Asynchronous Sequential Circuits

Zvi Kohavi and Niraj K. Jha

Definitions

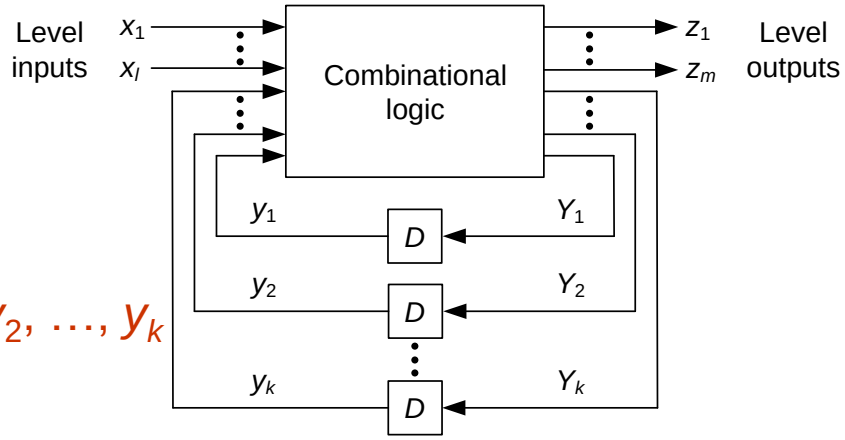
Asynchronous circuits: within large synchronous systems, it is often desirable to allow certain subsystems to operate asynchronously to reduce delay and power consumption

Total state: combination of signals that appear at the primary input and delay outputs

Input state: combination of input signals

$x_1, x_2, \ldots, x_l$

Secondary or internal state: combination of signals at the delay outputs $y_1, y_2, \ldots, y_k$



Secondary or internal variables: $y_1, y_2, \ldots, y_k$

Excitation variables: $Y_1, Y_2, \ldots, Y_k$

Modes of Operation

Stable state: for a given input state, the circuit is said to be in a stable state if and only if $y_i = Y_i$ for $i = 1, 2, \ldots, k$

- In response to a change in the input state: the combinational logic produces a new set of values for the excitation variables, entering an unstable state
- When the secondary variables assume their new values (when *y*'s become equal to the corresponding *Y*'s): the circuit enters its next stable state
  - Thus, a transition from one stable state to another occurs only in response to a change in the input state

Fundamental mode: when a change in input values has occurred, no other change in any input value occurs until the circuit enters a stable state

- Single-input change (SIC) fundamental mode: a single input value is allowed to change at a time
- Multiple-input change (MIC) fundamental mode: multiple input values can change at a time

Hazards

Two type of hazards (glitches): logic and function

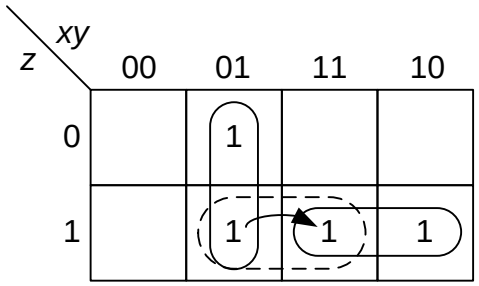
- Logic hazards: caused by noninstantaneous changes in circuit signals
- Function hazards: inherent in the functional specification

Hazards pose a fundamental problem: a glitch may be misunderstood by another part of the circuit as a valid transition and cause incorrect behavior

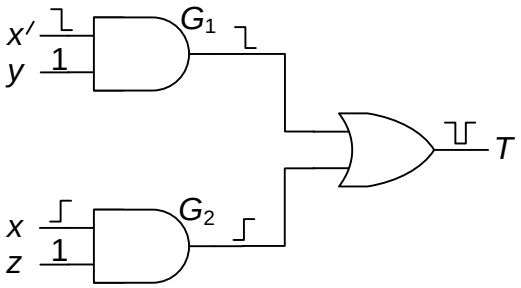
Design of SIC Hazard-free Circuits

Example: $T(x,y,z) = \sum(2,3,5,7)$

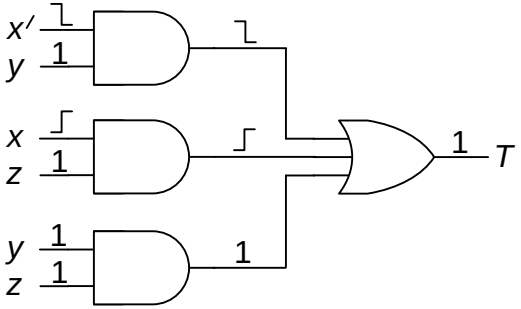
- Static-1 logic hazard (SIC)



(*a*) Map for $T = x'y + xz$.



(*b*) Gate network.



