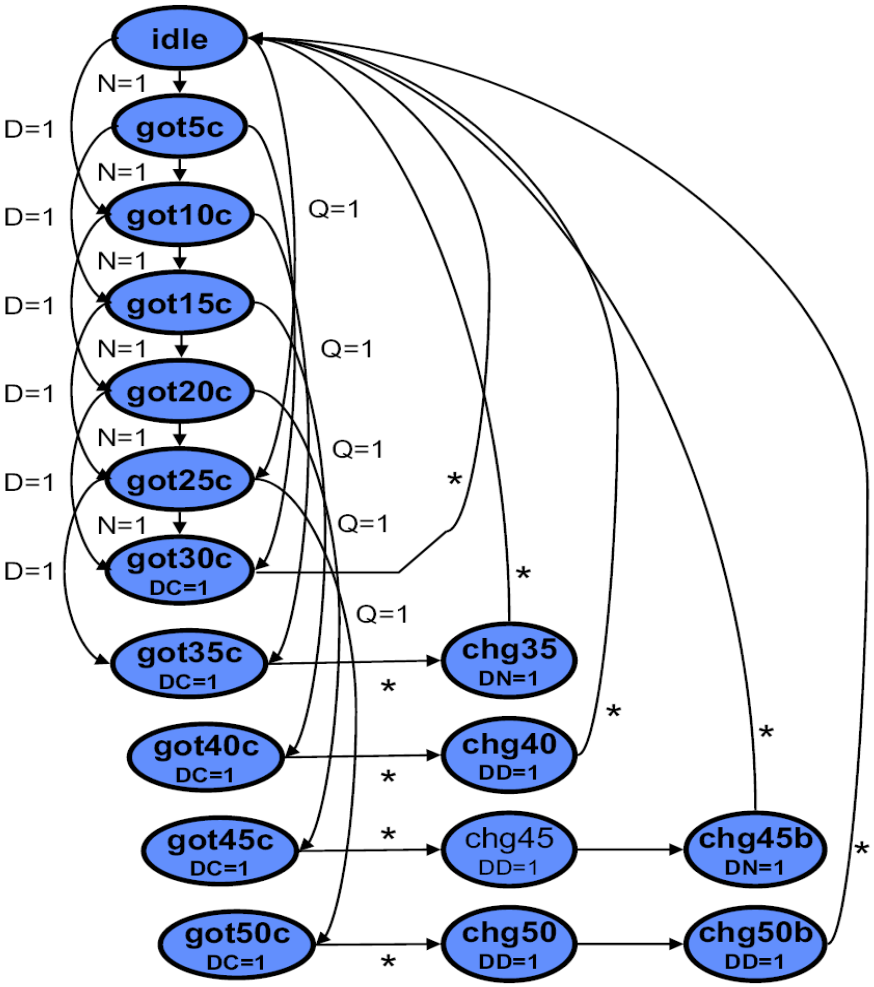
فرض کنید ماشین دستگاه فروش نوشابه در اختیار داریم که سکه های 5 و 10 و 25 سنتی قبول می کند و قیمت فروش هر نوشابه 30 سنت است. دیاگرام مور و کد وریلاگ ماشین فوق را بنویسید.



