Cal Poly Pomona

Advanced Computer Architecture (CS 5250.01)

Project # 1

Talha Ahmed

018960110

Submitted to: Professor Nima Davarpanah

Dated: Friday, 30 January 2026

Table of Contents

Solution

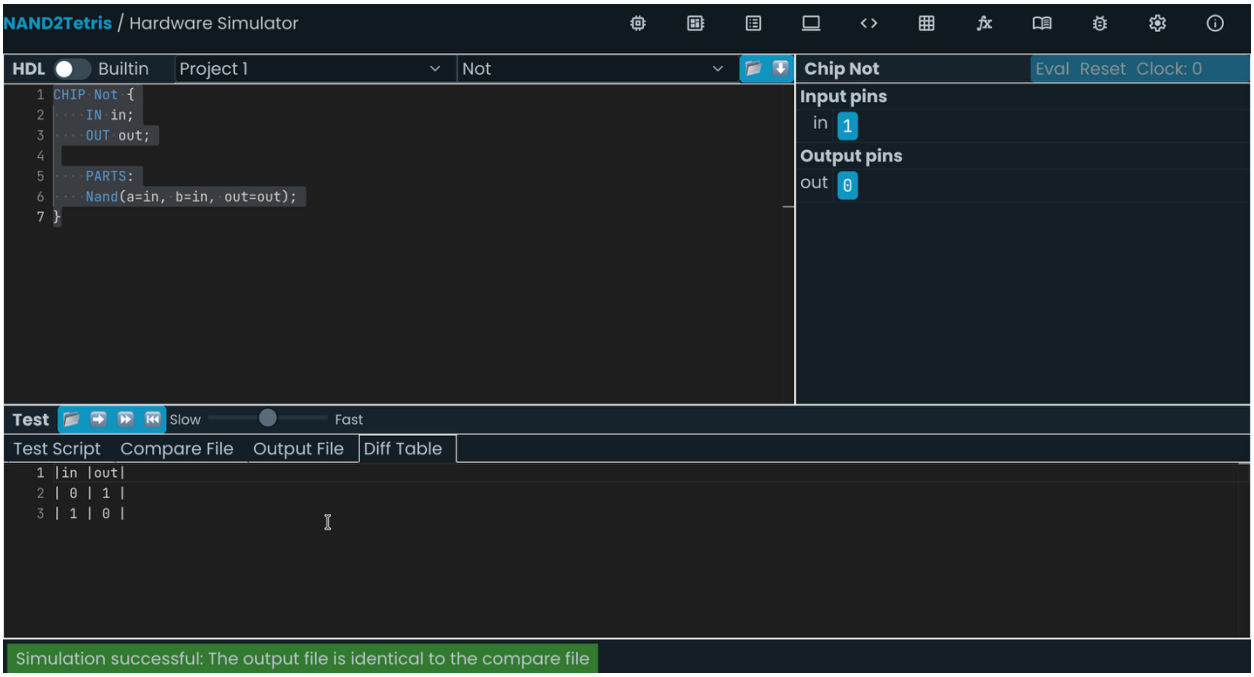
NOT Gate

Code

```
CHIP Not {
    IN in;
    OUT out;

    PARTS:
    Nand(a=in, b=in, out=out);
}
```

Result



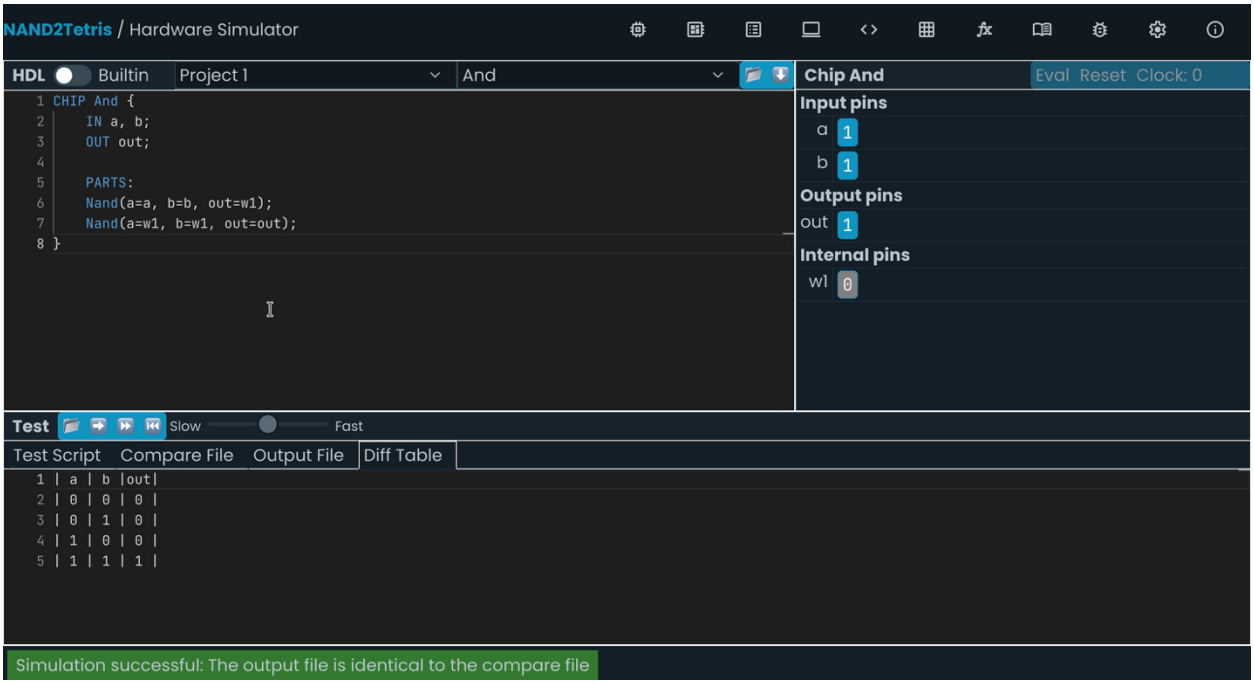
AND Gate

Code

```
CHIP And {
    IN a, b;
    OUT out;
```

```
    PARTS:
    Nand(a=a, b=b, out=w1);
    Nand(a=w1, b=w1, out=out);
}
```