(*c*) SIC hazard-free network

Adjacent combinations: differ in the value of a single variable

- E.g., *x'yz* and *xyz*

SIC static logic hazard: transition between a pair of adjacent input combinations, which correspond to identical output values, that may generate a momentary spurious output value

- Occurs when no cube in the K-map contains both combinations
  - Solution: cover both combinations with a cube

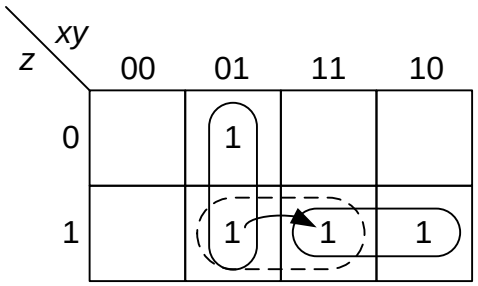
Transition/Required Cubes

Transition cube [m_1,m_2]: set of all minterms that can be reached from minterm m_1 and ending at minterm m_2

Example: Transition cube [010,100] contains: 000, 010, 100, 110

Required cube: transition cube that must be included in some product of the sum-of-products realization in order to get rid of the static-1 logic hazard

Example: Required cube is [011,111]



Static-0/Dynamic Hazard

Since in the sum-of-products realization of a function: no cube for any product term can contain either of the two input combinations involved in a 0->0 output transition, a static-0 logic hazard can only occur if a product term has both x_i and x_i' as input literals

- Since there is no need to include such products: such hazards can be trivially avoided

During a 0->1 output transition: if the 0 may change to 1 and then 0 and finally stabilize at 1, then the sum-of-products realization is said to have a dynamic 0->1 logic hazard

- Dynamic 1->0 logic hazard is similarly defined

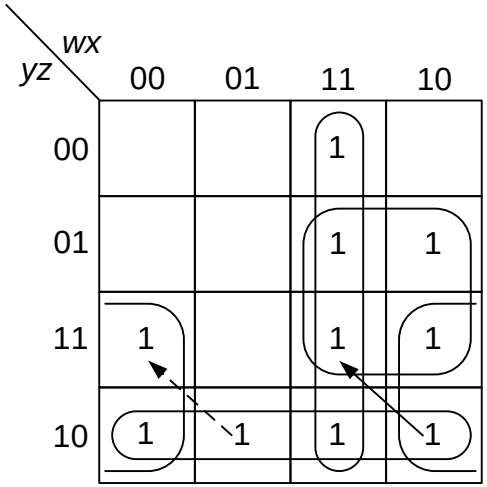
Based on above reasoning: a dynamic 0->1 and 1->0 logic hazard is also trivially avoidable in the SIC scenario

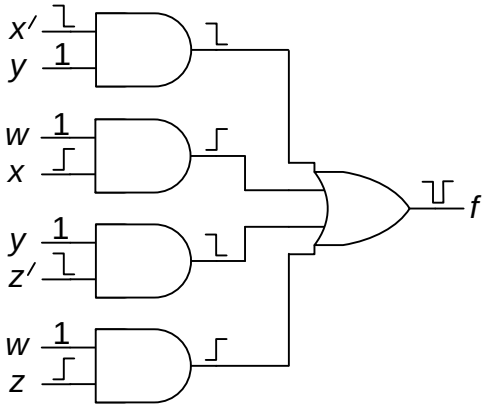
Design of MIC Hazard-free Circuits

MIC scenario: several inputs change values monotonically, i.e., at most once

- If in this process, the function changes values more than once: the transition is said to have a function hazard

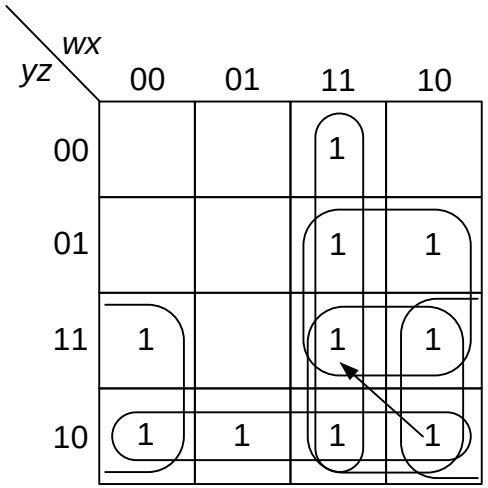
Example: Function hazard: dotted arrow; static-1 logic hazard: solid arrow

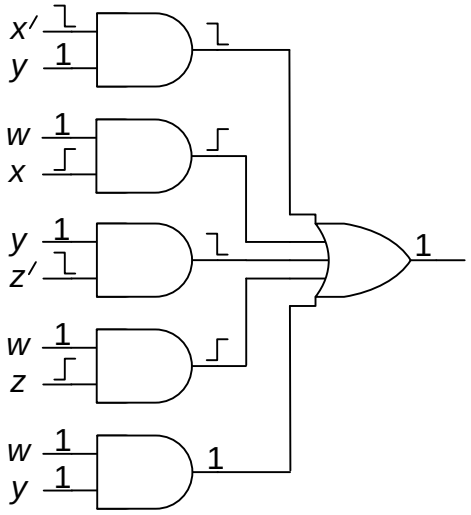




Getting Rid of the Static-1 Logic Hazard

Example (contd.): cover the solid arrow with a cube to get rid of the static-1 logic hazard

