

Universal Counter

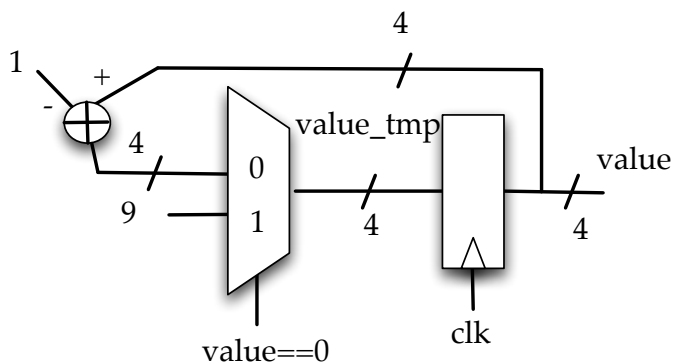
Hsi-Pin Ma

<http://lms.nthu.edu.tw/course/21094>

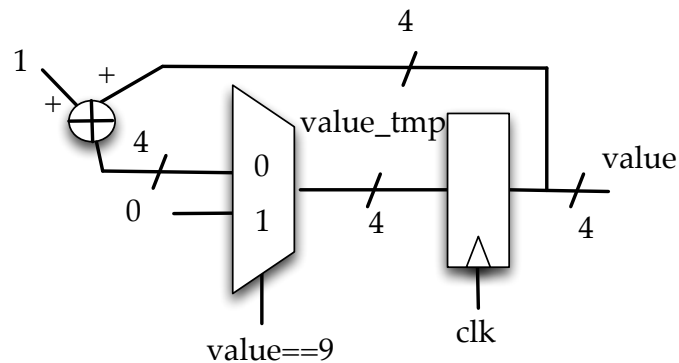
Department of Electrical Engineering

National Tsing Hua University

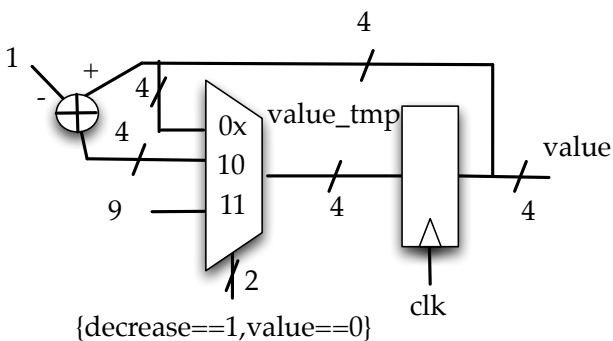
Counter Modules



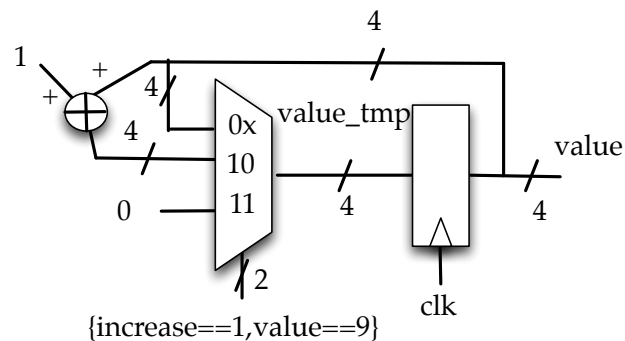
BCD down counter



