LECTURE 23

DIGITAL LOGIC FAMILIES

Primitive Flow Table

- The next step is to draw the state table giving the information in tabular form, i.e. the primitive flow table

Present State	Next State		Output Z
1	1	2	0
2	3	2	0
3	3	4	0
4	1	4	1

Flow Table

- Stable states are again indicated by circles around the stable state numbers in the Next State columns
  - 1, 2, 3, 4
  - Circled state will be the same as the number in the present state column.
- Output tries to attain to the stable state
- Primitive flow table should then be minimised where possible
  - no minimisation in this example.
- Secondary variables are now assigned.

Present State	Next State	Output Z
1	1	2	0
2	3	2	0
3	3	4	0
4	1	4	1

Assigning Secondary Variables

- Care must be taken not to make an assignment, which results in more than one variable change between states.

- Use a **transition table/map** which has states chosen for each square on the map

- Transitions from one state to another are marked on the map and if any show a diagonal path across two variable changes, a new assignment must be made.

Assigning Secondary Variables

The assigned flow table can then be written by inspection.

Present State y_1y_2	Next 0	State Y_1Y_2 1	Output Z
1	00	01	0
2	11	01	0
3	11	10	0
4	00	10	1

Swapping state assignments for 1 and 2 would result in an unsatisfactory map.

Circuit Implementation

- Two principal implementations possible

  1. **Purely combinational logic gates**
  2. **Combinational logic gates with asynchronous RS flip flops**.

- Historically, asynchronous sequential circuits were known and used before synchronous sequential circuits were developed

  - First practical digital systems were constructed with delays which were more adaptable to asynchronous type operations

  - For this reason, the traditional method of asynchronous sequential circuit configuration has been with components that are connected to form one or more feedback loops.

Circuit Implementation

- As electronic digital circuits were developed, it was realised that the flip-flop could be used as the memory element.
  - Use of RS-latch in asynchronous sequential circuits produces a more orderly pattern, which may result in a reduction of the circuit complexity.
  - An added advantage is that the circuit resembles the synchronous circuit in having distinct memory elements that store and specify the internal states.

- The RS-flip flip design approach assigns one flip-flop for each secondary variable.
  - The inputs to these flip-flops are determined by the required change of y to Y.

Circuit Implementation with RS Flipflops

- Using the following table

Required Change Q_t	Output To Q_{t+1}	Flip-flop Inputs S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

$\Rightarrow$ Obtain one function for each flip-flop input as shown below.

Circuit Implementation

S_2

y_1y_2 \ x	0	1
00	0	0
01	1	0
11	X	X
10	0	X

R_2

y_1y_2 \ x	0	1
00	X	X
01	0	X
11	0	0
10	1	0

S_1

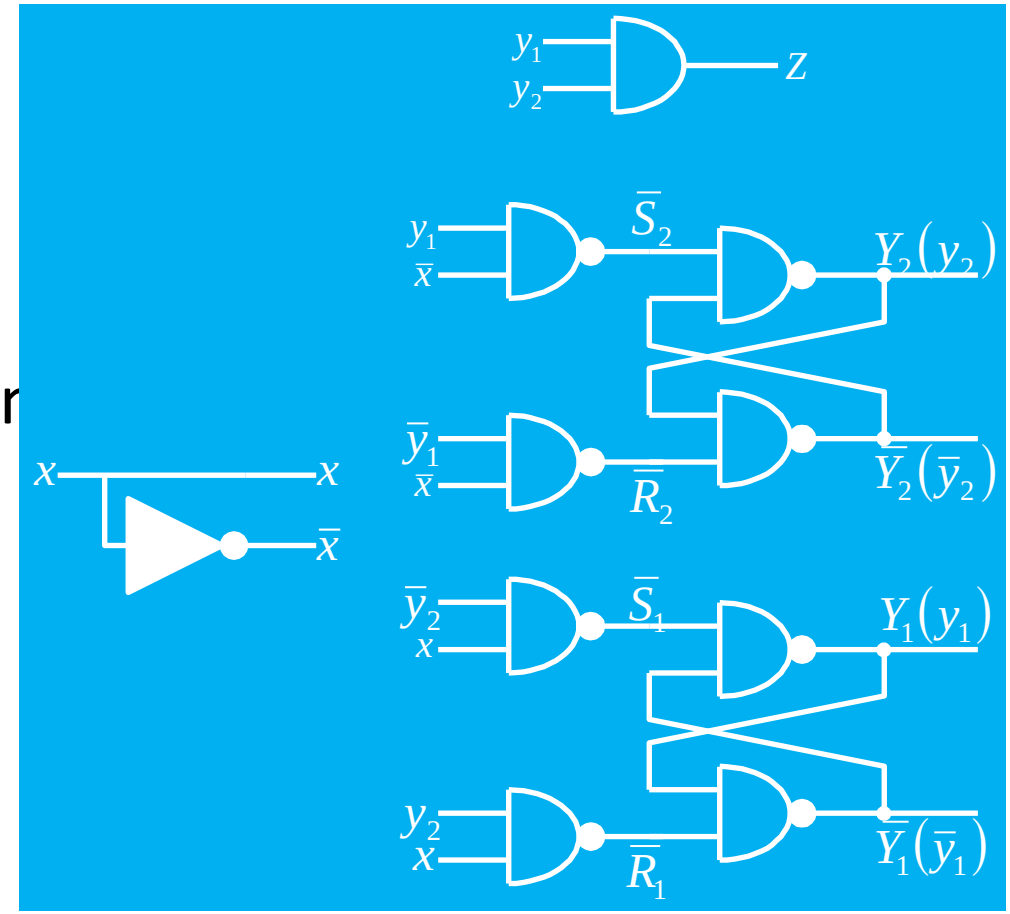
y_1y_2 \ x	0	1
00	0	1
01	X	X
11	X	0
10	0	0

R_1

y_1y_2 \ x	0	1
00	X	0
01	0	0
11	0	1
10	X	X

Circuit Implementation

The fir



Circuit Implementation

- A particular advantage of the RS flip-flop method is that it is not necessary to correct for static hazards
  - As all the prime implicants are present in both the set and reset functions, which will be the case in all problems.
  - Hence the RS flip-flop method often requires less components.

Circuit Implementation

- In the RS flip-flop method, both true and complemented y outputs are available for feedback to the flip-flop inputs.

  - If the set and reset function of the flip-flop includes true and complemented variables, it is possible that both Set and Reset are a 1 together during a transition, causing both the y and outputs to be 0.

    - This might cause a critical race hazard, though this is unlikely with two-level circuits. The inverse y and output can be generates using a separate gate is necessary.