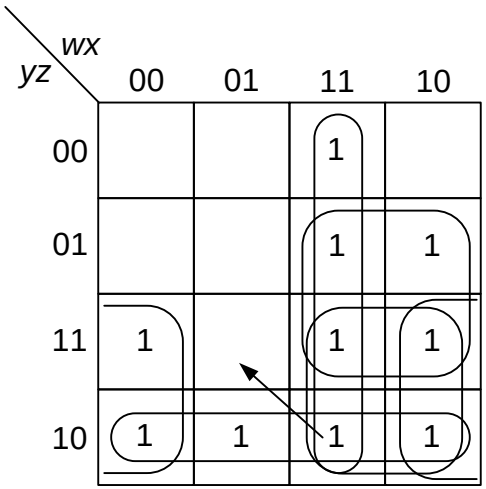


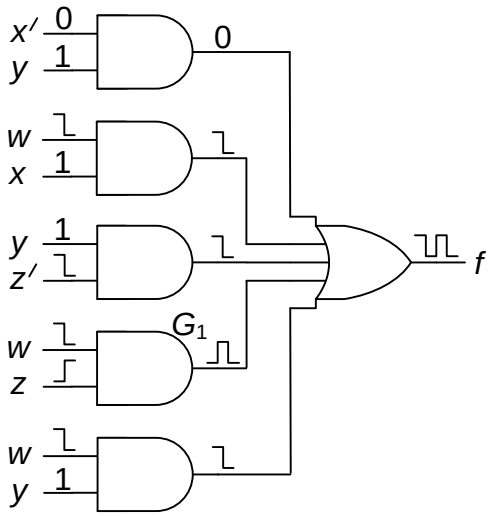


Avoiding a static-0 logic hazard is trivial: just as in the SIC case

MIC Dynamic Hazards

Example: solid arrow





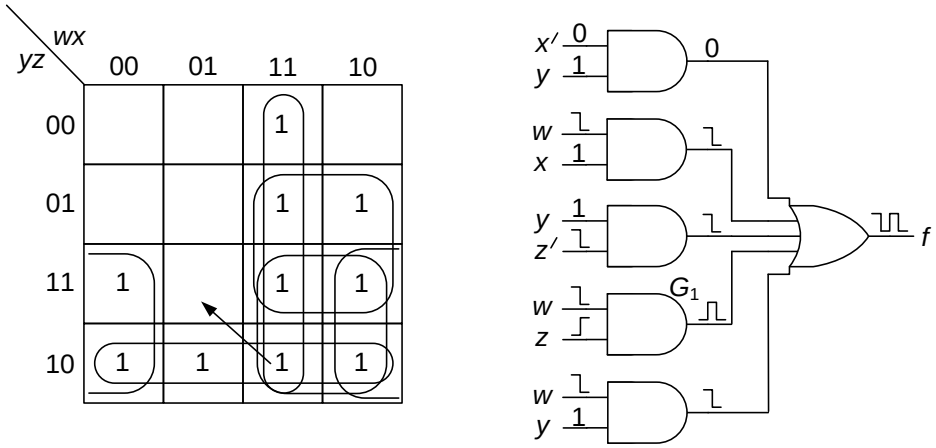
Necessary condition for the dynamic transition to be hazard-free

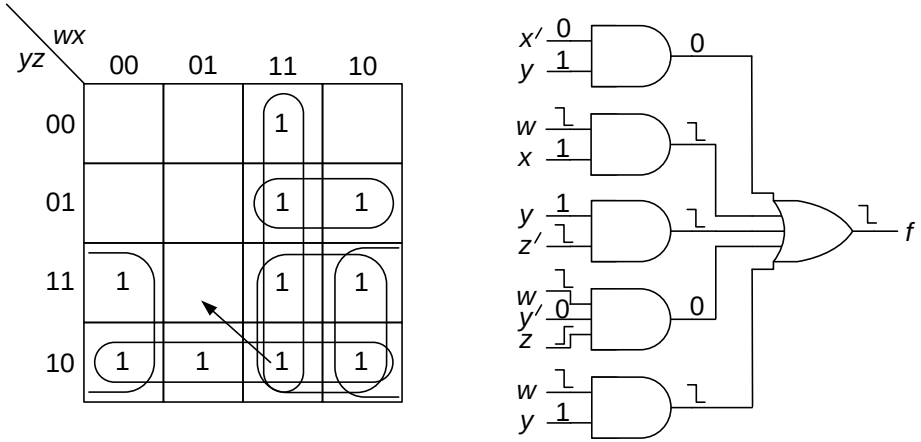
- Make sure each of its 1->1 subtransitions is also hazard-free: ensured by including these subtransitions in some product of the sum-of-products realization
- Subtransitions: [1110,1111], [1110,0110] – called required cubes of the dynamic transition
  - Necessary condition met in this example for these required cubes

Getting Rid of the Dynamic Hazard

Ensure that no AND gate turns on during the MIC transition

- G_1 temporarily turns on because product *wz* intersects the dynamic transition 1110 -> 0111: called illegal intersection
- Dynamic transition called a privileged cube
- During this transition: inputs could be momentarily at 1111, which is a minterm of *wz*
- Disallow illegal intersections of privileged cubes: reduce *wz* to *wy'z*





Eliminating Hazards for an MIC Transition

1->1 MIC transition: must be completely covered by a product term

0->0 MIC transition: does not lead to a hazard

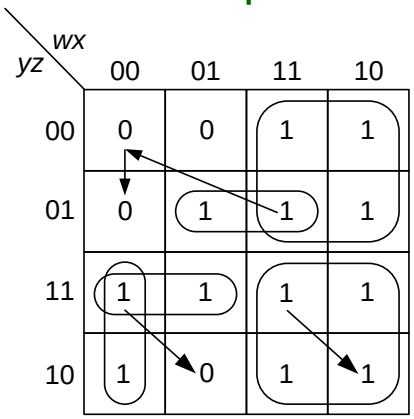
1->0 (0->1) MIC transition: ensure that every product term that intersects the MIC transition also contains its starting (end) point