Result



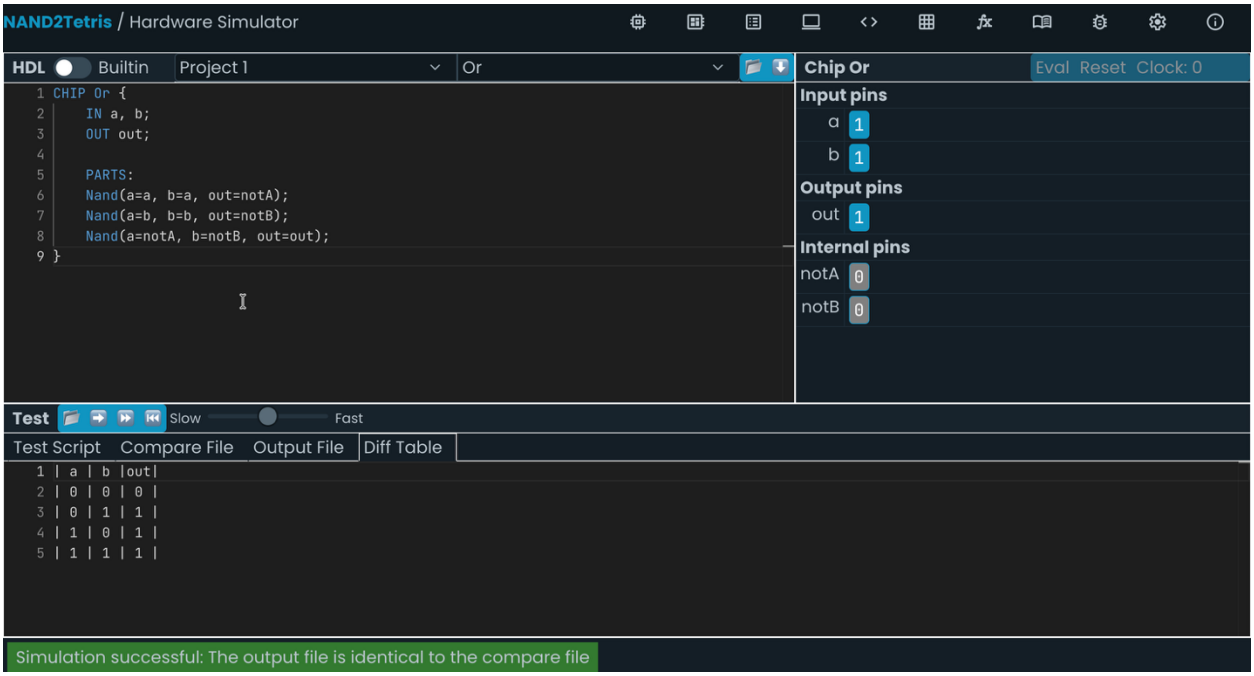
OR Gate

Code

```
CHIP Or {
    IN a, b;
    OUT out;

    PARTS:
    Nand(a=a, b=a, out=notA);
    Nand(a=b, b=b, out=notB);
    Nand(a=notA, b=notB, out=out);
}
```

Result



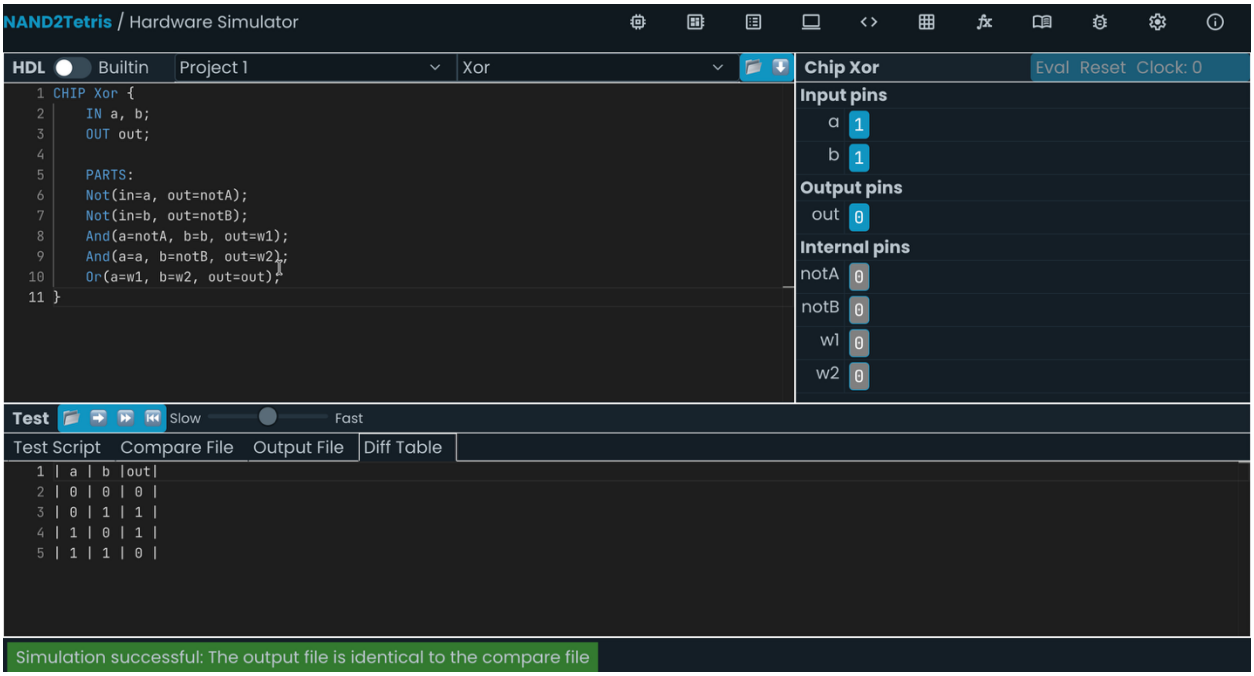
XOR Gate

Code

```
CHIP Xor {
    IN a, b;
    OUT out;

    PARTS:
    Not(in=a, out=notA);
    Not(in=b, out=notB);
    And(a=notA, b=b, out=w1);
    And(a=a, b=notB, out=w2);
    Or(a=w1, b=w2, out=out);
}
```

