Synthesis of SIC fundamental-mode asynchronous circuits

Motivation

- Theese are sequential m/cs working without a clock
- Challenges of operating a clock are avoided
- The m/c stays in a stable state for a given set of inputs called the input state
- In a particular stable state the m/c produces some stable output
- When there is a change in the inputs the m/c responds by changing its state
- An important assumption that will be made is that only one of the inputs change at a time, i.e. single input change (SIC)
- It is further assumed that there is no change in the input until the m/c has made a transition to another stable state corresponding to the new input state
- This is called the fundamental mode of operation

Flow table

- Consider a sequential asynchronous m/c with two inputs, x_1 and x_2 and one output z
- The primitive flow table is shown below

m/c state, output				
input state (x_1, x_2)				
00	01	11	10	
1,0	2	—	4	
1	**2,0**	3	—	
—	2	**3,1**	4	
1	—	5	**4,0**	
—	2	**5,0**	4	

- Its components are now explained
  - The stable state and the corresponding outputs are indicated in bold
  - Note that there may be multiple stable states for a given input
  - The input states are depicted so that they are adjacent to each other (as in a KMap table), as on change of a single input bit, the new input state will be adjacent to the old input state
  - Some of the entries are vacant because those are not adjacent to the stable state and are not directly reachable from the current stable state for a change in the input state
  - The other entries indicate unstable states (with the outputs not yet defined) to which a transition is made when there is a change in the input state
  - Such a flow table where many things still remain undefined is called a *primitive flow table*
  - An important point to note is that each row has only a single stable state, accordingly, that state may be considered the present state and the table can be presented exactly as the state transition table of a partially specified FSM

M_1^A	m/c state, output			
PS	input state (x_1, x_2)			
	00	01	11	10
1	**1,0**	2	—	4
2	1	**2,0**	3	—
3	—	2	**3,1**	4
4	1	—	5	**4,0**
5	—	2	**5,0**	4

The given flow table may be traced as follows:

- Let's start at the input state 00 and the stable m/c state is 1 when its output is 0
- The input may change to either 01 or 10
- When the input changes to 01, the new state is 2 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 2 and has output 0
- If the input changes to 10, the new state is 4 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 4 and has output 0

- Let's continue from the input state 01 and the stable m/c state is 2 when its output is 0
- The input may change to either 00 or 11
- When the input changes to 11, the new state is 3 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 3 and has output 1
- If the input changes to 00, the new state is 1 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 1 and has output 0

- Similarly, when the input state 10, the only stable m/c state is 4 when its output is 0
- The input may change to either 00 or 11
- When the input changes to 11, the new state is 5 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 5 and has output 0
- If the input changes to 00, the new state is 1 (as yet unstable, so change in the input is permitted now)
- The m/c then stabilises to the m/c state 1 and has output 0

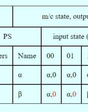
- When the input state 11, the stable m/c state may be 3 with output 1 or 5 with output 0
- In either case, if the input changes to 01 the new unstable state is 2 changing over to the stable state 2 with output 0
- If the input changes to 10 the new unstable state is 4 changing over to the stable state 4 with output 0

Along the earlier lines of minimisation of incompletely specified FSMs, the merger graph and the resulting reduced m/cs for two state mergings are shown below

Reduction of flow tables

- The undefined entries essentially give rise to an incompletely specified FSM

Merger graph for M1A



- The minimisation technique developed for minimising incompletely specified FSMs may be applied here also

M_1^A	m/c state, output			
PS	input state (x_1, x_2)			
	00	01	11	10
1	**1,0**	2	—	4
2	1	**2,0**	3	—
3	—	2	**3,1**	4
4	1	—	5	**4,0**
5	—	2	**5,0**	4

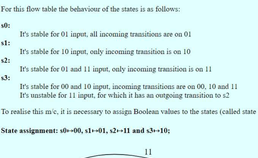
M_{1}^{Ar}	m/c state, output				
PS		input state (x_1, x_2)			
Members	Name	00	01	11	10
1,2,3	α	α,0	α,0	α,1	β,0
4,5	β	α,0	α,0	β,0	β,0

M_{1}^{Ar}	m/c state, output				
PS		input state (x_1, x_2)			
Members	Name	00	01	11	10
2,3	α	β,0	α,0	α,1	β,0
1,4,5	β	β,0	α,0	β,0	β,0

- Two possible mergings have been depicted in the tables for M_1^{Ar} and M_1^{Ar}
- The outputs marked red had been unspecified after merging the states; those have been set to match the outputs of the stable state
- Any choice made should avoid glitching
  - for M_1^{Ar}, z=1 for PS=β and inputs 00 and 01 produce static-0 hazards
  - for M_1^{Ar}, z=1 for PS=α and input 10 produces a static-0 hazard
  - similar hazards are also possible for M_1^{Ar}
- If desired, while avoiding glitching, the output may be achieve a fast or a slow transition to the output
- Thereafter, the problem of state encoding needs to be addressed

State assignment

Consider the following flow graph



For this flow table the behaviour of the states is as follows:

s0:
It's stable for 01 input, all incoming transitions are on 01

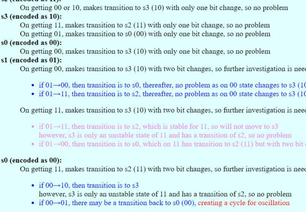
s1:
It's stable for 10 input, only incoming transition is on 10

s2:
It's stable for 01 and 11 input, only incoming transition is on 11

s3:
It's stable for 00 and 10 input, incoming transitions are on 00, 10 and 11
It's unstable for 11 input, for which it has an outgoing transition to s2