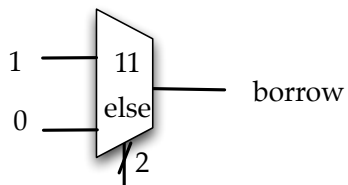
BCD up counter



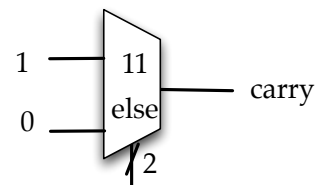
{decrease==1,value==0}



{increase==1,value==9}

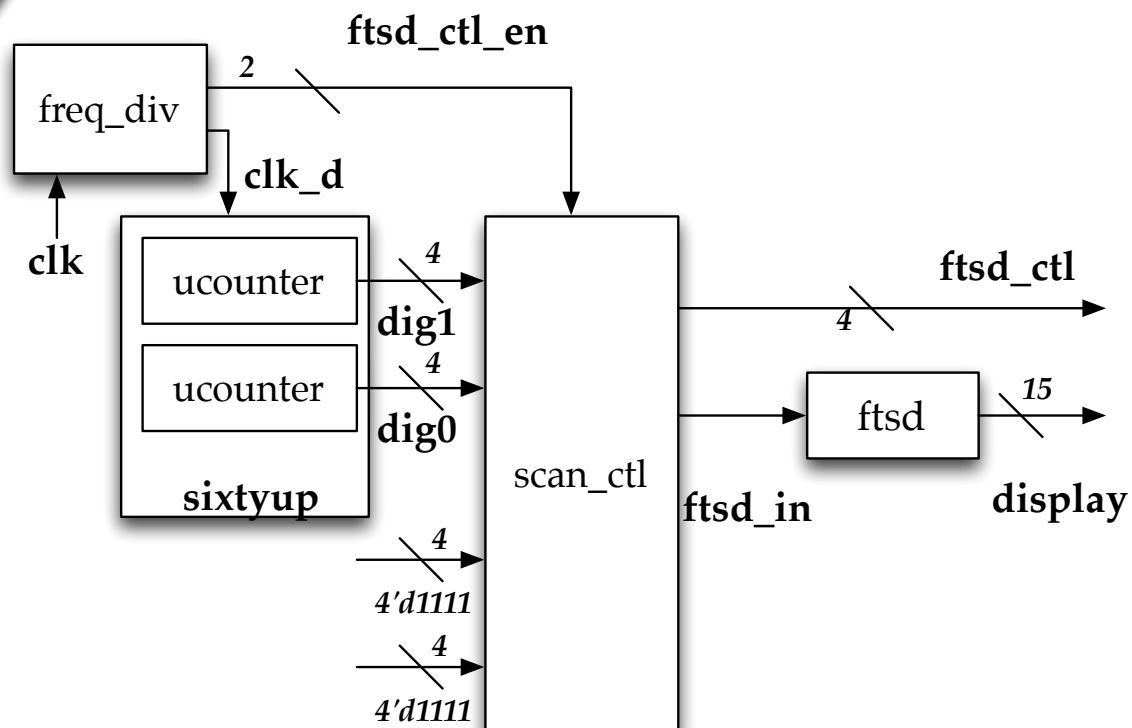
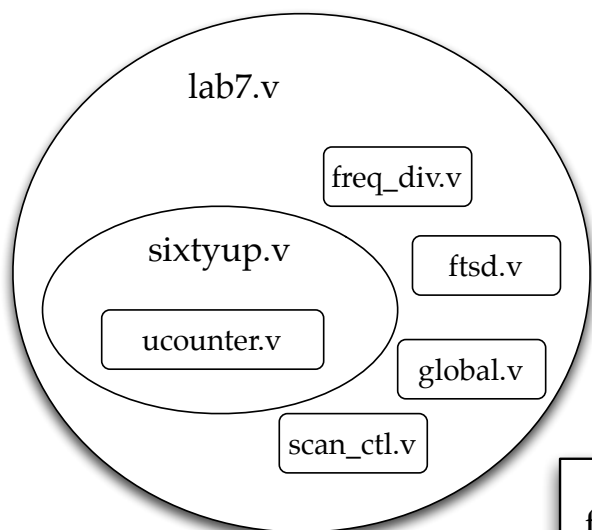


{decrease==1,value==0}

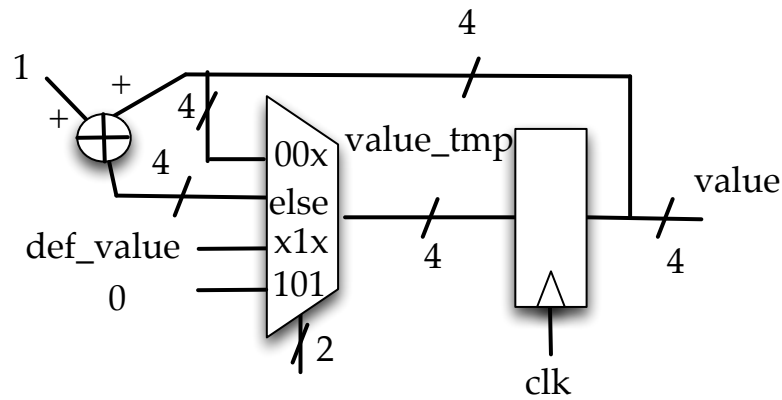
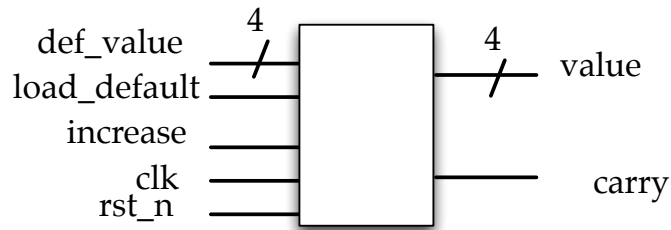


