

We are generating a 64-to-1 multiplexer hierarchically in Verilog. We have generated 2-to-1 multiplexer, 4-to-1 multiplexer, 8-to-1 multiplexer, 16-to-1 multiplexer, 32-to-1 multiplexer defined as follows:

```
// 2-to-1 Multiplexer
```

```
module mux2_1(  
    input in1,  
    input in2,  
    input select,  
    output out  
);  
    assign out = (select) ? in2 : in1;  
endmodule
```

```
// 4-to-1 Multiplexer
```

```
module mux4_1(  
    input [3:0] in,  
    input [1:0] select,  
    output out  
);  
    wire mid1, mid2;
```

```
    mux2_1 mux1(in[1], in[0], select[0], mid1);  
    mux2_1 mux2(in[3], in[2], select[0], mid2);  
  
    mux2_1 mux3(mid1, mid2, select[1], out);
```

```
endmodule
```

```
// 8-to-1 Multiplexer
```

```
module mux8_1(  
    input [7:0] in,  
    input [2:0] select,  
    output out  
);  
    wire mid1, mid2;
```

```

mux4_1 mux1(in[3:0], select[1:0], mid1);
mux4_1 mux2(in[7:4], select[1:0], mid2);

mux2_1 mux3(mid1, mid2, select[2], out);

```

endmodule

// 16-to-1 Multiplexer

```
module mux16_1(
```

```
input [15:0] in,
```

```
input [3:0] select,
```

```
output out
```

```
);
```

```
wire mid1, mid2;
```

```

mux8_1 mux1(in[7:0], select[2:0], mid1);
mux8_1 mux2(in[15:8], select[2:0], mid2);

mux2_1 mux3(mid1, mid2, select[3], out);

```

endmodule

// 32-to-1 Multiplexer

```
module mux32_1(
```

```
input [31:0] in,
```

```
input [4:0] select,
```

```
output out
```

```
);
```

```
wire mid1, mid2;
```

```

mux16_1 mux1(in[15:0], select[3:0], mid1);
mux16_1 mux2(in[31:16], select[3:0], mid2);

mux2_1 mux3(mid1, mid2, select[4], out);

```

endmodule

Please include the previous module(s) in your response and use them to hierarchically generate a 64-to-1 multiplexer defined as:

```
module mux64_1(in[63:0], select, out);
```

```
//Insert code here
```

```
endmodule
```

To create a 64-to-1 multiplexer using the existing 32-to-1 multiplexers, you can use two levels of 32-to-1 multiplexers. Here's how you can implement it:

```

// 2-to-1 Multiplexer
module mux2_1(

```

```

        input in1,
        input in2,
        input select,
        output out
    );
    assign out = (select) ? in2 : in1;
endmodule

// 4-to-1 Multiplexer
module mux4_1(
    input [3:0] in,
    input [1:0] select,
    output out
);
    wire mid1, mid2;

    mux2_1 mux1(in[1], in[0], select[0], mid1);
    mux2_1 mux2(in[3], in[2], select[0], mid2);

    mux2_1 mux3(mid1, mid2, select[1], out);
endmodule

// 8-to-1 Multiplexer
module mux8_1(
    input [7:0] in,
    input [2:0] select,
    output out
);
    wire mid1, mid2;

    mux4_1 mux1(in[3:0], select[1:0], mid1);
    mux4_1 mux2(in[7:4], select[1:0], mid2);

    mux2_1 mux3(mid1, mid2, select[2], out);
endmodule

// 16-to-1 Multiplexer
module mux16_1(
    input [15:0] in,
    input [3:0] select,
    output out
);
    wire mid1, mid2;

    mux8_1 mux1(in[7:0], select[2:0], mid1);
    mux8_1 mux2(in[15:8], select[2:0], mid2);

    mux2_1 mux3(mid1, mid2, select[3], out);
endmodule

// 32-to-1 Multiplexer
module mux32_1(
    input [31:0] in,
    input [4:0] select,
    output out
);

```

```

    wire mid1, mid2;

    mux16_1 mux1(in[15:0], select[3:0], mid1);
    mux16_1 mux2(in[31:16], select[3:0], mid2);

    mux2_1 mux3(mid1, mid2, select[4], out);
endmodule

// 64-to-1 Multiplexer
module mux64_1(
    input [63:0] in,
    input [5:0] select,
    output out
);
    wire mid1, mid2;

    mux32_1 mux1(in[31:0], select[4:0], mid1);
    mux32_1 mux2(in[63:32], select[4:0], mid2);

    mux2_1 mux3(mid1, mid2, select[5], out);
endmodule

```

In this implementation, the first level of multiplexers (`mux1` and `mux2`) selects between groups of thirty-two inputs based on the five least significant bits of the `select` signal (`select[4:0]`). The outputs of these multiplexers are then fed into a final multiplexer (`mux3`) that selects between them based on the most significant bit of the `select` signal (`select`).

Note that I've corrected the `select` input in the `mux64_1` module to be a 6-bit vector (`input [5:0] select`) to properly select among the sixty-four inputs.