Result



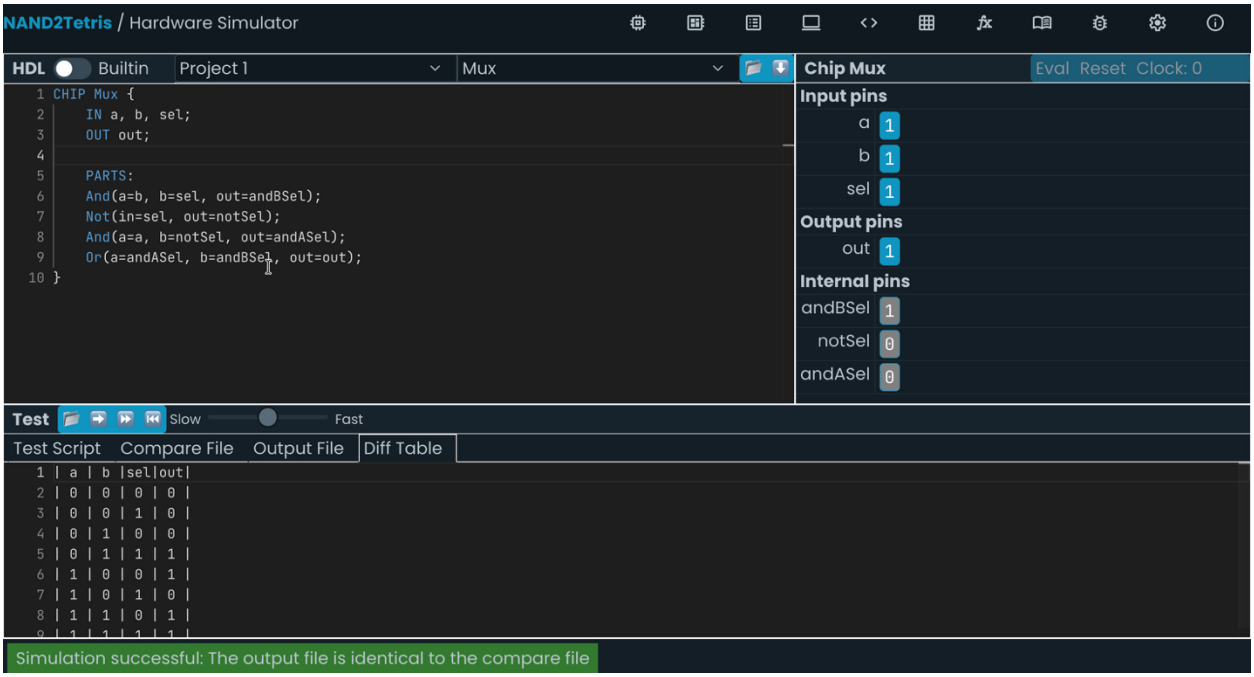
MUX Gate

Code

```
CHIP Mux {
    IN a, b, sel;
    OUT out;

    PARTS:
    And(a=b, b=sel, out=andBSel);
    Not(in=sel, out=notSel);
    And(a=a, b=notSel, out=andASel);
    Or(a=andASel, b=andBSel, out=out);
}
```

Result



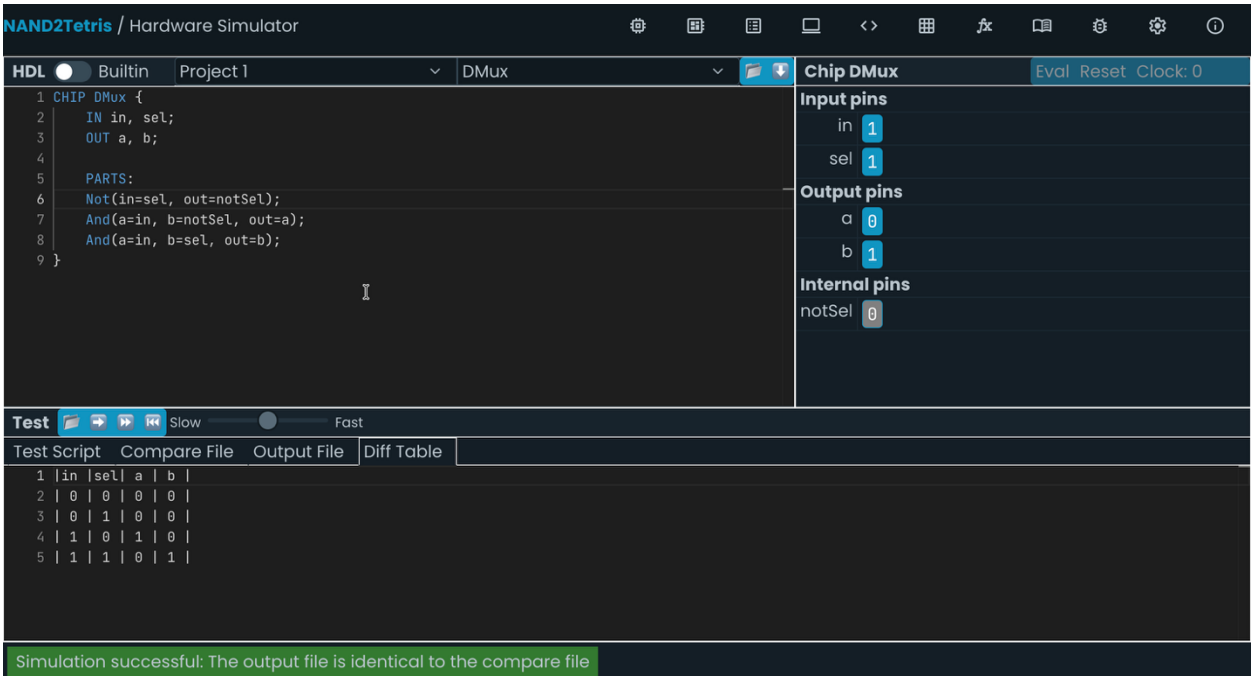
DMUX Gate

Code

```
CHIP DMux {
    IN in, sel;
    OUT a, b;

    PARTS:
    Not(in=sel, out=notSel);
    And(a=in, b=notSel, out=a);
    And(a=in, b=sel, out=b);
}
```