{increase==1,value==9}

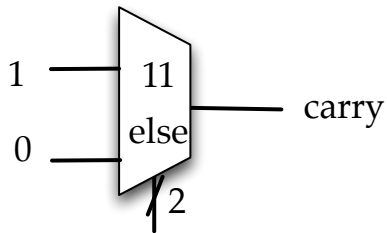
Lab7 example



ucounter.v (1/2)



{increase==0,load_default==1,value==9}



{increase==1,value==9}

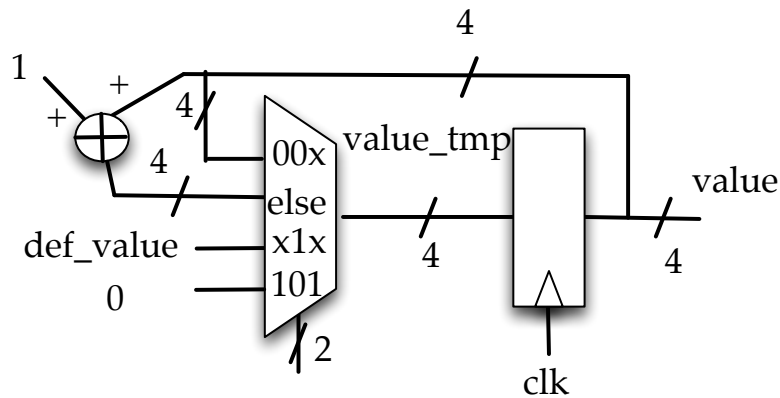
```
`include "global.v"
module upcounter(
    value, // counter value
    carry, // carryout to enable counting in next stage
    clk, // global clock
    rst_n, // active low reset
    increase, // enable control for counter
    load_default, // enable load default value
    def_value // default value for counter
);

// outputs
output [BCD_BIT_WIDTH-1:0] value; // counter value
output carry; // carryout to enable counting in next stage

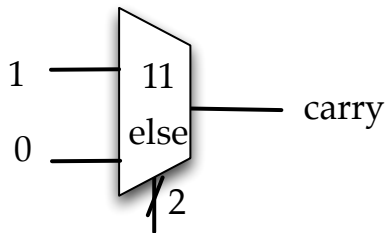
// inputs
input clk; // global clock
input rst_n; // active low reset
input load_def; // enable load default value
input increase; // enable control for counter
input [BCD_BIT_WIDTH-1:0] def_value; // counter upper limit

reg [BCD_BIT_WIDTH-1:0] value; // output (in always block)
reg [BCD_BIT_WIDTH-1:0] value_tmp; // input to dff (in always block)
reg carry; // carryout to enable counting in next stage
```

ucounter.v (2/2)



{increase==0,load_default==1,value==9}



{increase==1,value==9}

// combinational part for BCD counter

```
always @*
  if (increase==`DISABLED)
    value_tmp = value;
  else if (load_default==`ENABLED)
    value_tmp = def_value;
  else if ((increase==`ENABLED)&&(value==`BCD_NINE))
    value_tmp = `BCD_ZERO;
  else
    value_tmp = value + `INCREMENT;
```

