One-Hot State Machines

- Typical state machine designs minimize the number of flip flops
- This is a result of using highly-encoded states
- To decode the states, lots of combo logic is required.
- FPGAs are programmable logic devices that are rich in flip-flops and poor in combo logic
- One-Hot state machines use one flip-flop per state and thus need much less decode logic
- This makes one-hot encoding much more efficient for FPGAs.

- One Hot Encoding: 000001, 000010, 000100, 001000, 010000,....

One-Hot State Machines

- case usually evaluates a case_expression and then checks the result against several case_items in a list

```
case (case_expression)
  case_item1  :  case_item_statement1;
  case_item2  :  case_item_statement2;
  case_item3  :  case_item_statement3;
  case_item4  :  case_item_statement4;
  default     :  case_item_statement5;
endcase
```

One-Hot State Machines

- One hot encoding uses the *reversed case statement*
- In this style, `case_expression` and `case_item` are swapped
- In one-hot encoding:
  - `case_expression` is the literal (1'b1) to match against
  - `case_items` are single bits in the present state vector

```
case (1'b1)
  present_state[bit0]:  next_state_assignment;
  present_state[bit1]:  next_state_assignment;
  present_state[bit2]:  next_state_assignment;
  present_state[bit3]:  next_state_assignment;
endcase
```

- The index to bits in the _ps vector are declared as parameters or enumerated types
- This style infers efficient one-bit comparison logic against the _ps and _ns vectors

One-Hot State Machines

- One-hot state machine code
  - Infer the present state flip-flops

```
//instantiate the _ps vector flip flops
  always_ff @ (posedge clk, negedge rst_n)
    if (!rst_n) begin
      arbiter_oh_ps       <= 4'b0000; //reset, clear all bits
      arbiter_oh_ps[IDLE] <= 1'b1;    //set IDLE state bit
    end
    else arbiter_oh_ps    <= arbiter_oh_ns;
```