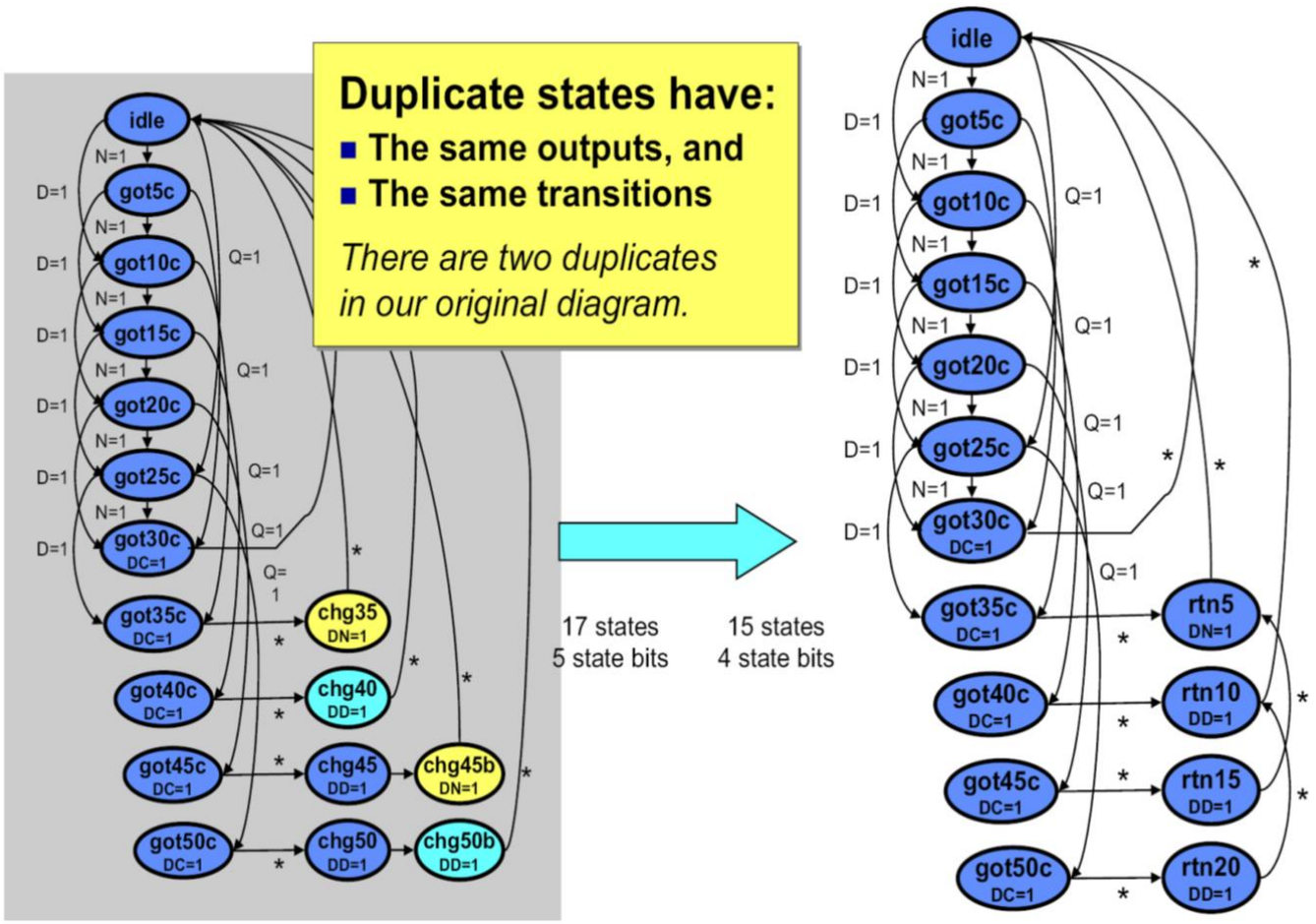
N => 5c

D => 10c

Q => 25c

DN => 5c

DC => 


Duplicate states have:
The same outputs, and
The same transitions
There are two duplicates in our original diagram.
idle
got5c
got10c
got15c
got20c
got25c
got30c
DC=1
got35c
DC=1
got40c
DC=1
got45c
DC=1
got50c
DC=1
chg35
DN=1
chg40
DD=1
chg45
DD=1
chg45b
DN=1
chg50
DD=1
chg50b
DD=1
N=1
D=1
Q=1
*
17 states
5 state bits
15 states
4 state bits
rtn5
DN=1
rtn10
DD=1
rtn15
DD=1
rtn20
DD=1

```
module mooreVender (N, D, Q, DC, DN, DD,
                    clk, reset, state);
  input N, D, Q, clk, reset;
  output DC, DN, DD;
  output [3:0] state;
  reg [3:0] state, next;
```

States defined with parameter keyword

```
  parameter IDLE = 0;
  parameter GOT_5c = 1;
  parameter GOT_10c = 2;
  parameter GOT_15c = 3;
  parameter GOT_20c = 4;
  parameter GOT_25c = 5;
  parameter GOT_30c = 6;
  parameter GOT_35c = 7;
  parameter GOT_40c = 8;
  parameter GOT_45c = 9;
  parameter GOT_50c = 10;
  parameter RETURN_20c = 11;
  parameter RETURN_15c = 12;
  parameter RETURN_10c = 13;
  parameter RETURN_5c = 14;
```

State register defined with sequential always block

```
  always @ (posedge clk or negedge reset)
    if (!reset)    state <= IDLE;
    else           state <= next;
```