To realise this m/c, it is necessary to assign Boolean values to the states (called state encoding) and the behaviour of the m/c with that state assignment needs to be analysed

State assignment: s0=00, s1=01, s2=11 and s3=10;



s2 (encoded as 11):
On getting 00 or 10, makes transition to s3 (10) with only one bit change, so no problem
s3 (encoded as 10):
On getting 11, makes transition to s2 (11) with only one bit change, so no problem
On getting 01, makes transition to s0 (00) with only one bit change, so no problem
s0 (encoded as 00):
On getting 00, makes transition to s3 (10) with only one bit change, so no problem
s1 (encoded as 01):
On getting 00, makes transition to s3 (10) with two bit changes, so further investigation is needed

- if 01→00, then transition is to s0, thereafter, no problem as on 00 state changes to s3 (10) with only one bit change
- if 01→11, then transition is to s2, thereafter, no problem as on 00 state changes to s3 (10) with only one bit change

On getting 11, makes transition to s3 (10) with two bit changes, so further investigation is needed

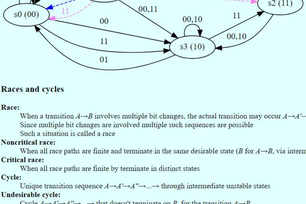
- if 01→11, then transition is to s2, which is stable for 11, so will not move to s3
  however, s3 is only an unstable state of 11 and has a transition of s2, so no problem
- if 01→00, then transition is to s0, which on 11 has transition to s2 (11) but with two bit changes, so more investigation is needed

s0 (encoded as 00):
On getting 11, makes transition to s2 (11) with two bit changes, so further investigation is needed

- if 00→10, then transition is to s3
  however, s3 is only an unstable state of 11 and has a transition of s2, so no problem
- if 00→01, there may be a transition back to s0 (00), creating a cycle for oscillation

- This state assignment is, therefore, problematic
- A difference state assignment may be attempted
- It is, nevertheless useful to note the transition from s0 to s2 on 11 may be modified to s2 (10), instead
- This change would solve the problem
- On similar lines, the transitions from s1 (01) to s3 (10) on 00 and 11 could be modified to s2
- However, for this state assignment that doesn't create a problem for those transitions
- What is evident is that *if there is a transtion from one state to another, it's better if their codes are adjacent*

The modified flow graph is shown below



Races and cycles

Race:
When a transition $A \dashrightarrow B$ involves multiple bit changes, the actual transition may occur $A \dashrightarrow A' \dashrightarrow A'' \dashrightarrow ...$
Since multiple bit changes are involved multiple such sequences are possible
Such a situation is called a race
Noncritical race:
When all race paths are finite and terminate in the same desirable state (B for $A \dashrightarrow B$, via intermediate states), the race is noncritical
Critical race:
When all race paths are finite by terminate in distinct states
Cycle:
Unique transition sequence $A \dashrightarrow A' \dashrightarrow A'' \dashrightarrow ...\dashrightarrow$ through intermediate unstable states
Undesirable cycle:
Cycle $A \dashrightarrow A' \dashrightarrow A'' \dashrightarrow ...\dashrightarrow$ that doesn't terminate on B, for the transition $A \dashrightarrow B$
Valid state assignment:
A state encoding that does not lead to critical races or undesirable cycles

Exercises

- For the above m/c, construct the next state and ouput functions
- Using the methods for hazard free circuit design, realise hazard free circuits for these
- Design an asynchronous sequential circuit with two inputs x and y and an output z so that whenever $y = 1$, $z = x$ but when $y = 0$, z retains its older state and does not change with x
- Give a minimum-row reduced-flow-table description of an SIC fundamental-mode two-input (x_1, x_2), one-output (z) sequential circuit that operates in the following manner: the output $z = 1$ if and only if the input state $x_1 = x_2 = 1$ and the next-to-last input variable change was a change in the value of x_1. Assume that the circuit is initially in the input state $x_1 = x_2 = 0$. Is the reduced flow table unique?
- At a junction of a single-track railroad and a road, traffic lights are to be installed. The lights are to be controlled by switches that are pressed or released by the trains. When a train approaches the junction from either direction and is within 1500m from it, the lights are to change from green to red and remain red until the train is 1500m past the junction.
- A completely automatic and independent traffic-light system for the intersection of roads x and y consists of two sensors, some processing circuitry, and the lights. The sensors and circuitry generate two outputs, z and w. Output z attains the value 1 if and only if $m(x) - m(y) \geq 6$, where $m(x)$ indicates the number of cars waiting to cross a road y. Output w attains the value 1 if and only if $m(y) - m(x) \geq 6$. We wish to design an SIC fundamental-mode sequential circuit with inputs (z, w, z', w') and outputs (G_x, R_x, G_y, R_y), where G and R refer to green and red lights, respectively, and the subscripts indicate the street from which the light is visible. The objective is to minimize intersection load by unloading whichever street is overloaded, i.e., has at least six cars more than the other. The lights of the street being unloaded should remain green until the other street becomes overloaded.
  a. Show a primitive flow table
  b. Give a reduced flow table
  c. Show a circuit realization; the outputs are to be fast and flicker-free
- Consider an arbiter with two high active inputs request lines a and b and two grant lines x and y. In case of exclusive request, the corresponding grant line is asserted.

  In the event of both request lines being active, there is no immediate change in the grant. However, a monochist is triggered (asserting $t = 1$ with z going high as a result and then resetting it, i.e. setting $t = 0$, thereafter).

  After timeout (z going from $1 \rightarrow 0$), if the same line has the grant at the time of triggering, the grant is switched and the arbiter continues as usual.

  Design the arbiter to operate asynchronously, as described above