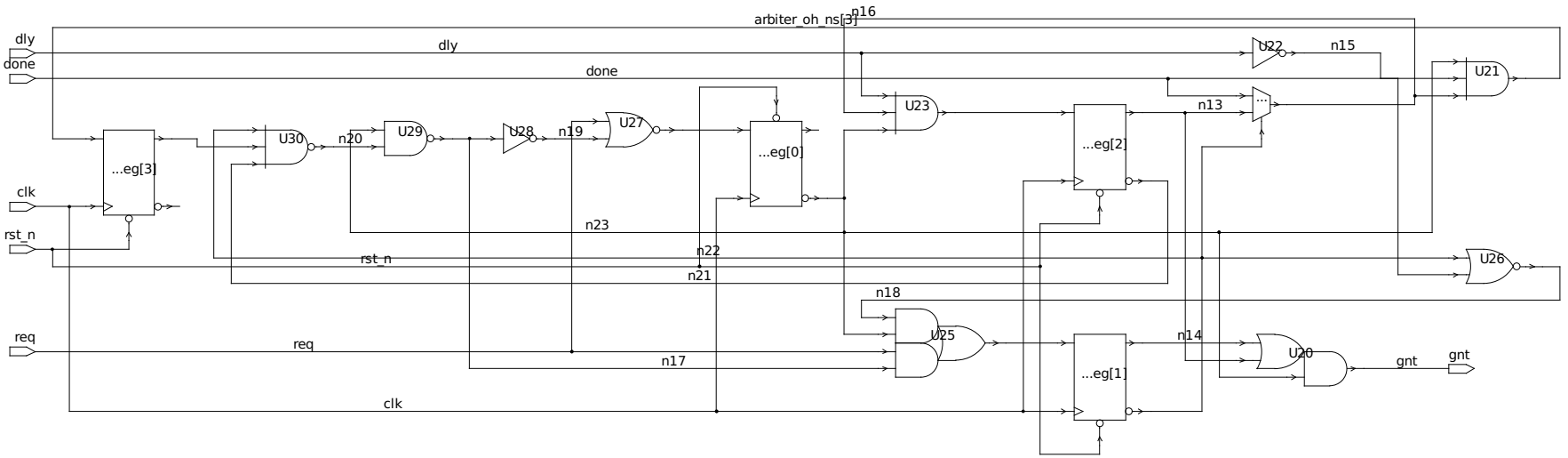
One-Hot State Machines

- Glitchless state machine code
  - Define _ns transitions and output

```
//describe the _ns steering logic and output decoder
  always_comb begin
  arbiter_oh_ns = 4'b0000;  //default for _ns vector
  gnt = 1'b0;               //default assignment
  case (1'b1)
    arbiter_oh_ps[IDLE] :
      if (req)       arbiter_oh_ns[BBUSY] = 1'b1;
      else           arbiter_oh_ns[IDLE]  = 1'b1;
    arbiter_oh_ps[BBUSY]:
    begin
      gnt = 1'b1;    //output is asserted here
      if (!done)     arbiter_oh_ns[BBUSY] = 1'b1;
      else if ( dly) arbiter_oh_ns[BWAIT] = 1'b1;
      else           arbiter_oh_ns[BFREE] = 1'b1;
    end
    arbiter_oh_ps[BWAIT]:
    begin
      gnt = 1'b1;
      if (!dly)      arbiter_oh_ns[BFREE] = 1'b1;
      else           arbiter_oh_ns[BWAIT] = 1'b1;
    end
    arbiter_oh_ps[BFREE]:
      if (req)       arbiter_oh_ns[BBUSY] = 1'b1;
      else           arbiter_oh_ns[IDLE]  = 1'b1;
    endcase
  end //always
```

One-Hot State Machines

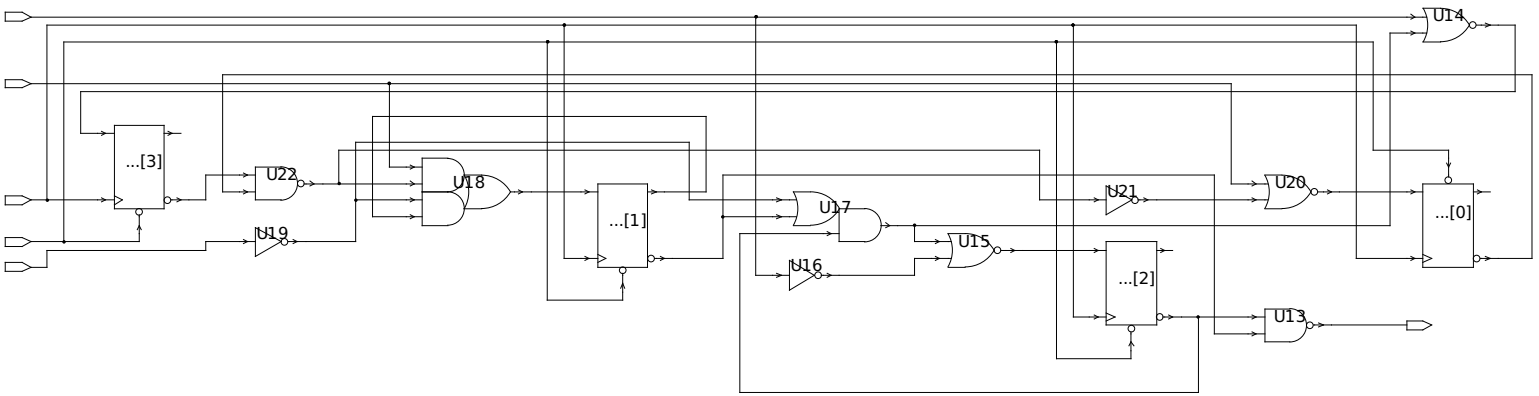
- Synthesis output from one-hot state machine
  - Could the output `gnt` glitch?



One-Hot State Machines

- The case items are unique, they may be decoded in parallel
- Good place to use `unique case`

```
always_comb begin
arbiter_oh_ns = 4'b0000;  //default for _ns vector
gnt = 1'b0;               //default assignment
unique case (1'b1)
  ...
```



- Difference in area: 218 without `unique`, 196 with `unique`