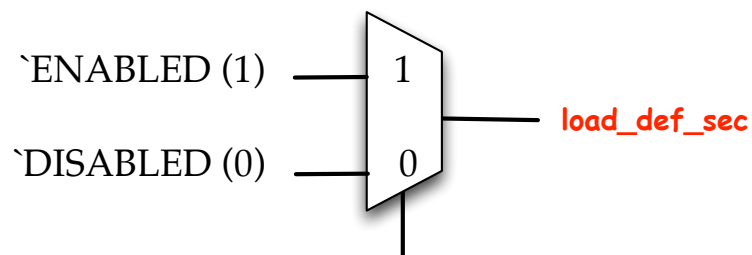
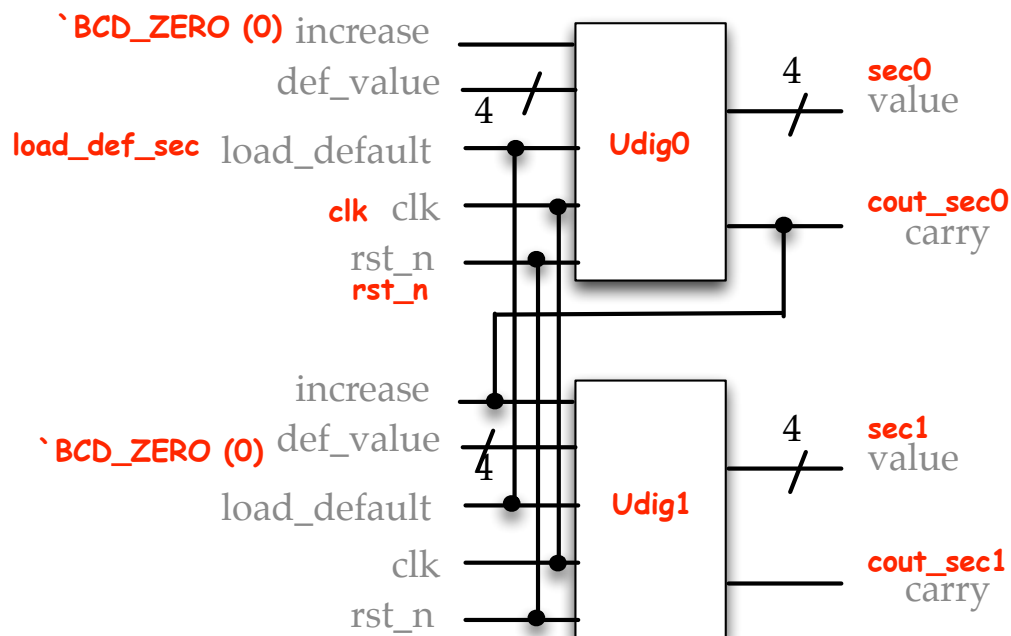
```
always @*
  if ((increase==`ENABLED)&&(value == `BCD_NINE))
    carry = `ENABLED;
  else
    carry = `DISABLED;
```

// register part for BCD counter

```
always @(posedge clk or negedge rst_n)
  if (~rst_n) value <= def_value;
  else value <= value_tmp;
```

endmodule

sixtyup.v (1/2)



`sec1==`BCD_FIVE && sec0==`BCD_NINE`

sixtyup.v (2/2)

1

```

`include "global.v"
module thirtyup(
    sec1, // digit 1 for second
    sec0, // digit 0 for second
    clk, // global clock
    rst_n // low active reset
);

// outputs
output [BCD_BIT_WIDTH-1:0] sec1; // digit 1 for second
output [BCD_BIT_WIDTH-1:0] sec0; // digit 0 for second
// inputs
input clk; // global clock signal
input rst_n; // low active reset

// temporary nets
reg load_def_sec; // enabled to load second value
wire cout_sec0, cout_sec1; // BCD counter carryout

// return from 59 to 00
always @(sec0 or sec1)
    if ((sec1==`BCD_FIVE)&&(sec0==`BCD_NINE))
        load_def_sec = `ENABLED;
    else
        load_def_sec = `DISABLED;

```

2

```

// counter for digit 0
ucounter Udig0(
    .value(sec0), // digit 0 of second
    .carry(cout_sec0), // carry out for digit 0
    .clk(clk), // clock
    .rst_n(rst_n), // asynchronous low active reset
    .increase(`ENABLED), // always increasing
    .load_default(load_def_sec), // enable load default value
    .def_value(`BCD_ZERO) // default value for counter
);

// counter for digit 1
ucounter Udig1(
    .value(sec1), // digit 1 of second
    .carry(cout_sec1), // carry out for digit 1
    .clk(clk), // clock
    .rst_n(rst_n), // asynchronous low active reset
    .increase(cout_sec0), // increasing when digit 0 carry out
    .load_default(load_def_sec), // enable load default value
    .def_value(`BCD_ZERO) // default value for counter
);

endmodule

```