To obtain a hazard-free sum-of-products implementation *H* of function *f*, ensure:

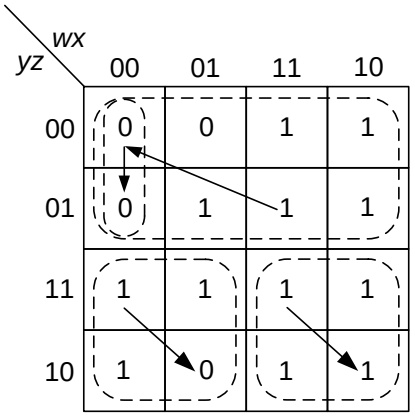
- Each required cube is contained in some implicant of *H*
- No implicant of *H* illegally intersects any specified dynamic transition
  - Such an implicant is called a dynamic-hazard-free implicant (dhf-implicant)
  - A dhf-prime implicant is a dhf-implicant not contained in any other dhf-implicant
- This problem requires that we only make use of dhf-prime implicants while covering every required cube in sum-of-products minimization
  - Similar to Quine-McCluskey minimization

Deriving a Hazard-free Sum-of-products

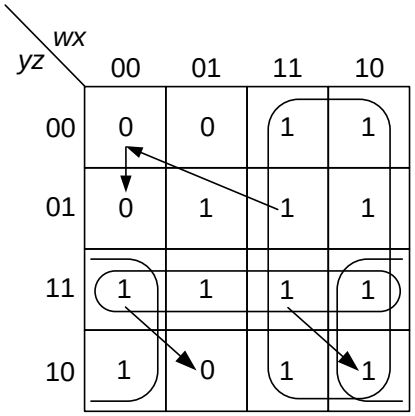
Example:



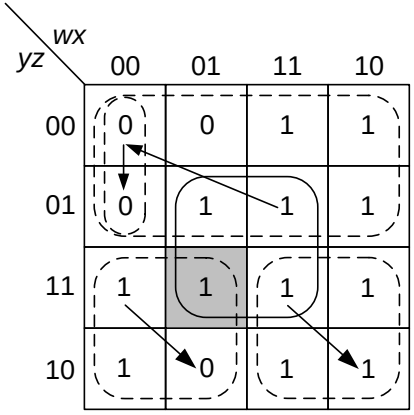
(*a*) Required cubes.



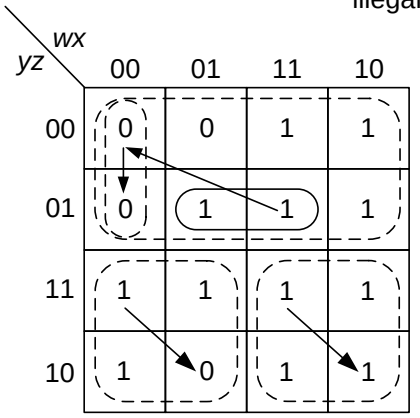
(*b*) Privileged cubes.



(*c*) Prime implicants with no illegal intersections.



(*d*) Prime implicant *xz* has an illegal intersection.



(*e*) Prime implicant *xz* reduced to dhf-prime implicant *xy'z*.

dhf-prime implicants	Required cubes				
	wy'	wy	$xy'z$	$w'yz$	$w'x'y$
w	×	×			
yz				×	
$x'y$					×
$xy'z$			×		

Hazard-free sum-of-products:

w + yz + x'y + xy'z

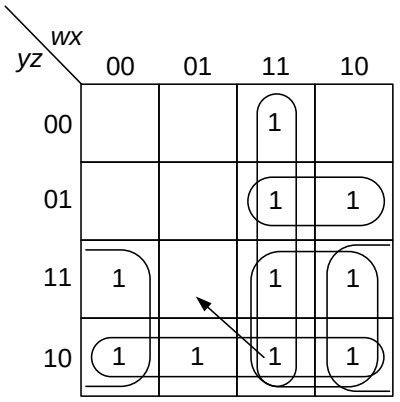
Hazard-non-increasing Logic Transformations

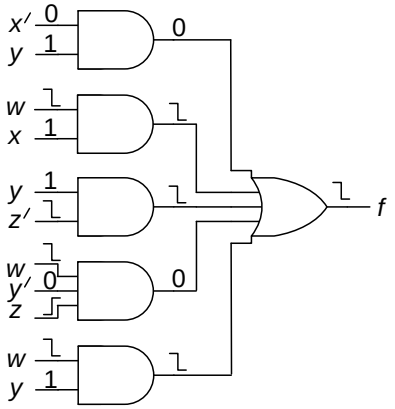
Hazard-non-increasing logic transformations: used to derived hazard-free multi-level realization from hazard-free two-level realization

