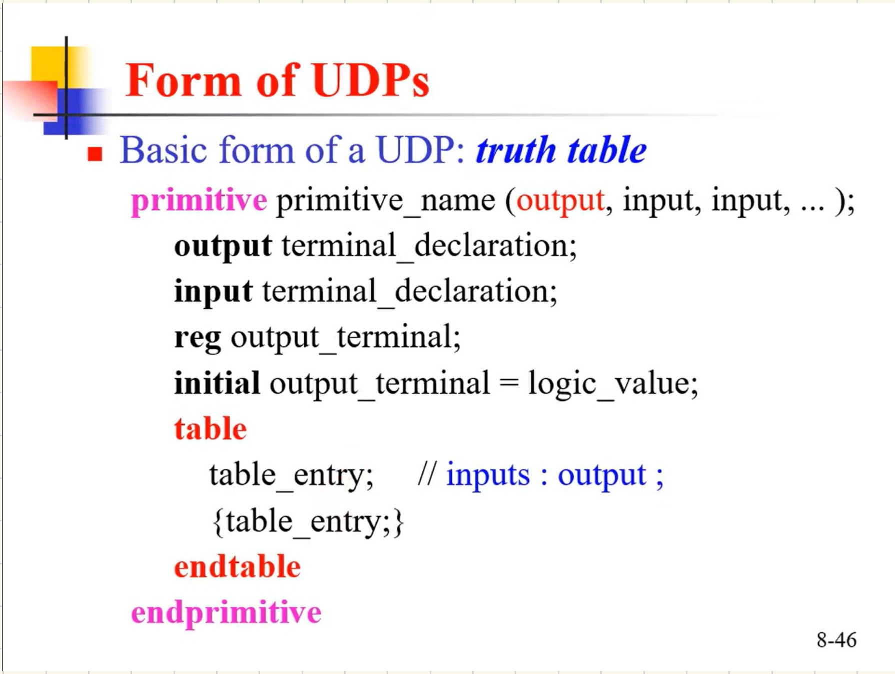
8.5 User Defined Primitives

1 Introduction

- multiple input ports, one output port
- define its functionality in
  - truth table : combinational
  - state table : sequential
- can be instantinted as built-in Primitives.
- ports :
  - bidirectional inout ports are not allowed in UDPs.
  - No vector ports are allowed.
  - tristate / high-impedance state (Z) is not allowed. (0, 1, x are allowed)
    - Z inputs are interpreted as X
  - In seq UDPs, the output always has the same value as the internal state.