Result



NOT16 Gate

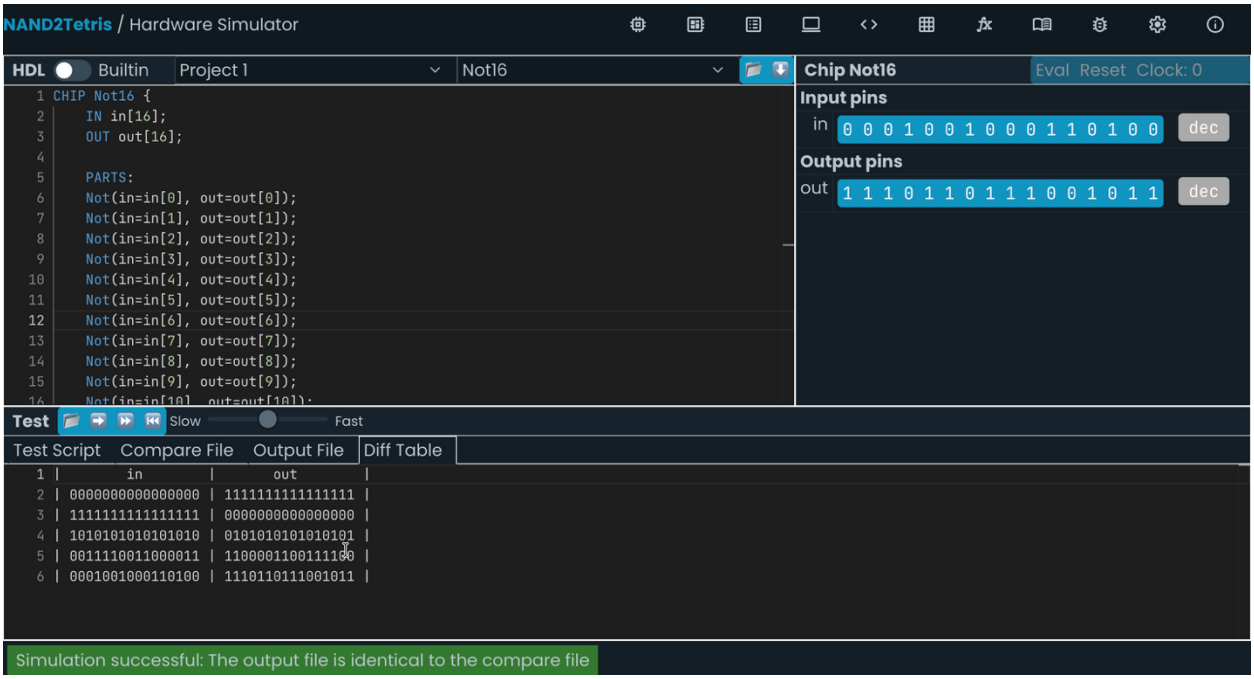
Code

```
CHIP Not16 {
    IN in[16];
    OUT out[16];

    PARTS:
    Not(in=in[0], out=out[0]);
    Not(in=in[1], out=out[1]);
    Not(in=in[2], out=out[2]);
    Not(in=in[3], out=out[3]);
    Not(in=in[4], out=out[4]);
    Not(in=in[5], out=out[5]);
    Not(in=in[6], out=out[6]);
    Not(in=in[7], out=out[7]);
    Not(in=in[8], out=out[8]);
    Not(in=in[9], out=out[9]);
    Not(in=in[10], out=out[10]);
    Not(in=in[11], out=out[11]);
```

```
    Not(in=in[12], out=out[12]);
    Not(in=in[13], out=out[13]);
    Not(in=in[14], out=out[14]);
    Not(in=in[15], out=out[15]);
}
```

Result



AND16 Gate

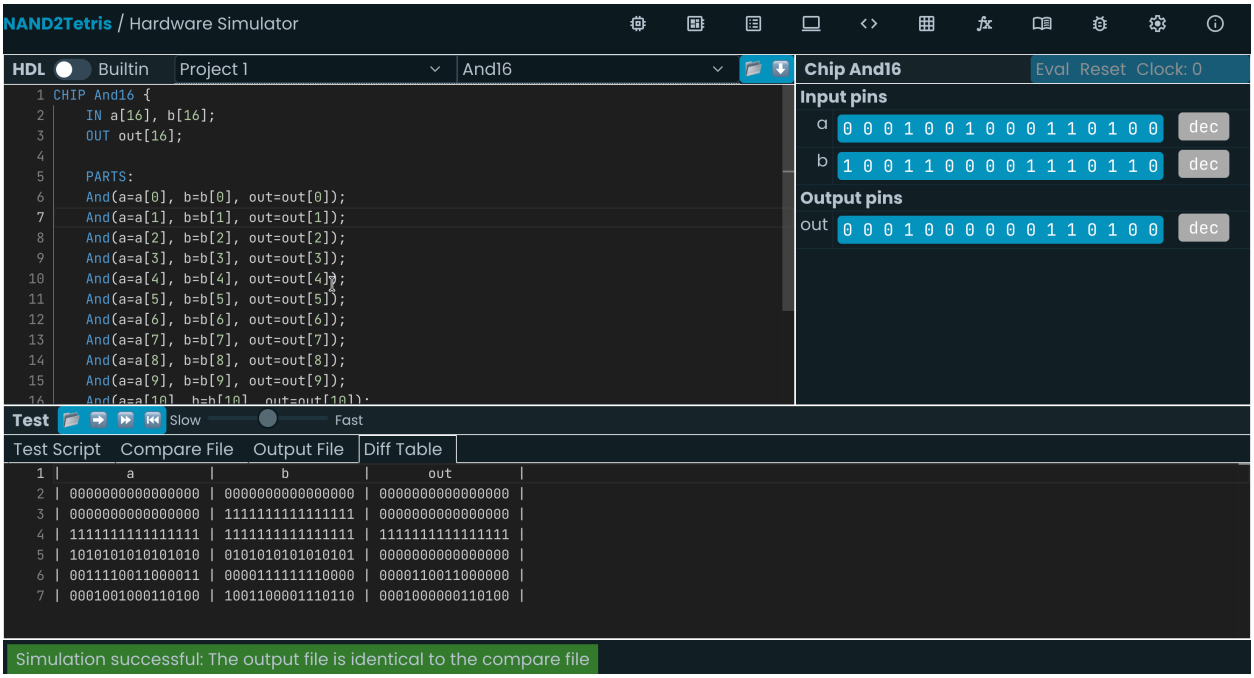
Code

```
CHIP And16 {
    IN a[16], b[16];
    OUT out[16];

    PARTS:
    And(a=a[0], b=b[0], out=out[0]);
    And(a=a[1], b=b[1], out=out[1]);
    And(a=a[2], b=b[2], out=out[2]);
    And(a=a[3], b=b[3], out=out[3]);
    And(a=a[4], b=b[4], out=out[4]);
    And(a=a[5], b=b[5], out=out[5]);
```

```
    And(a=a[6], b=b[6], out=out[6]);
    And(a=a[7], b=b[7], out=out[7]);
    And(a=a[8], b=b[8], out=out[8]);
    And(a=a[9], b=b[9], out=out[9]);
    And(a=a[10], b=b[10], out=out[10]);
    And(a=a[11], b=b[11], out=out[11]);
    And(a=a[12], b=b[12], out=out[12]);
    And(a=a[13], b=b[13], out=out[13]);
    And(a=a[14], b=b[14], out=out[14]);
    And(a=a[15], b=b[15], out=out[15]);
}
```

