ❍ Complete the *Output Table* to avoid momentary false outputs when switching between stable total states.
- ❍ Draw *logic diagram* that shows ideal combinational next-state and output functions as well as necessary delay elements.

Conversion to Mealy Form for Merging: Example 1

	X_1X_2 00	01	11	10	Z 00	01	11	10
1	(1)	2	-	3	0	-	-	-
2	1	(2)	4	-	-	0	-	-
3	1	-	5	(3)	-	-	-	0
4	-	6	(4)	3	-	-	1	-
5	-	6	(5)	3	-	-	0	-
6	1	(6)	5	-	-	0	-	-

Create Merger Diagram



What we are doing is choosing *internal* states that can have the *same* code, though the *total* states will still be different (since they will not have the same code for the same inputs; if that were possible they would have been redundant).

Two rows of a reduced primitive Mealy flow table are *compatible* and can be merged into a single row iff there are no state or output conflicts in any column.

Draw a line between any pair of rows (states) which are compatible and can be merged.

Choose the sub-groups of fully-connected rows that will result in the maximum reduction. In this case, {1,2} and {3,5,6} are best. We could also have chosen {1,3} and {5,6}, or just {3,6} in which case no others would be permitted.

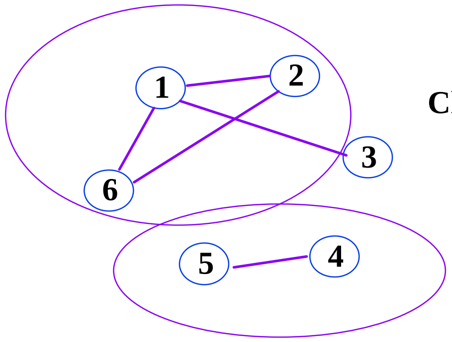
Example 1: After Merging

	X1X2 00	01	11	10	00	01	11	10
a(1,2)	1	2	4	3	0	0	-	-
b(3,5,6)	1	6	5	3	-	0	0	0
c (4)	-	6	4	3	-	-	1	-

Conversion to Mealy Form for Merging: Example 2

	CS 00	01	11	10	Z 00	01	11	10
1	1	3	-	2	0	-	-	-
2	1	-	6	2	-	-	-	0
3	1	3	4	-	-	0	-	-
4	-	3	4	5	-	-	1	-
5	1	-	-	5	-	-	-	1
6	-	3	6	-	-	-	0	-

Create Merger Diagram: Example 2



Choose {1,2,6} and {4,5}

Steps to Asynchronous FSM Design

Construct a *Primitive Flow Table* from the word statement of the problem.

✓ Derive a minimum-row primitive flow table or *Reduced Primitive Flow Table* by eliminating redundant, stable total-states.

✓ Convert the resulting table to Mealy form, if necessary, so that the output value is associated with the total state rather than the internal state.

✓ Derive a minimum-row flow table, or *Merged Flow Table*, by merging compatible rows of the reduced primitive flow table using a merger diagram. (Note: solution not necessarily unique)

❍ Perform race-free, or critical-race-free, state assignment, adding additional states if necessary.

❍ Complete the *Output Table* to avoid momentary false outputs when switching between stable total states.

❍ Draw *logic diagram* that shows ideal combinational next-state and output functions as well as necessary delay elements.

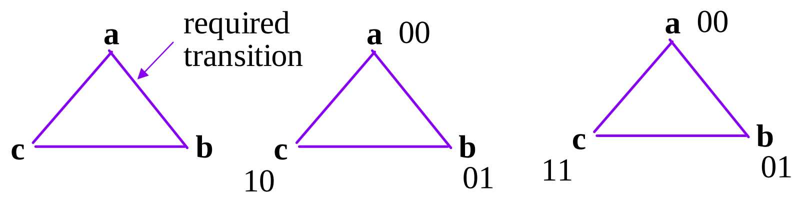
Example: After State Reduction and Merging

X1X2	00	01	11	10	00	01	11	10
a(1,2)	1	2	4	3	0	0	-	-
b(3,5,6)	1	6	5	3	-	0	0	0
c (4)	-	6	4	3	-	-	1	-

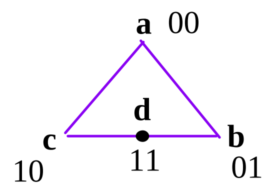
Example 1: States Labelled in Terms of Internal States

X1X2	00	01	11	10	00	01	11	10
a	a	a	c	b	0	0	-	-
b	a	b	b	b	-	0	0	0
c	-	b	c	b	-	-	1	-

Example: Race-Free State Assignment



❍ Line between states indicates required transition.
❍ Need assignment such that only one state variable changes during each state transition



Example: Critical-Race-Free State Assignment

X1X2	00	01	11	10	00	01	11	10
a 00	00	00	10	01	0	0	-	0v
b 01	00	01	01	01	0u	0	0	0
d 11	-	01	-	-	-	-s	-	-
c 10	-	11	10	00	-	1t	1	-

❍ In col. 01, row C, 10->11->01: critical race avoided
❍ In col 10, could have directed transition via row d but already have transition to b in row a, so use it.
❍ Fill in outputs corresponding to unstable states to avoid momentary false outputs during transitions

Critical-Race-Free State Assignment via Shared-Row Assignments

	00	01	11	10
a	1	2	5	4
b	7	2	3	10
c	1	8	3	4
d	7	8	5	4
e	1	9	5	6
f	11	8	3	6

- **Required transitions:**
  - **col 00: e,c -> a; d->b**
  - **col 01: a->b; c,f->d**
  - **col 11: b,f->c; a,d->e**
  - **col 10: a,c->d; e->f**

Shared Row Assignment

- Consider required transitions: e,c -> a
- Could implement as: e->a, c->a; or e->c->a; or c->e->a; or any of the above going through an intermediate state (or states).
- To avoid critical races, must ensure that {a,c,e} are placed in adjacent squares on the assignment map. Similar for other constraints. Must satisfy:

  {a,c,e}, {b,d}, {a,b}, {c,d,f}, {b,c,f}, {a,d,e}, {a,c,d}, {e,f}

a	e
b	d
x	c
y	f

Completion of the Output Table

X1X2	00	01	11	10	00	01	11	10
a 00	00	00	10	01	0	0	-	0v
b 01	00	01	01	01	0u	0	0	0
d 11	-	01	-	-	-	-s	-	-
c 10	-	11	10	00	-	1t	1	-

From State 00, input change 00->10 causes transition to stable state 01. To avoid glitch if logic synthesis assigns "don't care" in output to 1 for input value 10, make it a 0 (labelled v above).

From State 01, input change 01->00 causes transition to State 00, so avoid possible glitch by assigning output in State 01 for input 00 to 0 (u above)

From State 10, input change 11->01 causes transition to stable state 01, via 11. Since output goes from 1 to 0, choose output at t above to be 1, consistent with the starting value, but leave output at s a "don't care" since must make the transition somewhere and either before or after State 11 is the same.

Steps to Asynchronous FSM Design

Construct a *Primitive Flow Table* from the word statement of the problem.

✓ Derive a minimum-row primitive flow table or *Reduced Primitive Flow Table* by eliminating redundant, stable total-states.

✓ Convert the resulting table to Mealy form, if necessary, so that the output value is associated with the total state rather than the internal state.

✓ Derive a minimum-row flow table, or *Merged Flow Table*, by merging compatible rows of the reduced primitive flow table using a merger diagram. (Note: solution not necessarily unique)

✓ Perform race-free, or critical-race-free, state assignment, adding additional states if necessary.

✓ Complete the *Output Table* to avoid momentary false outputs when switching between stable total states.

❍ Draw *logic diagram* that shows ideal combinational next-state and output functions as well as necessary delay elements.