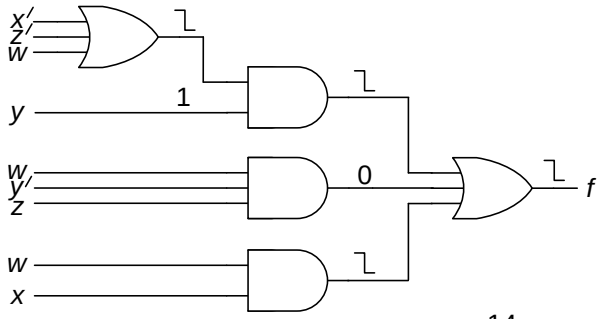
- If the initial circuit is hazard-free: so is the final multi-level circuit
- Associative law and its dual: $(x + y) + z \Leftrightarrow x + (y + z)$; $(xy)z \Leftrightarrow x(yz)$
- De Morgan's theorem and its dual: $(x + y)' \Leftrightarrow x'y'$; $(xy)' \Leftrightarrow x' + y'$
- Distributive law: $xy + xz \Rightarrow x(y + z)$
- Absorption law: $x + xy \Rightarrow x$
- $x + x'y \Rightarrow x + y$ law
- Insertion of inverters at primary inputs and circuit output

Example: AND-OR realization free of dynamic hazard for 1110 -> 0111

- So is the multi-level realization: $x'y + wx + yz' + wy'z + wy = (x' + z' + w)y + wx + wy'z$



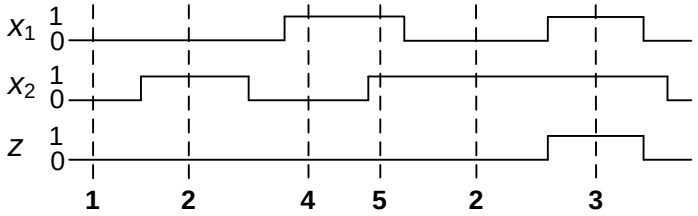




Synthesis of SIC Fundamental-mode Circuits

Flow table: analogous to the state table

Example: Consider a sequential circuit with two inputs x_1 and x_2 and one output z. The initial input state is $x_1 = x_2 = 0$. The output value is to be 1 if and only if the input state is $x_1 = x_2 = 1$ and the preceding input state is $x_1 = 0$, $x_2 = 1$



Input-output sequences

State, output				
x_1x_2				
00		01	11	10
1,0	→	2		
		↓		
		2,0	→ 3	
			↓	
			3,1	

Partial flow table

State, output			
x_1x_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

Primitive flow table

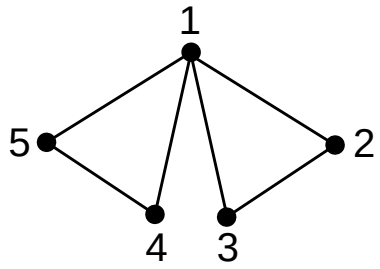
Reduction of Flow Tables

Reduction of primitive flow table has two functions:

- Elimination of redundant stable states
- Merging those stable states which are distinguishable by input states

Example: Rewrite primitive flow table like a state table

PS	*State, output* x_1x_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



Merger graph

Maximal compatibles: {(123), (145)}

Reduced flow tables

State, output x_1x_2 00	01	11	10
1,0	**2**,0	**3**,1	4,0
1,0	2,0	**5**,0	**4**,0

State, output x_1x_2 00	01	11	10
1,0	2,0	**5**,0	**4**,0
1,0	**2**,0	**3**,1	4,0

(*a*) Closed covering {(123),(45)}. (*b*) Closed covering {(145),(23)}.

Specifying the Output Symbols

Assignment of output values to the unstable states in the reduced flow table

- When the circuit is to go from one stable state to another stable state associated with the same output value: assign the same output value to the unstable state en route to avoid a momentary opposite value
- When the state changes from one stable state with a given output value to another stable state with a different output value: the transition may be associated with either of these output values
  - When the relative timing of the output value change is of no importance: choose the output value so as to minimize logic

State, output			
x_1x_2			
00	01	11	10
1,0	2	**3**,0	4
1	**2**,1	3	**4**,0
5,1	6	**7**,1	8
5	**6**,0	7	**8**,0

(*a*) Reduced flow table.

State, output			
x_1x_2			
00	01	11	10
1,0	2,1	**3**,0	4,0
1,0	**2**,1	3,0	**4**,0
5,1	6,0	**7**,1	8,0
5,1	**6**,0	7,1	**8**,0

(*b*) Reduced flow table with output values specified.

Excitation and Output Tables

Example: Reduced flow table

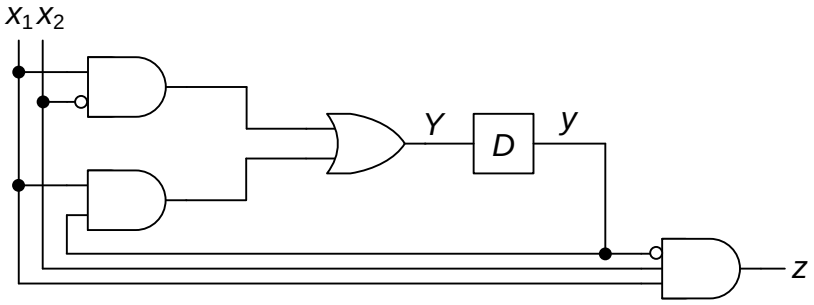
State, output			
x_1x_2			
00	01	11	10
1,0	**2**,0	**3**,1	4,0
1,0	2,0	**5**,0	**4**,0

Excitation and output table

	Y, z			
y	x_1x_2			
	00	01	11	10
0	0,0	0,0	0,1	1,0
1	0,0	0,0	1,0	1,0

$Y = x_1x_2' + x_1y$

$z = x_1x_2y'$



Synthesis Procedure

Synthesis procedure for SIC fundamental-mode asynchronous circuits:

1. Construct a primitive flow table from the verbal description: specify only those output values that are associated with stable states
2. Obtain a minimum-row reduced flow table: use either the merger graph or merger table for this purpose
3. Assign secondary variables to the rows of the reduced flow table and construct excitation and output tables: specify output values associated with unstable states according to design requirements
4. Derive excitation and output functions, and the corresponding hazard-free circuit

Synthesis Example

Example: Design an asynchronous sequential circuit with two inputs, x_1 and x_2, and two outputs, G and R, as follows.