Result



OR16 Gate

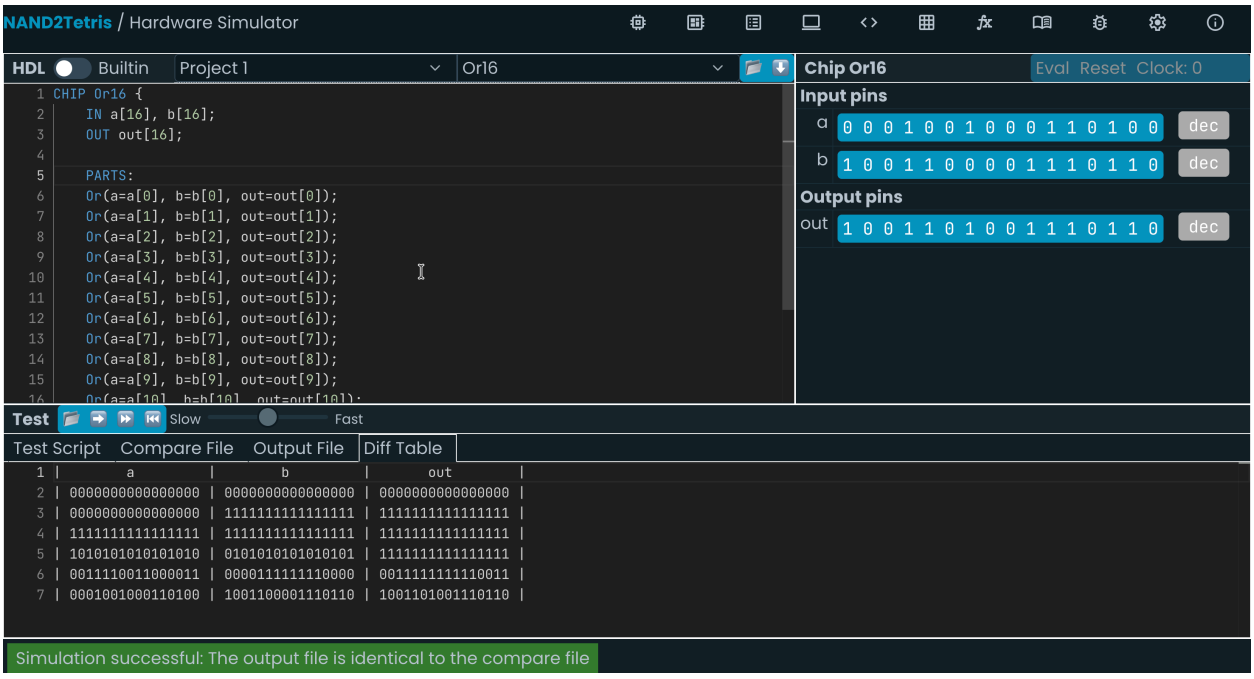
Code

```
CHIP Or16 {
    IN a[16], b[16];
    OUT out[16];

    PARTS:
```

```
    Or(a=a[0], b=b[0], out=out[0]);
    Or(a=a[1], b=b[1], out=out[1]);
    Or(a=a[2], b=b[2], out=out[2]);
    Or(a=a[3], b=b[3], out=out[3]);
    Or(a=a[4], b=b[4], out=out[4]);
    Or(a=a[5], b=b[5], out=out[5]);
    Or(a=a[6], b=b[6], out=out[6]);
    Or(a=a[7], b=b[7], out=out[7]);
    Or(a=a[8], b=b[8], out=out[8]);
    Or(a=a[9], b=b[9], out=out[9]);
    Or(a=a[10], b=b[10], out=out[10]);
    Or(a=a[11], b=b[11], out=out[11]);
    Or(a=a[12], b=b[12], out=out[12]);
    Or(a=a[13], b=b[13], out=out[13]);
    Or(a=a[14], b=b[14], out=out[14]);
    Or(a=a[15], b=b[15], out=out[15]);
}
```