2 Combinational UDPs



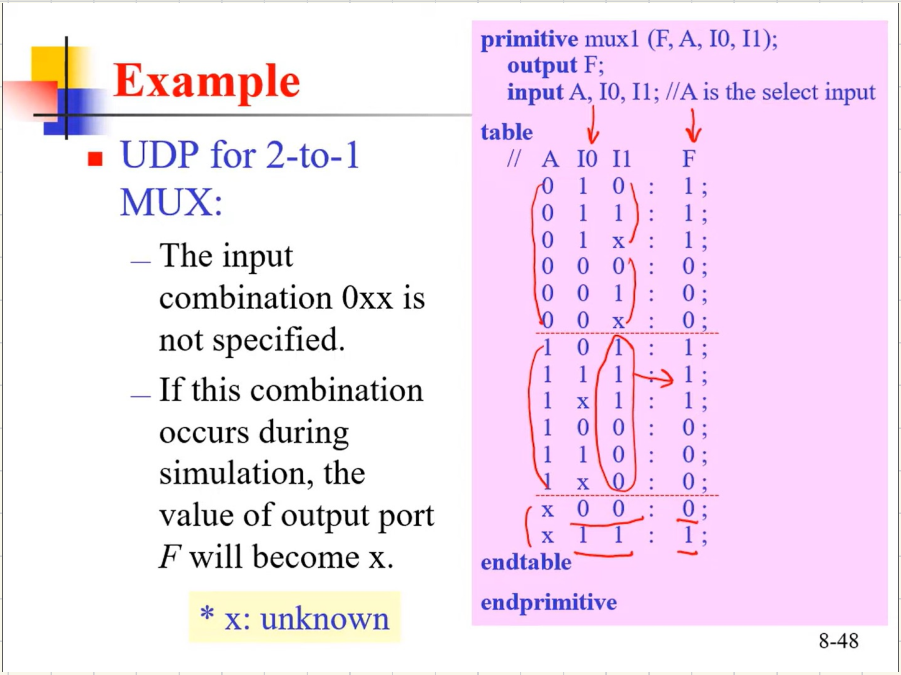
Form of UDPs

- Basic form of a UDP: ***truth table***

```
primitive primitive_name (output, input, input, ... );
    output terminal_declaration;
    input terminal_declaration;
    reg output_terminal;
    initial output_terminal = logic_value;
    table
       table_entry;     // inputs : output ;
       {table_entry;}
    endtable
endprimitive
```

8-46

Ex. UDP for 2x1 MUX



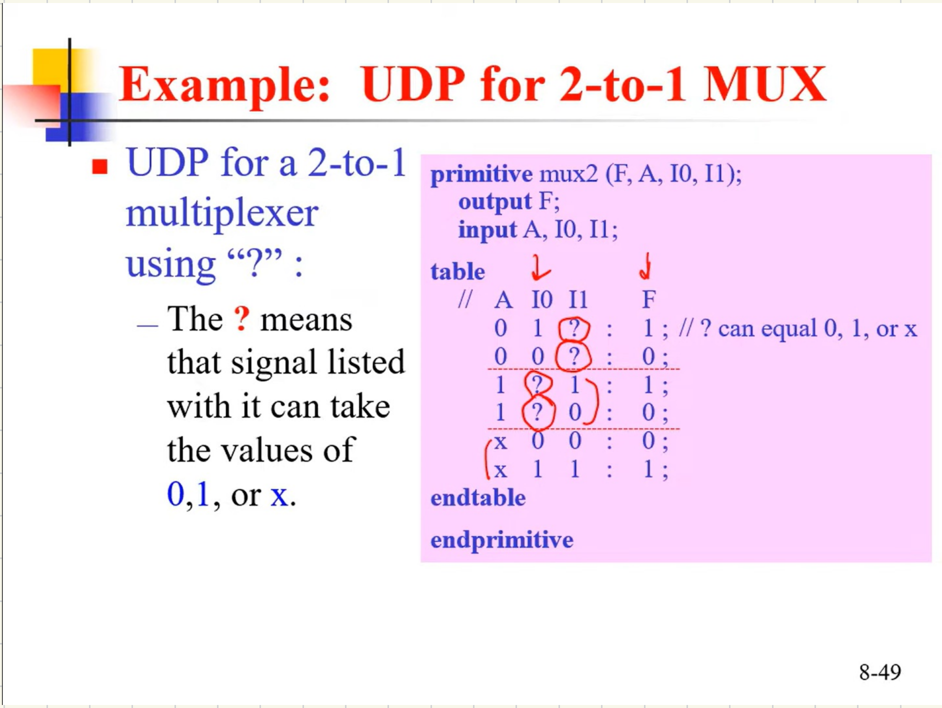
Example

- UDP for 2-to-1 MUX:
  - The input combination 0xx is not specified.
  - If this combination occurs during simulation, the value of output port *F* will become x.

* x: unknown

```
primitive mux1 (F, A, I0, I1);
   output F;
   input A, I0, I1; //A is the select input
table
   //  A  I0  I1      F
       0   1   0  :   1 ;
       0   1   1  :   1 ;
       0   1   x  :   1 ;
       0   0   0  :   0 ;
       0   0   1  :   0 ;
       0   0   x  :   0 ;
       1   0   1  :   1 ;
       1   1   1  :   1 ;
       1   x   1  :   1 ;
       1   0   0  :   0 ;
       1   1   0  :   0 ;
       1   x   0  :   0 ;
       x   0   0  :   0 ;
       x   1   1  :   1 ;
endtable
endprimitive
```