- Initially, both input values and both output values are 0
- Whenever $G = 0$ and either x_1 or x_2 becomes 1, G becomes 1
- When the second input becomes 1, R becomes 1
- The first input value that changes from 1 to 0 turns G equal to 0
- R becomes 0 when G is 0 and either input value changes from 1 to 0

	State, GR			
x_1x_2	00	01	11	10
00	**1**,00			
01		**2**,10		
		3,01		
11			**4**,11	
10				**5**,10
				6,01

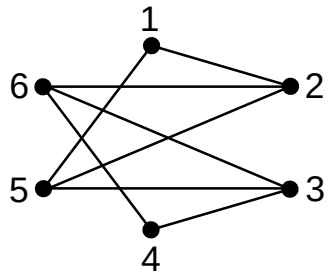
(*a*) Table containing only stable states.

	State, GR			
x_1x_2	00	01	11	10
00	**1**,00	2	–	5
01	1	**2**,10	4	–
	1	**3**,01	4	–
11	–	3	**4**,11	6
10	1	–	4	**5**,10
	1	–	4	**6**,01

(*b*) Completed primitive flow table.

Synthesis Example (Contd.)

Merger graph



Reduced flow table

State, GR			
x_1x_2			
00	01	11	10
1,00	**2**,10	4,11	**5**,10
1,01	**3**,01	**4**,11	**6**,01

Excitation and output table

	Y, GR			
y	x_1x_2			
	00	01	11	10
0	0,00	0,10	1,11	0,10
1	0,01	1,01	1,11	1,01

$Y = (x_1 + x_2)y + x_1x_2$

$G = (x_1 + x_2)y' + x_1x_2$

$R = y + x_1x_2$

Races and Cycles

Assignment of the secondary variable values to the rows of the reduced flow table should be such that: the circuit will operate correctly even if different delays are associated with the secondary elements

Race: where a change of more than one secondary variable is required

- Noncritical race: the final state does not depend on the order in which the secondary variables change
- Critical race: the final state reached depends on the order in which the secondary variables change – must always be avoided
- Races can sometimes be avoided by directing the circuit through intermediate unstable states
  - Cycle: circuit goes through a unique sequence of unstable states

y_1y_2	x_1x_2 / Y_1Y_2 00	01	11	10
00	11	**00**	10	01
01	11	00	11	**01**
11	**11**	00	10	**11**
10	11	**10**	**10**	11

Illustration of races and cycles

y_1y_2	x_1x_2 / Y_1Y_2 00	01	11	10
00	10	**00**	10	01
01	10	00	11	**01**
10	**10**	00	11	**10**
11	10	**11**	**11**	10

Valid assignment that eliminates critical races

Methods of Secondary Assignment

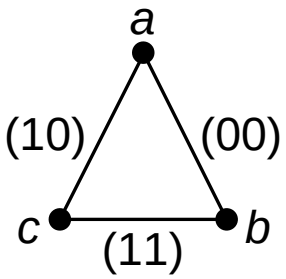
Valid state assignment: avoids critical races and undesired cycles

Adjacent states: states whose assignments differ in only one variable

Example: Flow table

PS	State x_1x_2 00	01	11	10
a	**1**	3	**4**	**6**
b	1	3	**5**	**7**
c	**2**	**3**	5	6

Transition diagram



Column 00: Row *b* must be adjacent to row *a*

Column 01: Rows *a* and *b* must be adjacent to row *c*

Column 11: Row *c* must be adjacent to row *b*

Column 10: Row *c* must be adjacent to row *a*

If noncritical races are permitted: column 01 requirement may be eliminated

Secondary Assignment (Contd.)

Avoiding all races: not possible when the transition diagram is a triangle

- Use augmented flow table

y_1y_2	x_1x_2 = 00	01	11	10
	Y_1Y_2			
$a \longrightarrow 00$	**00**	10	**00**	**00**
$b \longrightarrow 01$	00	11	**01**	**01**
$c \longrightarrow 11$	**11**	**11**	01	10
$c \longrightarrow 10$	**10**	**10**	11	00

(a) Two assignments to row c.

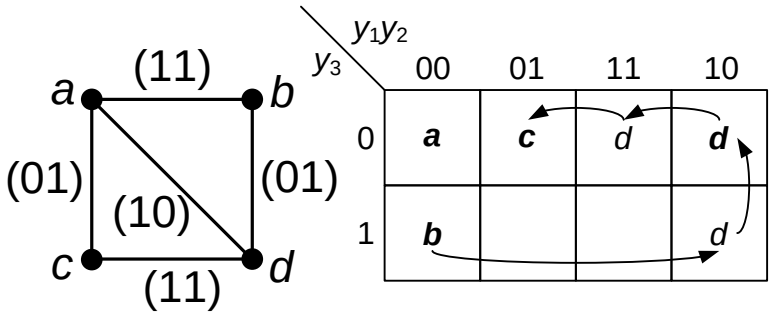
y_1y_2	x_1x_2 = 00	01	11	10
	Y_1Y_2			
$a \longrightarrow 00$	**00**	01	**00**	**00**
$b \longrightarrow 01$	00	11	**01**	**01**
$c \longrightarrow 11$	**11**	**11**	01	10
10	--	--	--	00

(b) Utilizing an unspecified entry as an unstable state.

