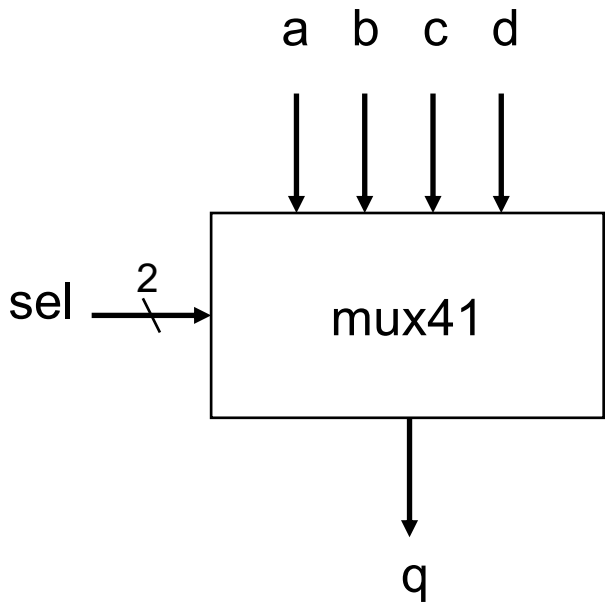
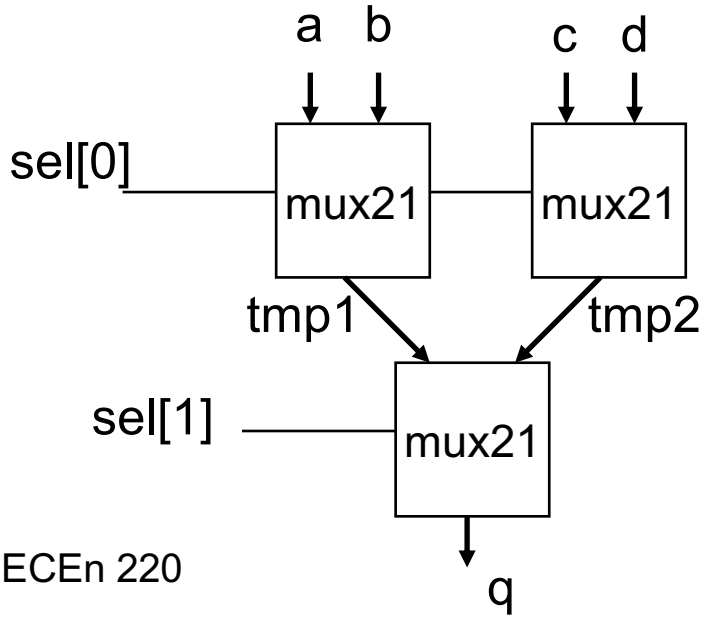
Hierarchical SystemVerilog Design

```
module mux41(
       output logic q,
       input wire logic[1:0] sel,
       input wire logic a, b, c, d);

       logic tmp1, tmp2;

       mux21 M0(tmp1, sel[0], a, b);
       mux21 M1(tmp2, sel[0], c, d);
       mux21 M2(q, sel[1], tmp1, tmp2);
endmodule
```





Hierarchical SystemVerilog Design

```
module mux41(
        output logic q,
        input wire logic[1:0] sel,
        input wire logic a, b, c, d);

        logic tmp1, tmp2;

        mux21 M0(tmp1, sel[0], a, b);
        mux21 M1(tmp2, sel[0], c, d);
        mux21 M2(q, sel[1], tmp1, tmp2);
endmodule
```

Multiple-bit wires can be declared and used. sel is a 2-bit wire.

Individual bits of a multi-bit wire can be accessed using C-like array subscript notation.

