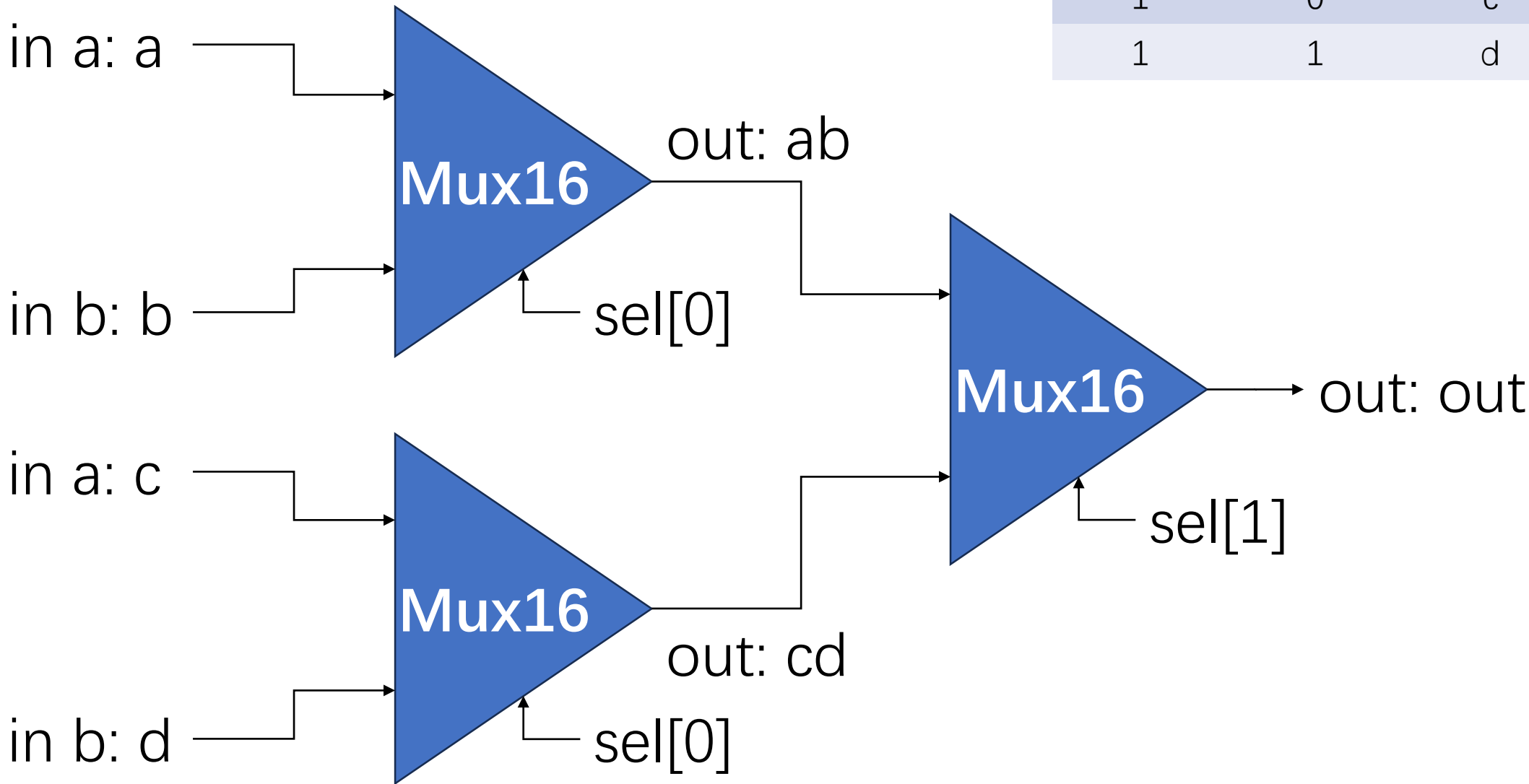
Brief Hints on Lab 02

Mux4Way16

sel[1]	sel[0]	out
0	0	a
0	1	b
1	0	c
1	1	d

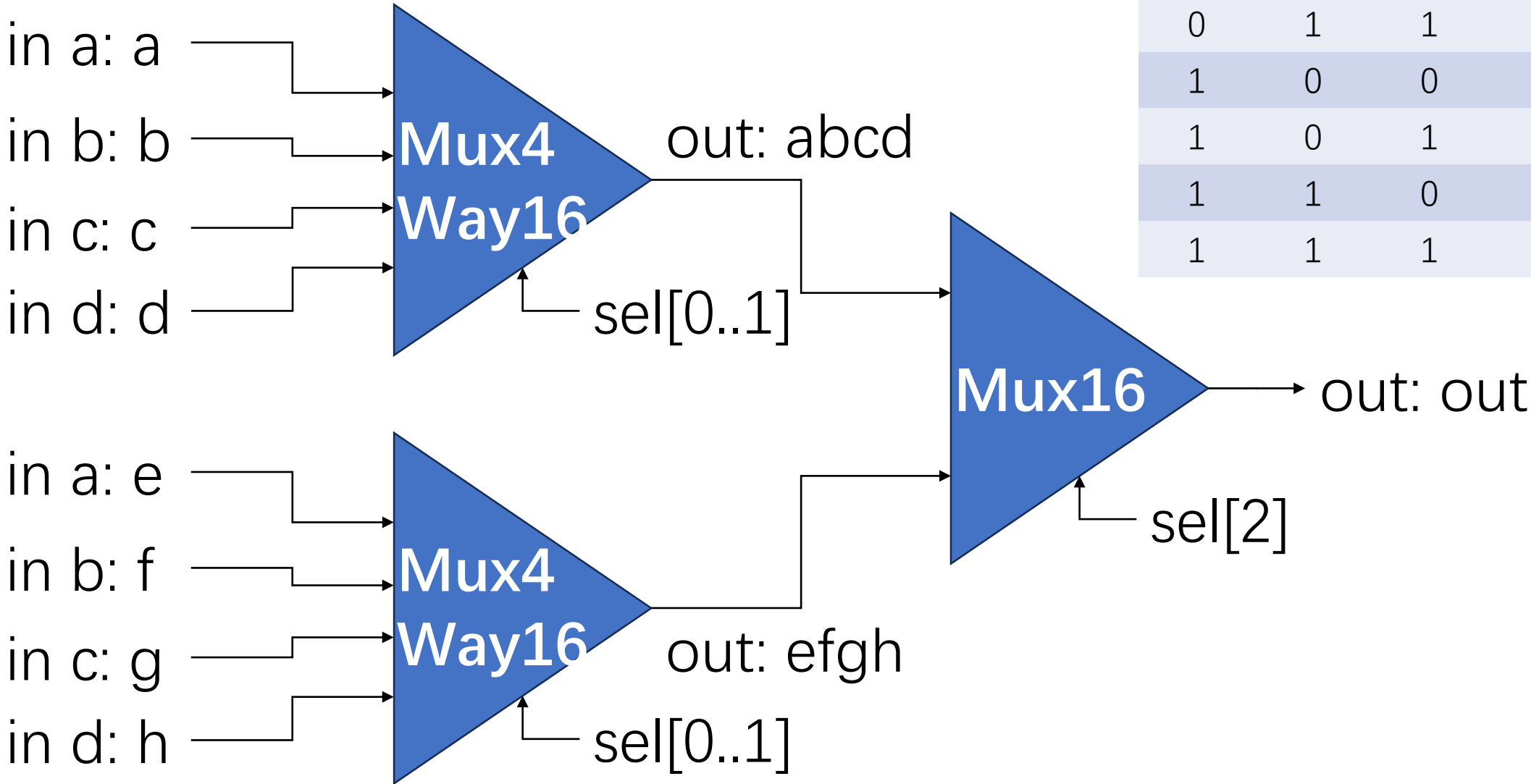


Mux4Way16

```
CHIP Mux4Way16 {
    IN a[16], b[16], c[16], d[16], sel[2];
    OUT out[16];

    PARTS:
    Mux16 (a=a,  b=b,  sel=sel[0], out=ab);
    Mux16 (a=c,  b=d,  sel=sel[0], out=cd);
    Mux16 (a=ab, b=cd, sel=sel[1], out=out);
}
```

Mux8Way16

sel[2]	sel[1]	sel[0]	out
0	0	0	a
0	0	1	b
0	1	0	c
0	1	1	d
1	0	0	e
1	0	1	f
1	1	0	g
1	1	1	h



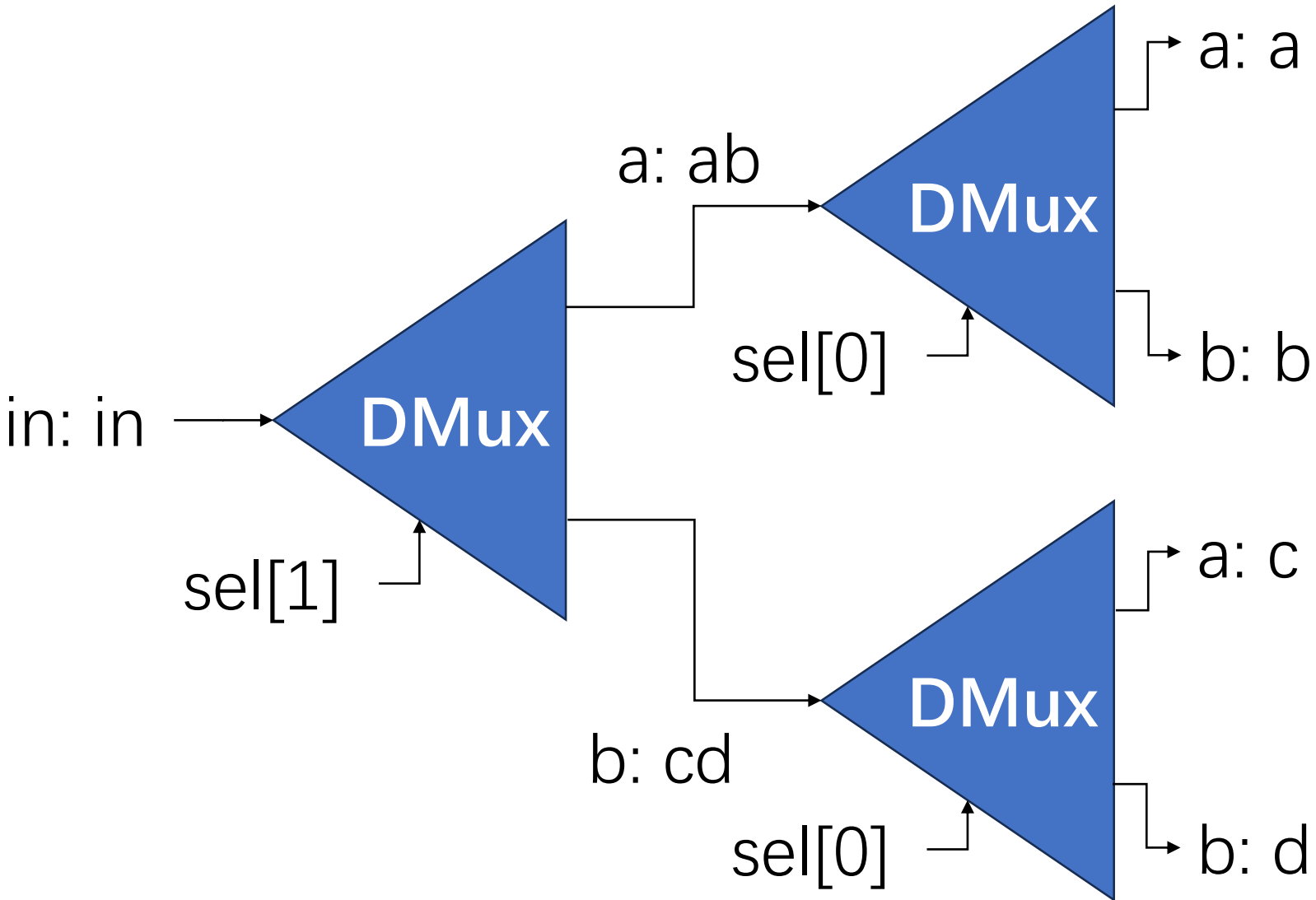
Mux8Way16

```
CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];

    PARTS:
    // Binary tree of 2-way multiplexors
    Mux16 (a=a,    b=b,    sel=sel[0], out=ab);
    Mux16 (a=c,    b=d,    sel=sel[0], out=cd);
    Mux16 (a=e,    b=f,    sel=sel[0], out=ef);
    Mux16 (a=g,    b=h,    sel=sel[0], out=gh);
    Mux16 (a=ab,   b=cd,   sel=sel[1], out=abcd);
    Mux16 (a=ef,   b=gh,   sel=sel[1], out=efgh);
    Mux16 (a=abcd, b=efgh, sel=sel[2], out=out);
}
```

Or you can write by using CHIP Mux4Way16.

```
CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];

    PARTS:
    // Binary tree of 2-way multiplexors
    Mux4Way16(a=a, b=b, c=c, d=d, sel=sel[0..1], out=abcd);
    Mux4Way16(a=e, b=f, c=g, d=h, sel=sel[0..1], out=efgh);
    Mux16(a=abcd, b=efgh, sel=sel[2], out=out);
}
```

DMux4Way

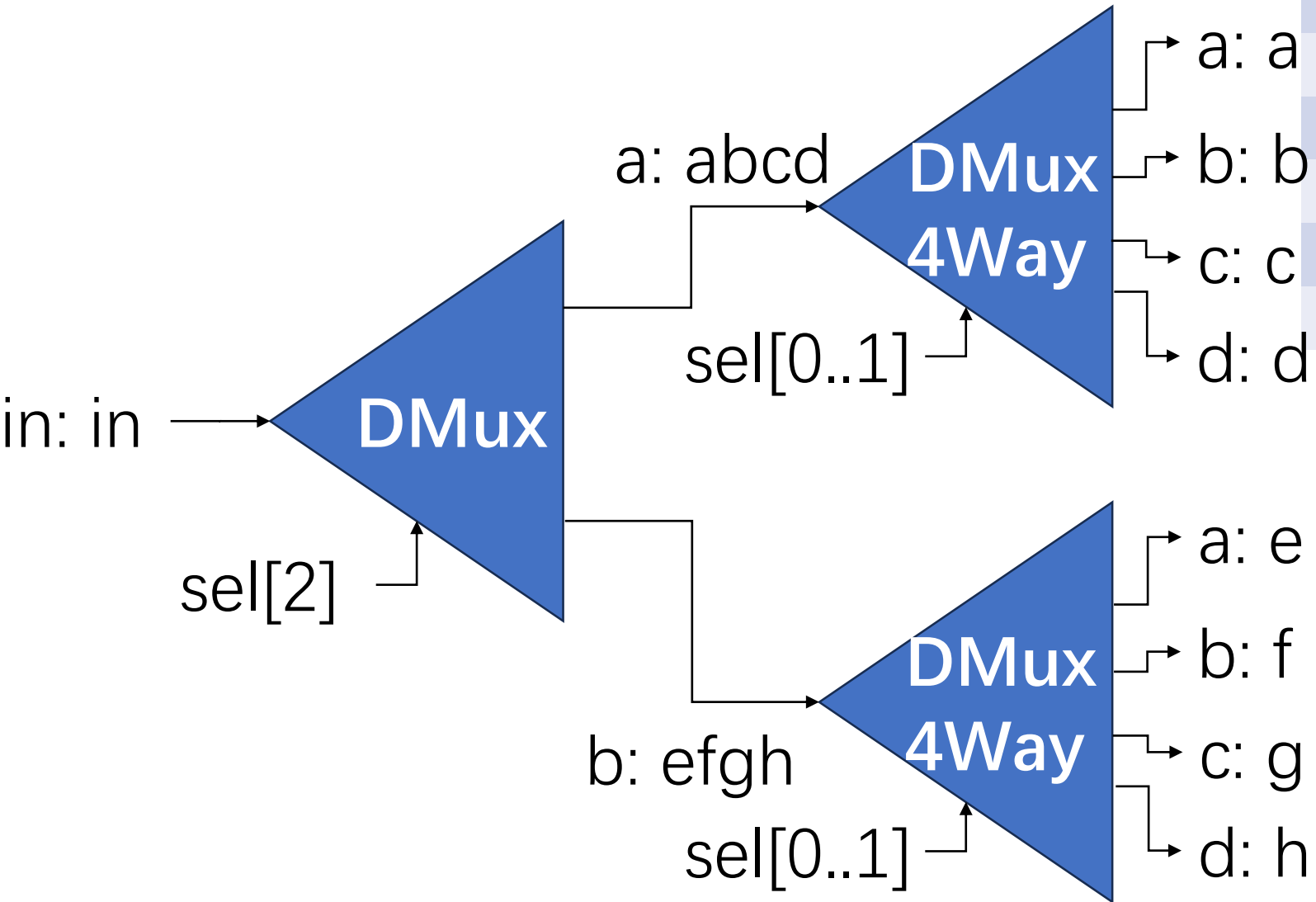


sel[1]	sel[0]	a	b	c	d
0	0	in	0	0	0
0	1	0	in	0	0
1	0	0	0	in	0
1	1	0	0	0	in

DMux4Way

```
CHIP DMux4Way {
    IN in, sel[2];
    OUT a, b, c, d;

    PARTS:
    DMux (in=in, sel=sel[1], a=ab, b=cd);
    DMux (in=ab, sel=sel[0], a=a,  b=b);
    DMux (in=cd, sel=sel[0], a=c,  b=d);
}
```

DMux8Way



sel [2]	sel [1]	sel [0]	a	b	c	d	e	f	g	h
0	0	0	in	0	0	0	0	0	0	0
0	0	1	0	in	0	0	0	0	0	0
0	1	0	0	0	in	0	0	0	0	0
0	1	1	0	0	0	in	0	0	0	0
1	0	0	0	0	0	0	in	0	0	0
1	0	1	0	0	0	0	0	in	0	0
1	1	0	0	0	0	0	0	0	in	0
1	1	1	0	0	0	0	0	0	0	in

DMux8Way

```
CHIP DMux8Way {
    IN in, sel[3];
    OUT a, b, c, d, e, f, g, h;

    PARTS:
    // Binary tree of demultiplexors.
    DMux (sel=sel[2], in=in,   a=abcd, b=efgh);
    DMux (sel=sel[1], in=abcd, a=ab,   b=cd);
    DMux (sel=sel[1], in=efgh, a=ef,   b=gh);
    DMux (sel=sel[0], in=ab,   a=a,    b=b);
    DMux (sel=sel[0], in=cd,   a=c,    b=d);
    DMux (sel=sel[0], in=ef,   a=e,    b=f);
    DMux (sel=sel[0], in=gh,   a=g,    b=h);
}
```

Or you can write by using CHIP DMux4Way.

```
CHIP DMux8Way {
    IN in, sel[3];
    OUT a, b, c, d, e, f, g, h;

    PARTS:
    // Binary tree of demultiplexors.
    DMux (sel=sel[2], in=in,   a=abcd, b=efgh);
    DMux4Way (sel=sel[0..1], in=abcd, a=a, b=b, c=c, d=d);
    DMux4Way (sel=sel[0..1], in=efgh, a=e, b=f, c=g, d=h);
}
```