Example: Augmentation of the flow table may require an increase in the number of secondary variables

PS	State x_1x_2 = 00	01	11	10
a	**1**	**2**	4	**6**
b	1	3	**4**	**7**
c	1	2	**5**	**8**
d	1	**3**	5	6

Flow table



Transition diagram

$y_1y_2y_3$	State x_1x_2 = 00	01	11	10
$a \longrightarrow 000$	**1**	**2**	4	**6**
$b \longrightarrow 001$	1	3	**4**	**7**
011				
$c \longrightarrow 010$	1	2	**5**	**8**
110			5	
111				
101		3		
$d \longrightarrow 100$	1	**3**	5	6

Race-free flow table

Synthesis of Burst-mode Circuits

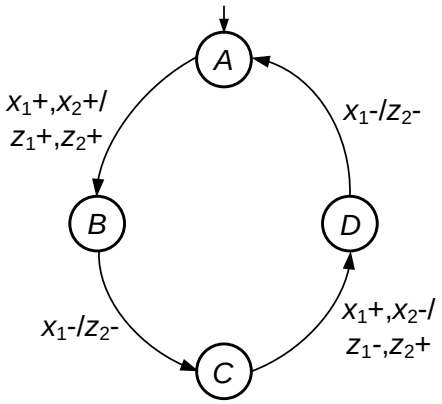
MIC fundamental-mode machines: several inputs change in a narrow time interval and no further inputs change values until the machine has stabilized

- Narrow time interval: still quite restrictive

Burst-mode machines: also allow several inputs to change values concurrently

- However, all the changes need not occur in a narrow time interval
- They can monotonically change in any order at any time within a given input burst and respond with a set of output value changes, called the output burst

Burst-mode specification: initial values of inputs and outputs can be specified or just assumed to have a default value of 0



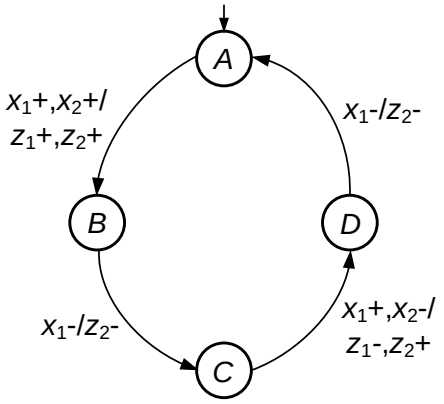
Burst-mode Specification Restrictions

Restrictions on burst-mode specifications:

- Non-empty input bursts: if no input undergoes a transition, the machine remains in its current state
- Maximal set property: no input burst on an outgoing arc from any state must be a subset of an input burst on another outgoing arc from the same state
- Unique entry point: each state should have a unique set of input and output values through which it is entered

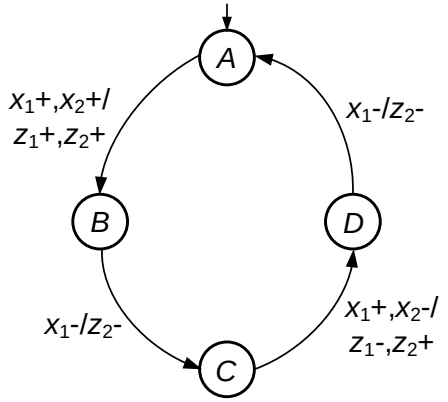
Example: Assume in starting state *A*, $x_1x_2 = 00$ and $z_1z_2 = 00$

- *B*: 11/11
- *C*: 01/10
- *D*: 10/01



Flow Table

Example (contd.): Specification



Flow table

PS	$State,\ z_1z_2$ x_1x_2 00	01	11	10
A	A,00	A,00	B,11	A,00
B	–	C,10	B,11	–
C	C,10	C,10	C,10	D,01
D	A,00	–	–	D,01

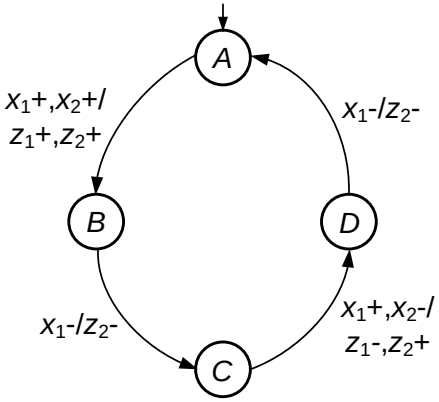
Complete state: the state the machine goes to and corresponding output values

Flow table for a burst-mode specification does not have any function hazards: since the complete state does not change until the full input burst has arrived

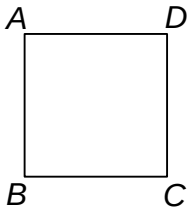
- It is always possible to obtain a hazard-free sum-of-products realization *H* for each secondary variable and output: since for each such variable, the required cube can be included in some product of *H* and no product of *H* illegally intersects any privileged cube because all transitions in any row of the flow table have the same complete start state which will be included in the required cubes for these transitions