Most significant bit of sel

Least significant bit of sel

Hierarchical SystemVerilog Design

```
module mux41(
       output logic q,
       input wire logic[1:0] sel,
       input wire logic a, b, c, d);

       logic tmp1, tmp2;

       mux21 M0(tmp1, sel[0], a, b);
       mux21 M1(tmp2, sel[0], c, d);
       mux21 M2(q, sel[1], tmp1, tmp2);
endmodule
```

Previously defined instance mux21 can be used within this module.

module name

instance name (each instance name must be unique)

General instance format:

```
moduleName  instanceName(port1, port2, … portn);
```

More on Instances

- Each instance must have an instance name
  - Except for built-in gates
- Alternate syntax
  - Specify port name and wire
  - So you don't have to remember order

```
mux21 M1(.sel(sel[0]), .b(d), .a(c), .q(tmp2));
```

Constants

- Often need to specify constant value

```
fullAdd FA0(a[0], b[0], 1'b0, s[0], c[0]);
```

Wire a 1-bit binary '0' value to the carry-in of this full adder

Constants

`1'b0`	A 1-bit number whose value is binary 0
`3'b110`	A 3-bit number whose value is binary 110
`4'b00`	An error
`8'd13`	An 8-bit number whose value is decimal 13
`24'h00FFD2`	A 24-bit number whose value is hex '00FFD2'

General form: `#bits ' base Value`

(But without the spaces…)

More on Constants

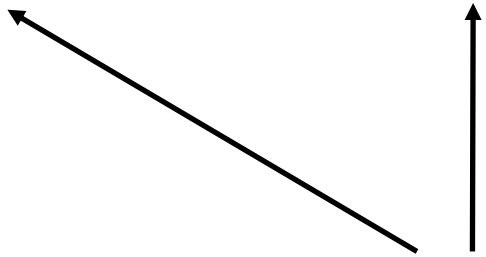
`4'b1100`	A 4-bit number whose value is binary 1100
`4'd12`	A 4-bit number whose decimal value is 12
`4'hC`	A 4-bit hexadecimal number whose value is C
`12`	A decimal number 12 (default decimal)
`3'd12`	Error! Needs 4 bits for decimal 12
`6'h`	Error! Needs 4 bits per hexadecimal digit

General form: `#bits ' base Value`

(But without the spaces…)

More on Multi-Bit Wires

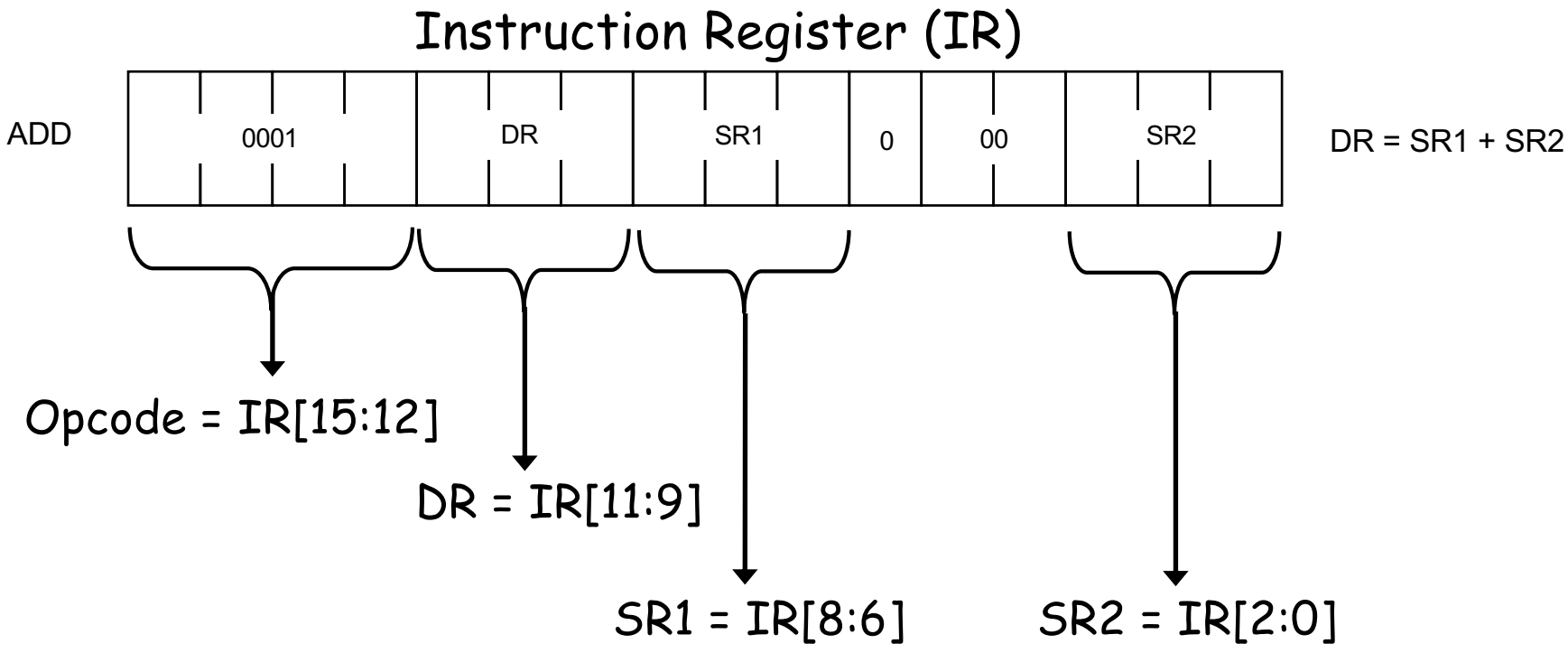
```
input wire logic[7:0] x;  // An 8-bit input wire
... x[0] ...    // The LSB of x
... x[7] ...    // The MSB of x
... x[2:0]...   /* 3 LSB's of x */
```