Result



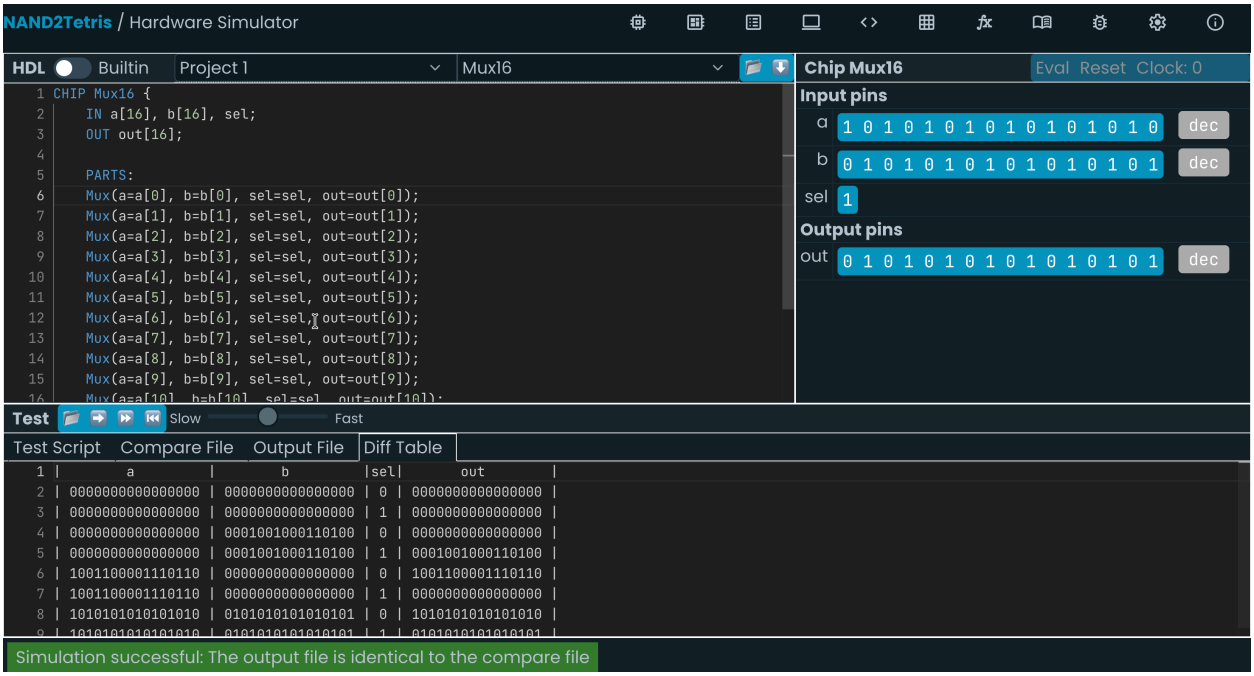
MUX16 Gate

Code

```
CHIP Mux16 {
    IN a[16], b[16], sel;
    OUT out[16];

    PARTS:
    Mux(a=a[0], b=b[0], sel=sel, out=out[0]);
    Mux(a=a[1], b=b[1], sel=sel, out=out[1]);
    Mux(a=a[2], b=b[2], sel=sel, out=out[2]);
    Mux(a=a[3], b=b[3], sel=sel, out=out[3]);
    Mux(a=a[4], b=b[4], sel=sel, out=out[4]);
    Mux(a=a[5], b=b[5], sel=sel, out=out[5]);
    Mux(a=a[6], b=b[6], sel=sel, out=out[6]);
    Mux(a=a[7], b=b[7], sel=sel, out=out[7]);
    Mux(a=a[8], b=b[8], sel=sel, out=out[8]);
    Mux(a=a[9], b=b[9], sel=sel, out=out[9]);
    Mux(a=a[10], b=b[10], sel=sel, out=out[10]);
    Mux(a=a[11], b=b[11], sel=sel, out=out[11]);
    Mux(a=a[12], b=b[12], sel=sel, out=out[12]);
    Mux(a=a[13], b=b[13], sel=sel, out=out[13]);
    Mux(a=a[14], b=b[14], sel=sel, out=out[14]);
    Mux(a=a[15], b=b[15], sel=sel, out=out[15]);
}
```

Result



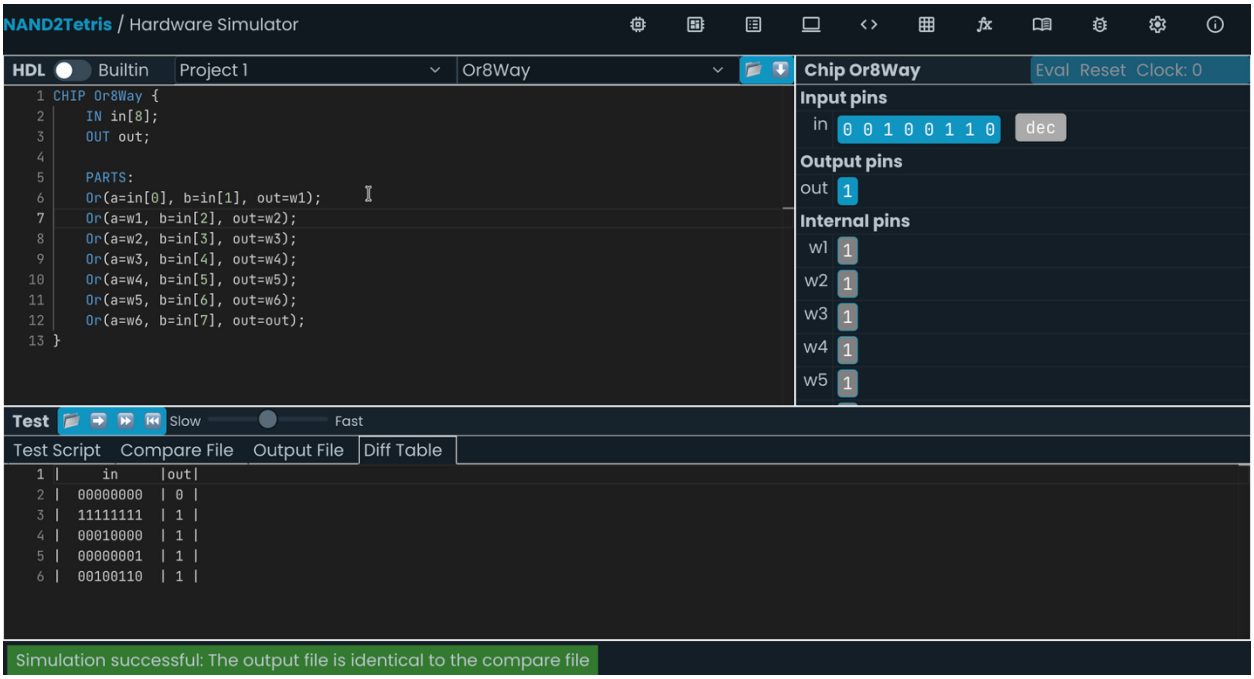
OR8WAY Gate

Code

```
CHIP Or8Way {
    IN in[8];
    OUT out;

    PARTS:
    Or(a=in[0], b=in[1], out=w1);
    Or(a=w1, b=in[2], out=w2);
    Or(a=w2, b=in[3], out=w3);
    Or(a=w3, b=in[4], out=w4);
    Or(a=w4, b=in[5], out=w5);
    Or(a=w5, b=in[6], out=w6);
    Or(a=w6, b=in[7], out=out);
}
```

Result



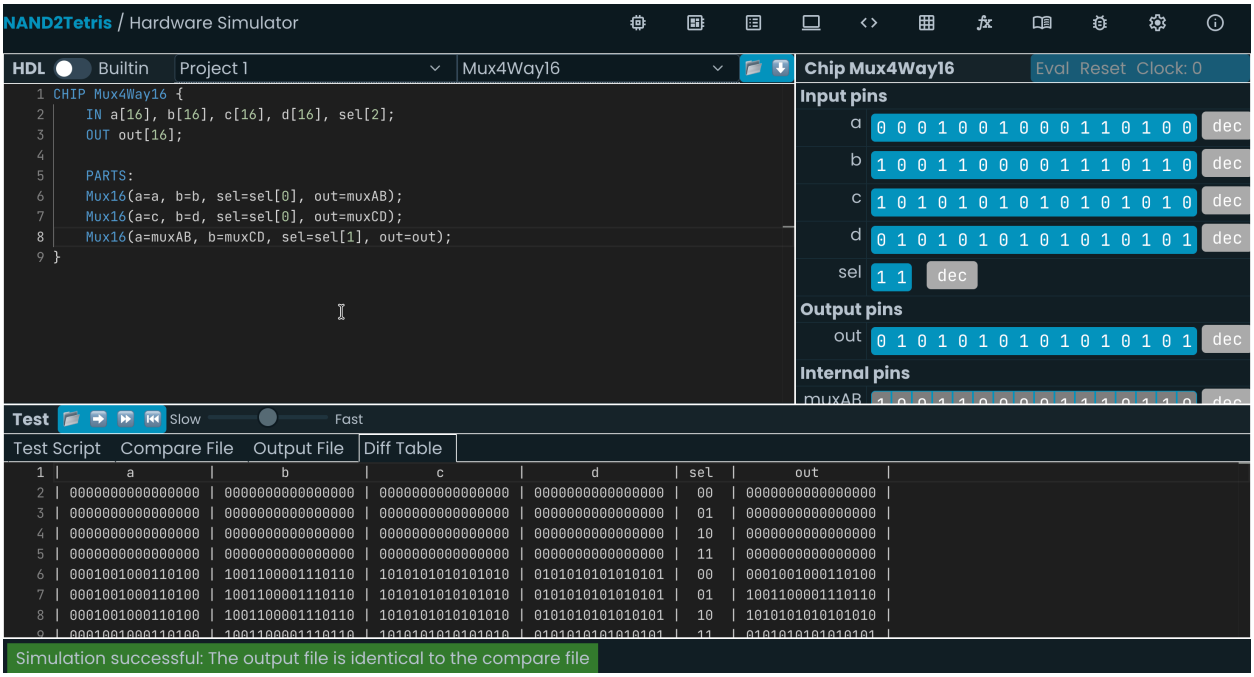
MUX4WAY16 Gate

Code

```
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=muxAB);
    Mux16(a=c, b=d, sel=sel[0], out=muxCD);
    Mux16(a=muxAB, b=muxCD, sel=sel[1], out=out);
}
```

Result



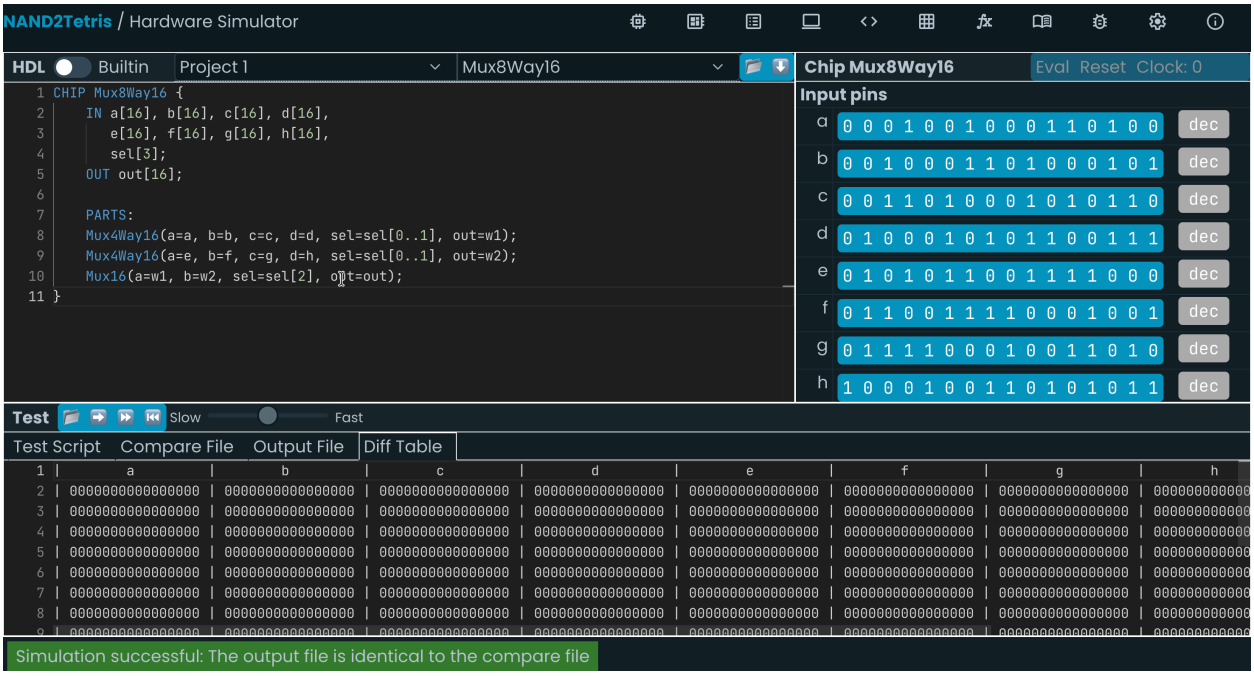
MUX8WAY16 Gate

Code

```
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:
    Mux4Way16(a=a, b=b, c=c, d=d, sel=sel[0..1], out=w1);
    Mux4Way16(a=e, b=f, c=g, d=h, sel=sel[0..1], out=w2);
    Mux16(a=w1, b=w2, sel=sel[2], out=out);
}
```