Synthesis Example

Example: Specification



Transition diagram



State assignment

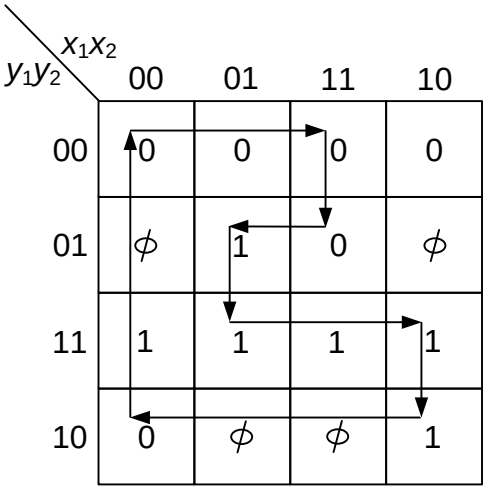
y_2 \ y_1	0	1
0	A	D
1	B	C

Excitation and output table

y_1y_2	Y_1Y_2, z_1z_2			
	x_1x_2 00	01	11	10
00	00,00	00,00	01,11	00,00
01	–	11,10	01,11	–
11	11,10	11,10	11,10	10,01
10	00,00	–	–	10,01

Synthesis Example (Contd.)

Y_1, Y_2:

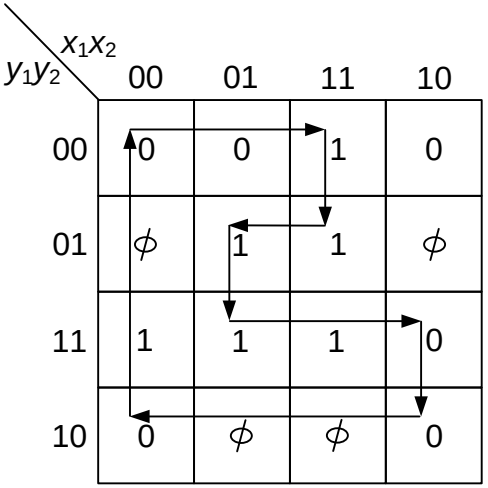


Y_1 map

dhf-prime implicants	Required cubes		
	$x_1'x_2y_2$	y_1y_2	$x_1x_2'y_1$
$x_1'y_2$	×		
$x_2'y_2$			
y_1y_2		×	
x_1y_1			×
x_2y_1			

Minimal hazard-free sum-of-products

$Y_1 = x_1'y_2 + y_1y_2 + x_1y_1$



Y_2 map

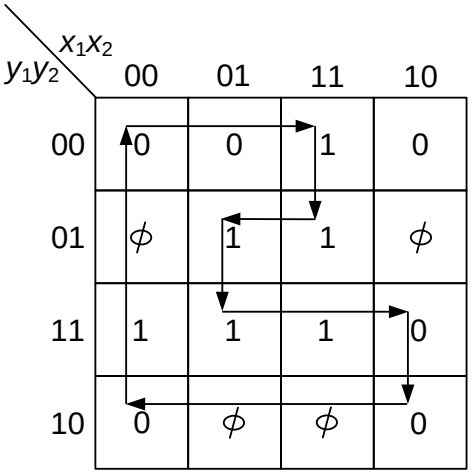
dhf-prime implicants	Required cubes				
	$x_1x_2y_1'$	$x_2y_1'y_2$	$x_1'x_2y_2$	$x_2y_1y_2$	$x_1'y_1y_2$
$x_1x_2y_1'$	×				
$y_1'y_2$		×			
$x_1'y_2$			×		×
x_2y_2		×	×	×	
x_2y_1				×	

Minimal hazard-free sum-of-products

$Y_2 = x_1x_2y_1' + x_1'y_2 + x_2y_2$

Synthesis Example (Contd.)

z_1, z_2:

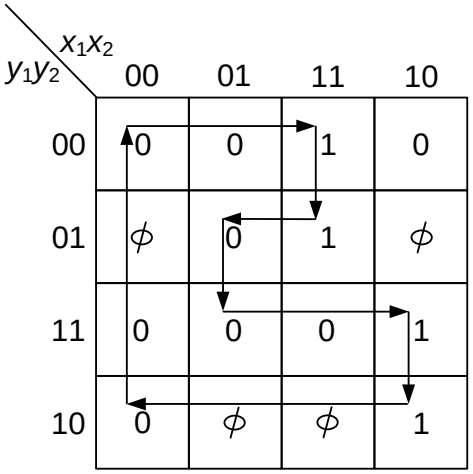


z_1 map

$\Longrightarrow$

Minimal hazard-free sum-of-products

$z_1 = Y_2 = x_1x_2y_1' + x_1'y_2 + x_2y_2$



z_2 map

$\Longrightarrow$

dhf-prime implicants	Required cubes	
	$x_1x_2y_1'$	$x_1x_2'y_1$
$x_1x_2y_1'$	×	
$x_1y_1'y_2$		
$x_1x_2'y_1$		×
$x_1y_1y_2'$		
$x_1x_2y_2'$		

Minimal hazard-free sum-of-products

$z_2 = x_1x_2y_1 + x_1x_2y_1'$

Synthesis Example (Contd.)

Synthesized circuit: