State Assignment

When implementing a given state table, we often desire a "state assignment" that will minimize the amount of logic required. Having 1's next to each other in a K-map will generally result in simpler (lower cost) logic equations. Our objective, therefore, is to somehow make an assignment that results in groups of 1's being next to each other. One solution is to simply try all possible assignments and then pick the one that results in the least amount of logic. However, this is not practical when there are more than a handful of states. A more practical approach is to follow a set of guidelines that tend to result in a lot of 1's next to each other in the required K-maps. (Recall that adjacent cells in a K-map differ by only one variable.)

We will use the following guidelines:

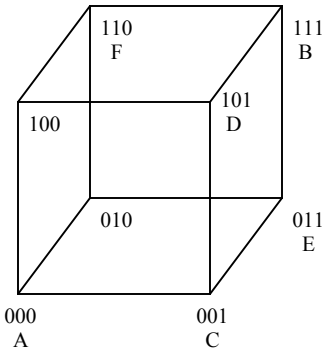
1) (Highest priority) States which have the same next state for each possible input combination should be given adjacent assignments. Identical next states in only some "columns" (i.e., for some input combinations) may be included in this list but with lesser priority.

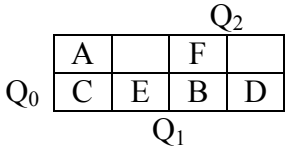
2) (Medium priority) States which are next states of the same state should be given adjacent assignments.

3) (Lowest priority) States which have the same output for each input should be given adjacent assignments. States having the same output for only some columns (input combinations) may still be included but with lesser priority

You might want a particular state to be a "Reset" state, so assign that one to the state having all bits as zero. (E.g., state "A" below.) Clearing all flip-flops (assuming a clear line is attached to each one) will therefore put the state machine in this state.

We won't necessarily be able to satisfy all guidelines, but in general the better job we do in satisfying the higher priority guidelines, the simpler the logic tends to be. We can either lay out the states on a Boolean "assignment" cube or in a K-map style grid. A six-state example assignment (requiring three state variables; i.e., three flip-flops) might be:





K-map version

The following example is a Moore state table with two inputs (X and Y) and two outputs (Z_1 and Z_2)

PS	NS				Output
	XY = 00	XY = 01	XY = 10	XY = 11	Z_1Z_2
A	B	A	C	E	10
B	B	F	F	G	11
C	G	A	B	E	01
D	A	D	F	G	00
E	B	A	C	E	11
F	F	E	F	A	00
G	G	F	E	B	10

Applying the above guidelines gives us the following "wish list" for states that we would like to have next to each other on the assignment cube. The first says that states A and E would like to be next to each other. Rows above have higher priority than those below.

Guideline 1) (A E) (have same next states for all input combinations)
(A C) (B D) (C E) (have same next states in two columns)
(A B) (B E) (B F) (B G) (C G) (D F) (same N.S. in one column)

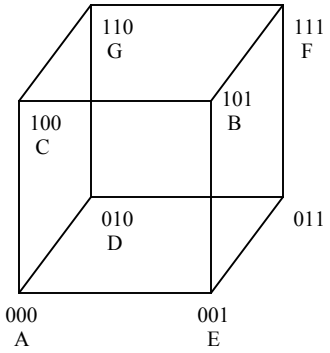
Guideline 2) (A B C E) (These are next states of both A and E)
(B F G) (A B E G) (A D F G) (A E F) (B E F G) (NS of some state)

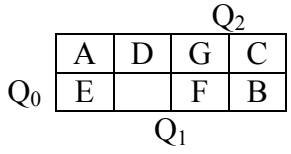
Guideline 3) (A G) (B E) (D F) (States having the same output)

For Guideline 2, "BACE" occurred for two states (both A and E), so it has a higher priority than other Guideline 2 entries.

Note that when 3 or 4 states are desired to be next to each other, the best you can do is to try to get them on the same "plane" in the assignment cube.

The following assignment seems to do a fair job of satisfying the above guidelines. Other assignments may satisfy the guidelines about as well. Some may result in simpler logic than others.





K-map version

The state table can now be redrawn with the selected state assignment:

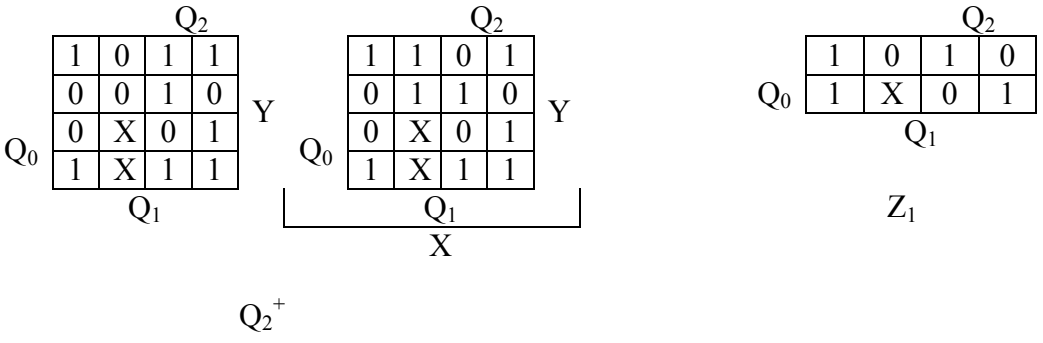
PS	NS				Output
	XY = 00	XY = 01	XY = 10	XY = 11	Z_1Z_2
000	101	000	100	001	10
101	101	111	111	110	11
100	110	000	101	001	01
010	000	010	111	110	00
001	101	000	100	001	11
111	111	001	111	000	00
110	110	111	001	101	10

We can now rearrange the rows so they are in order (making it easier to transcribe the next states to the individual K-maps for each flip flop)

PS	NS				Output
	XY = 00	XY = 01	XY = 10	XY = 11	Z_1Z_2
000	101	000	100	001	10
001	101	000	100	001	11
010	000	010	111	110	00
011	XXX	XXX	XXX	XXX	XX
100	110	000	101	001	01
101	101	111	111	110	11
110	110	111	001	101	10
111	111	001	111	000	00

Note that each flip-flop's next state is a function of the present state (three bits) and the present input (two bits), requiring five variable K-maps. The two outputs are each a function of the present state only (three bits).

The first next state bit, Q_2^+, and the first output, Z_1, are shown below.



Q_2^+