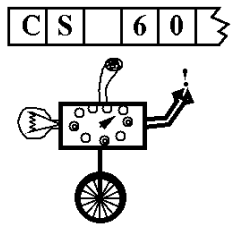


Combinational Logic: Conclusion

November 14, 2001

A 4-D Karnaugh Map

For inputs yxvw:

yx \ vw	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	1

is this the only way to draw the diagram?

Prime Implicants

- Another term for maximal sub-cubes
  - Particularly when referring to the corresponding formulas.
- The simplest SOP form for a function is a sum (logical-or) of prime implicants.
- But, not necessarily *all* of the prime implicants

Implementation using NAND gates

- In some families of logic, and- and or- gates might not be available.
- Instead the only gate is nand (negated and).
- The question of sufficiency arises.

Sufficiency

- We know that functions `and`, `or`, and `not` are sufficient to realize any logic function.
- Two proofs:
  - minterm expansion: a sum of minterms, and each minterm uses `and` and `not` only
  - Boole-Shannon expansion (applied recursively)

Sufficiency

- To show that a given set of gate types is sufficient, it is enough to show that `and`, `or`, and `not` can be realized with those gate types.
- Actually it is sufficient to show just `and` and `not` can be realized, since `or` can be realized as:
  `a + b = (a' b')'`
  by DeMorgan's law. So if `and` and `not` are realizable, so is `or`.

nand alone is sufficient

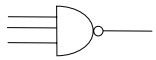
```
a' = nand(a, a)

   and(a, b)
 = nand(a, b)'
= nand(nand(a,b), nand(a,b))

    or(a, b)
 = nand(a', b')
= nand(nand(a,a), nand(b,b))
```

Exercises

- Show that `nor` alone is sufficient.
- Is `xor` alone sufficient?

Bubble Logic

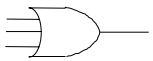
nand



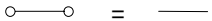
DeMorgan:

```
nand(a, b) =
and(a, b)' =
or(a', b')
```

or

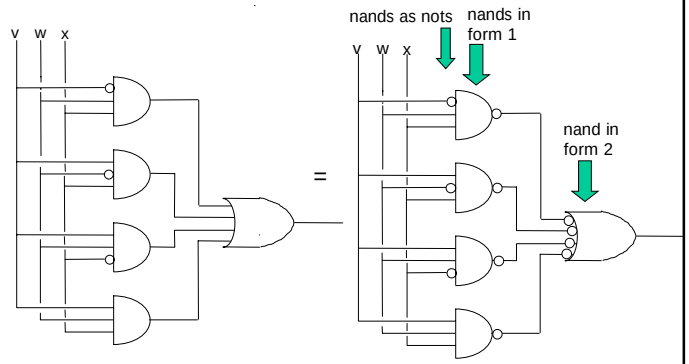


nand, another way



(a')' = a

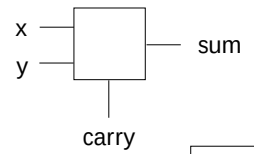
SOP form using nand only



Common Logic Packages

- 1-bit adder (half adder)
- 1-bit adder with carry-in (full adder)
- 4-bit adder
- decoder (binary to one-hot)
- encoder (one-hot to binary)
- (MUX) multiplexor
- (DMUX) demultiplexor

Half Adder

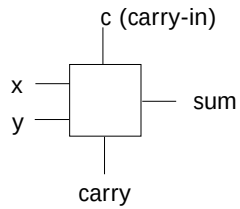


exclusive or

$sum(x, y) = x \oplus y = xy' + x'y$

$carry(x, y) = x\ y$

Full Adder

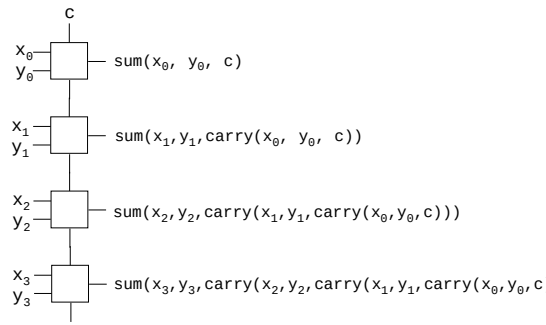


$sum(x, y, c) = x \oplus y \oplus c$

$carry(x, y, c) = xy + xc + yc$

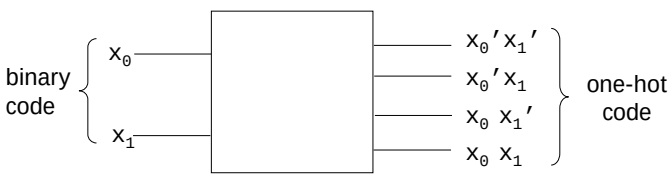
$= majority(x, y, c)$

4-bit adder

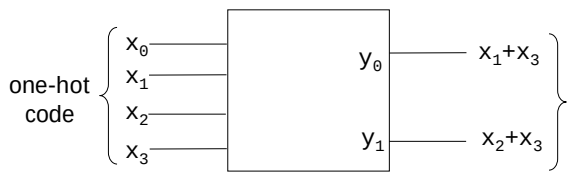


Note: There are other ways to implement this.

4-bit decoder

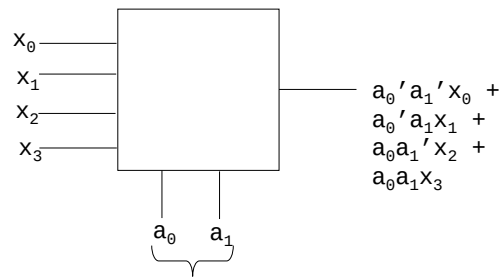


4-bit encoder

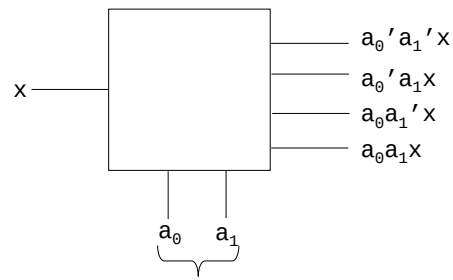


	x_0	x_1	x_2	x_3	y_1	y_0
unlisted combinations are don't-care	1	0	0	0	0	0
	0	1	0	0	0	1
	0	0	1	0	1	0
	0	0	0	1	1	1

Multiplexor



demultiplexor



Exercises

- How would you build a decoder out of a demultiplexor?
- How would you build a 16-way multiplexor out of 4-way multiplexors?
- How would you build a 16-way demultiplexor out of 4-way demultiplexors?