Result



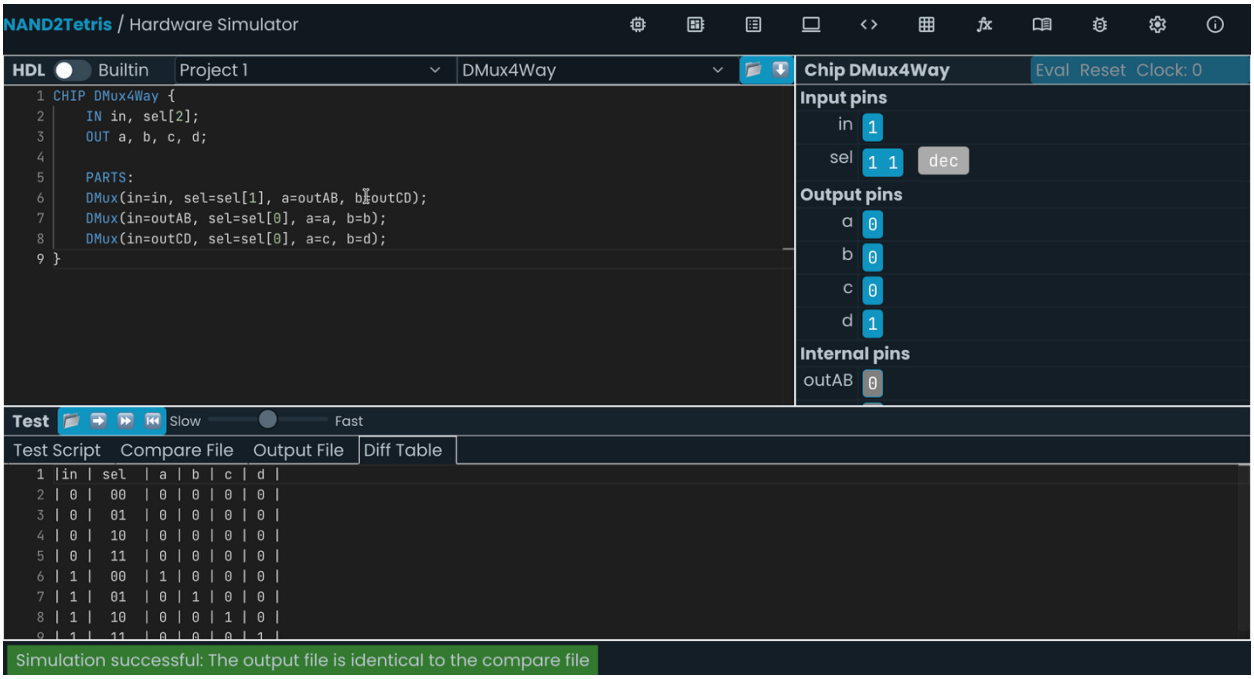
DMUX4WAY Gate

Code

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

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

Result



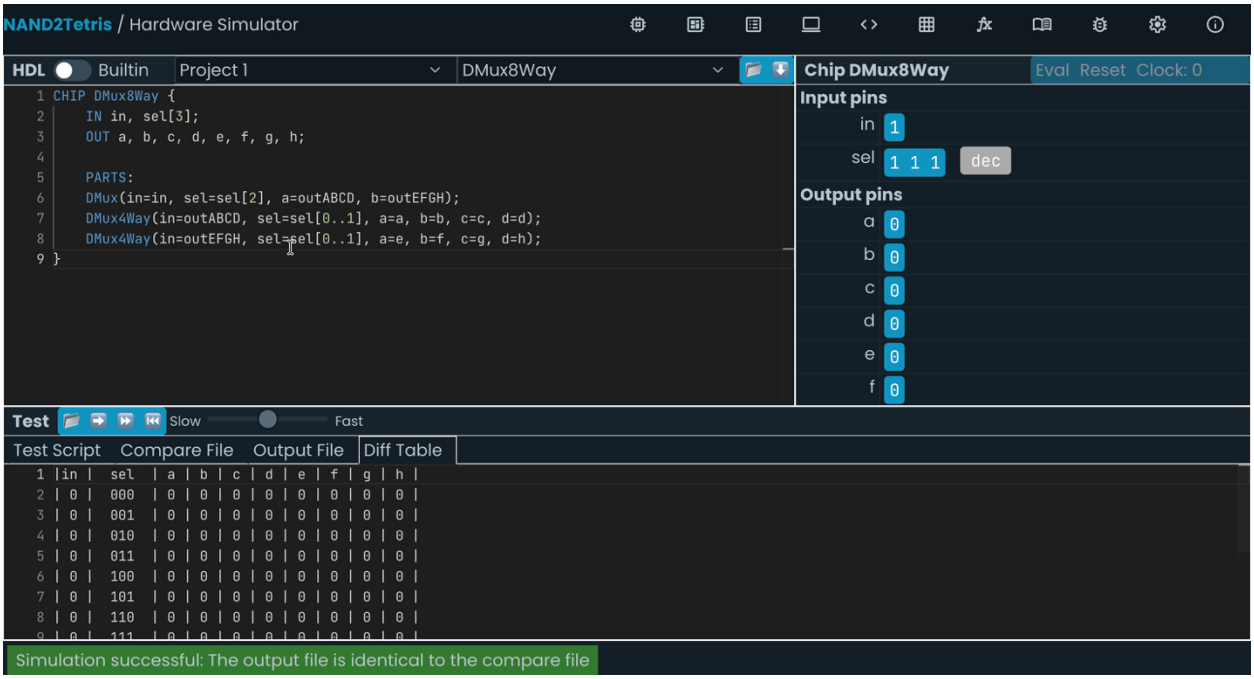
DMUX8WAY Gate

Code

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

    PARTS:
    DMux(in=in, sel=sel[2], a=outABCD, b=outEFGH);
    DMux4Way(in=outABCD, sel=sel[0..1], a=a, b=b, c=c, d=d);
    DMux4Way(in=outEFGH, sel=sel[0..1], a=e, b=f, c=g, d=h);
}
```

Result


NAND2Tetris / Hardware Simulator
HDL
Builtin
Project 1
DMux8Way
CHIP DMux8Way {
IN in, sel[3];
OUT a, b, c, d, e, f, g, h;
PARTS:
DMux(in=in, sel=sel[2], a=outABCD, b=outEFGH);
DMux4Way(in=outABCD, sel=sel[0..1], a=a, b=b, c=c, d=d);
DMux4Way(in=outEFGH, sel=sel[0..1], a=e, b=f, c=g, d=h);
}
Chip DMux8Way
Eval Reset Clock: 0
Input pins
in 1
sel 1 1 1
dec
Output pins
a 0
b 0
c 0
d 0
e 0
f 0
Test
Slow
Fast
Test Script
Compare File
Output File
Diff Table
in	sel	a	b	c	d	e	f	g	h
0	000	0	0	0	0	0	0	0	0
0	001	0	0	0	0	0	0	0	0
0	010	0	0	0	0	0	0	0	0
0	011	0	0	0	0	0	0	0	0
0	100	0	0	0	0	0	0	0	0
0	101	0	0	0	0	0	0	0	0
0	110	0	0	0	0	0	0	0	0
Simulation successful: The output file is identical to the compare file