Next-state logic within a combinational always block

```
always @ (state or N or D or Q) begin

   case (state)
     IDLE:      if (Q) next = GOT_25c;
           else if (D) next = GOT_10c;
              else if (N) next = GOT_5c;
              else next = IDLE;

     GOT_5c:    if (Q) next = GOT_30c;
           else if (D) next = GOT_15c;
              else if (N) next = GOT_10c;
              else next = GOT_5c;

     GOT_10c:   if (Q) next = GOT_35c;
           else if (D) next = GOT_20c;
              else if (N) next = GOT_15c;
              else next = GOT_10c;

     GOT_15c:   if (Q) next = GOT_40c;
           else if (D) next = GOT_25c;
              else if (N) next = GOT_20c;
              else next = GOT_15c;

     GOT_20c:   if (Q) next = GOT_45c;
           else if (D) next = GOT_30c;
              else if (N) next = GOT_25c;
              else next = GOT_20c;

     GOT_25c:   if (Q) next = GOT_50c;
           else if (D) next = GOT_35c;
              else if (N) next = GOT_30c;
              else next = GOT_25c;

          GOT_30c:  next = IDLE;
          GOT_35c:  next = RETURN_5c;
          GOT_40c:  next = RETURN_10c;
          GOT_45c:  next = RETURN_15c;
          GOT_50c:  next = RETURN_20c;

          RETURN_20c:  next = RETURN_10c;
          RETURN_15c:  next = RETURN_5c;
          RETURN_10c:  next = IDLE;
          RETURN_5c:   next = IDLE;

          default: next = IDLE;
      endcase
   end
```

Combinational output assignment

```
assign DC = (state == GOT_30c || state == GOT_35c ||
             state == GOT_40c || state == GOT_45c ||
             state == GOT_50c);
assign DN = (state == RETURN_5c);
assign DD = (state == RETURN_20c || state == RETURN_15c ||
             state == RETURN_10c);
endmodule
```

Next-state and output logic combined into a single `always` block

```
always @ (state or N or D or Q) begin

  DC = 0; DD = 0; DN = 0;     // defaults

   case (state)
     IDLE:      if (Q) next = GOT_25c;
           else if (D) next = GOT_10c;
           else if (N) next = GOT_5c;
           else next = IDLE;

     GOT_5c:    if (Q) next = GOT_30c;
           else if (D) next = GOT_15c;
           else if (N) next = GOT_10c;
           else next = GOT_5c;

     GOT_10c:   if (Q) next = GOT_35c;
           else if (D) next = GOT_20c;
           else if (N) next = GOT_15c;
           else next = GOT_10c;

     GOT_15c:   if (Q) next = GOT_40c;
           else if (D) next = GOT_25c;
           else if (N) next = GOT_20c;
           else next = GOT_15c;

     GOT_20c:   if (Q) next = GOT_45c;
           else if (D) next = GOT_30c;
           else if (N) next = GOT_25c;
           else next = GOT_20c;

     GOT_25c:   if (Q) next = GOT_50c;
           else if (D) next = GOT_35c;
           else if (N) next = GOT_30c;
           else next = GOT_25c;

          GOT_30c:  begin
                      DC = 1; next = IDLE;
                    end
          GOT_35c:  begin
                      DC = 1; next = RETURN_5c;
                    end
          GOT_40c:  begin
                      DC = 1; next = RETURN_10c;
                    end
          GOT_45c:  begin
                      DC = 1; next = RETURN_15c;
                    end
          GOT_50c:  begin
                      DC = 1; next = RETURN_20c;
                    end

          RETURN_20c: begin
                        DD = 1; next = RETURN_10c;
                      end
          RETURN_15c: begin
                        DD = 1; next = RETURN_5c;
                      end
          RETURN_10c: begin
                        DD = 1; next = IDLE;
                      end
          RETURN_5c:  begin
                        DN = 1; next = IDLE;
                      end

          default: next = IDLE;
      endcase
   end
```

```
reg DC,DN,DD;

 // Sequential always block for state assignment
 always @ (posedge clk or negedge reset) begin
   if (!reset)   state <= IDLE;
   else if (clk) state <= next;

   DC <= (next == GOT_30c || next == GOT_35c ||
          next == GOT_40c || next == GOT_45c ||
          next == GOT_50c);
   DN <= (next == RETURN_5c);
   DD <= (next == RETURN_20c || next == RETURN_15c ||
          next == RETURN_10c);
 end
```