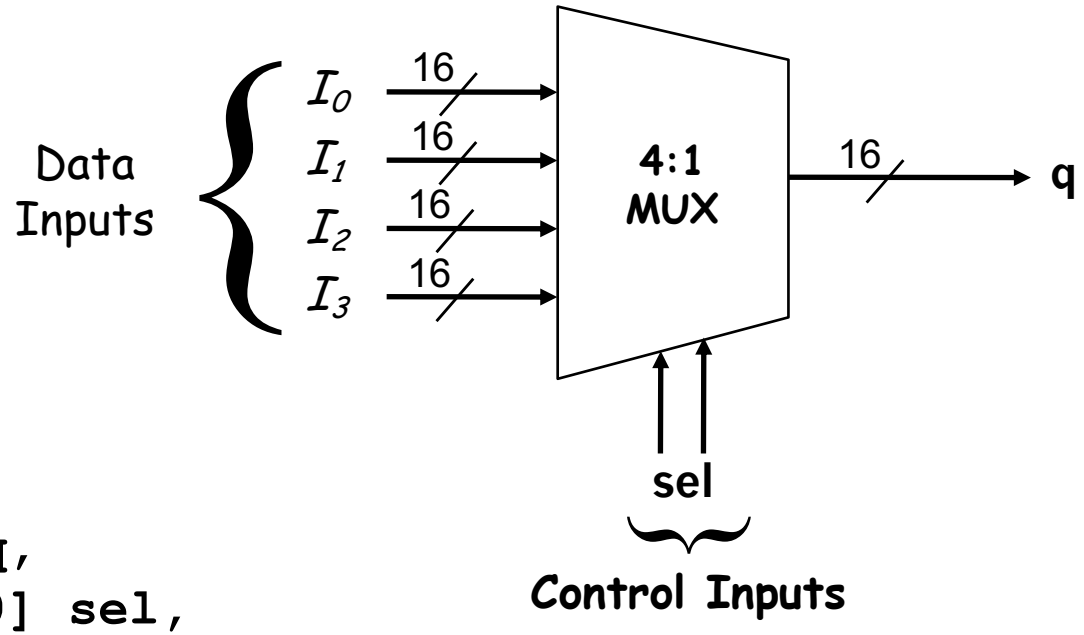
Notice that C-style comments are allowed…

More on Multi-Bit Wires

Take for example an instruction register from a CPU



Multi-bit Multiplexers




```
module mux41(
       output logic[15:0] q,
       input wire logic[1:0] sel,
       input wire logic[15:0] I0, I1, I2, I3);

       assign q = (sel == 2'b00) ? I0:
                  (sel == 2'b01) ? I1 :
                  (sel == 2'b10) ? I2 :
                  I3;
endmodule
```

More on Naming

- Instance names can clash with signal names:

```
someCell a2(a2, sel, b);    // Bad: the a2's clash
```

- *A Good Convention:*
  - Signal names always start with lower case letter
  - Instance names are always all uppercase
  - Remember, SystemVerilog *is* case-sensitive

```
someCell A2(a2, sel, b);    // Good, no clashes
```