8-48



Example: UDP for 2-to-1 MUX

- UDP for a 2-to-1 multiplexer using "?" :
  - The ? means that signal listed with it can take the values of 0,1, or x.

```
primitive mux2 (F, A, I0, I1);
   output F;
   input A, I0, I1;
table
   //  A  I0  I1     F
       0  1   ?   :  1 ; // ? can equal 0, 1, or x
       0  0   ?   :  0 ;
       1  ?   1   :  1 ;
       1  ?   0   :  0 ;
       x  0   0   :  0 ;
       x  1   1   :  1 ;
endtable

endprimitive
```

3 Sequential UDPs

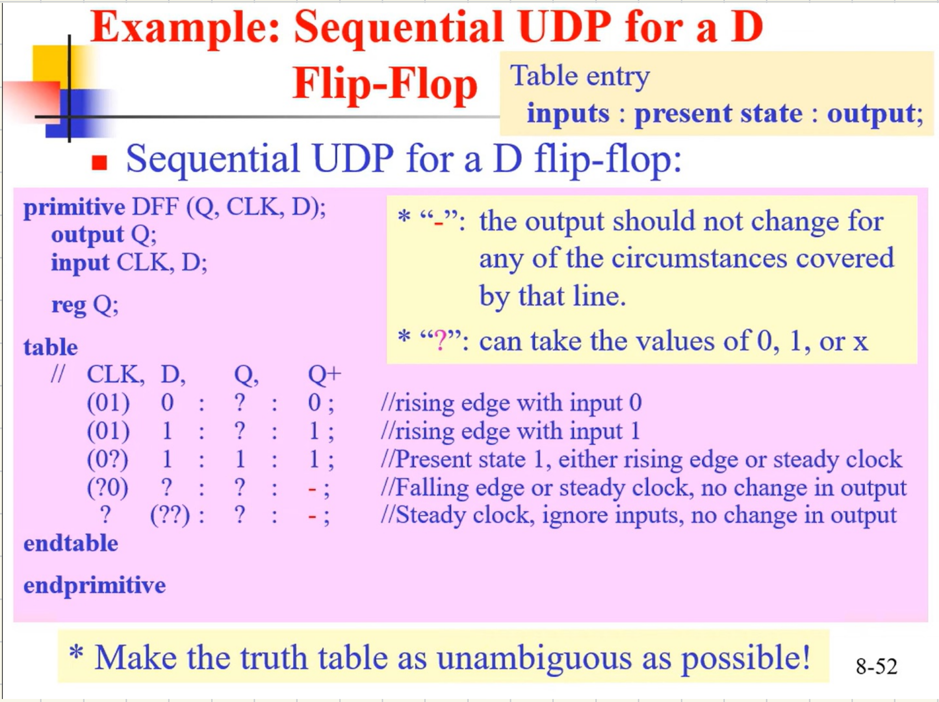
- Outputs must have the same state as the internal state.
- The output must be defined as reg.
- Can model both edge-sensitive and level-sensitive behavior.

- edge-sensitive behavior can be represented in tabular form by listing the value before and after the edge.
  - 01 : rising edge
  - 10 : falling edge

- each table entry

  inputs : present state : outputs

- If level-sensitive behavior such as async set and reset are in a table along w/ edge-sensitive behavior for data, the level-sensitive behavior should be listed before the edge-sensitive behavior.



Example: Sequential UDP for a D Flip-Flop

Table entry
inputs : **present state** : **output;**

- Sequential UDP for a D flip-flop:

```
primitive DFF (Q, CLK, D);
   output Q;
   input CLK, D;

   reg Q;

table
   //   CLK,  D,      Q,      Q+
        (01)   0   :   ?   :   0 ;     //rising edge with input 0
        (01)   1   :   ?   :   1 ;     //rising edge with input 1
        (0?)   1   :   1   :   1 ;     //Present state 1, either rising edge or steady clock
        (?0)   ?   :   ?   :   - ;     //Falling edge or steady clock, no change in output
         ?    (??) :   ?   :   - ;     //Steady clock, ignore inputs, no change in output
endtable

endprimitive
```

* "-": the output should not change for any of the circumstances covered by that line.

* "?": can take the values of 0, 1, or x

* Make the truth table as unambiguous as possible!