Digitaltechnik

Contents

MSB - Most significant bit
LSB - Least significant bit

$x\%1$ - drop decimal value from x.
2^n - number of possible states with n bits.

Resistance of a wire

ρ - resistivity of the metal (Ωm)
l - length of the wire
A - cross sectional area of the wire
$R = \frac{\rho l}{A}$

Modern electronics uses 0.8V as high.

Floating Voltage - when a pin / contact is not connected by a "normal" (lower than that of air) resistance to V_DD / circuit ground. Essentially the same as any conductive surface in the room, on which a very weak 50Hz signal is usually seen due to induction from all the EM sources in the room.

Schaltfunktionen

Schaltfunktion - $Y = f(X_0, X_1, X_2, ..., X_{N-1})$ - Nimmt mehrere Bits als Input und produziert eine einzige Bit als Ausgang.

Alle Schaltfunktionen lassen sich als einer Wahrheitstabelle darstellen mit mindestens $N + 1$ Spalten und 2^N Zeilen, wo N ist der Nummer von Inputs.

NOT'ing a gate usually means the resistor just needs to be moved before the transistors (essentially appending a NOT gate).

OR - Disjunction

AND - Conjunction - The resistor after the output point is needed to prevent a short circuit when both inputs are high.

Antivalenz (XOR) - High if only one of the inputs is high.

XNOR - High if both inputs are the same, gate symbol is a =.

CMOS (Complementary Metal Oxide Semiconductor) Technology

Transistor - Trans-Resistor (changable resistor)
MOS Transistor - Electronic component with contacts **S** ource, **D** rain und **G** ate. Charge carriers flow from S to D. They are always controlled through a voltage between Gate and Source (unlike a current with BJT) and is therefore more efficient for very low / high power applications. They are also easier to etch in ICs and are therefore predominantly used in logic circuits.

Although very high pull up resistors vastly reduce power loss when using a single MOS transistor, such large resistances are difficult to fabricate in ICs. CMOS uses a PMOS instead which has practically ∞ resistance when "open".

$|V_{GS}| < |V_{th}|, R_{SD} \to \infty$ - The transistor is off
$|V_{GS}| > |V_{th}|, R_{SD} \to 0$ - The transistor is on

N-Type (NMOS) - Threshold voltage is positive. Negative electrons flow from S to D (Hence D is connected to the positive terminal in a circuit)
P-Type (PMOS) - Threshold voltage is negative. Positive Holes flow from S to D. Circle at the gate in symbol.

- CMOS Gatter müssen aus genau so vielen NMOS und PMOS Transistoren bestehen
- Bei m Eingängen gibt es m NMOS und m PMOS transistoren

The V_D of an "off" MOS transistor is floating (undefined) unless it is pulled up / down.

A CMOS gate can be split into two networks / Pfads:

	Pull-up	Pull-down
MOS Type	PMOS	NMOS
NAND	Parallel	Series
NOR	Series	Parallel

These can be converted between one another by breaking the circuit into parallel / series blocks until each block contains one transistor, then switching the type of transistor and connecting them again in the opposite manner (parallel ⇔ series). V_DD becomes the output and the output becomes ground.

Switching Delays

- t_{pHL}, t_{pLH} - Propogation delay - Time taken between a 50% change in the input voltage leading to a 50% change in the output
- $t_{tHL}(t_{\text{fall}}), t_{tLH}(t_{\text{rise}})$ - Time between the output rising / falling between 10% and 90% voltage

$t_d = \frac{t_{pHL}+t_{pLH}}{2}$ - Average switching time, easier to work with in practice

Boolean Algebra

The following properties apply to an expresion only containing AND / OR gates:
- Commutative, order does not matter
- Associative, grouping / order of (the same) operations is irrelevant
- Distributive, $A \land (B \lor C) = (A \land B) \lor (A \land C)$

Some not so obvious axioms of boolena algebra:
- $A \lor (A \land B) = A = A \land (A \lor B)$ - consider the effect on the whole circuit when the outer variable is high / low
- $(A \land B) \lor (\neg A \land B) = (A \lor B) \land (\neg A \lor B) = B$ - Neighbourhood law

Order of Operations

1. Brackets
2. Negation
3. AND, NAND, OR, NOR
4. XOR, XNOR

An expression with missing brackets is ambiguous and invalid.

De Morgan's Laws

$$\neg(A \land B \land C \land \ldots) \equiv \neg A \lor \neg B \lor \neg C \lor \ldots$$
$$\neg(A \lor B \lor C \lor \ldots) \equiv \neg A \land \neg B \land \neg C \land \ldots$$

The conversion between the pull up and pull down expression in a CMOS circuit uses De Morgan's laws:

$$\mathbf{Y_{pd}} = \overline{((A \cdot (B + C) \cdot D) + F + (G \cdot H)) \cdot E}$$
$$\mathbf{Y_{pu}} = ((\bar{A} + (\bar{B} \cdot \bar{C}) + \bar{D}) \cdot \bar{F} \cdot (\bar{G} + \bar{H})) + \bar{E}$$

Universal Gates

Any logic circuit can be expressed using only NAND / NOR gates. This is very advantageous as all gates in the circuit would have the same timing properties, reducing costs and errors.

To convert a logical expression into NAND / NOR, double negation + De Morgan's laws can be used, for example:

$$A \land B \equiv ?$$
$$\neg\neg A \land B \equiv \neg\neg A \lor \neg B$$
$$\neg A \text{ NOR } \neg B \equiv (A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$$

Min / Maxterms

For n variables there are 2^n possible min / maxterms:
- Minterm: Expression that outputs 1 for **one** specific combination of inputs, if we want 1 only when A low, B High, minterm: $\neg A \land B$
- Maxterm: Expression that outputs 0 for **one** specific combination of inputs, if we want 0 only when

Let us consider we want expressions that output high / low only when $A = 0, B = 1$:

- Minterm: $\neg A \land B$, high only in this case
- Maxterm: $A \lor \neg B$, low only in this case

Normal Forms

A way of expressing a boolean expression that can easily be determined from the desired truth table (and later simplified). After constructing either a list of minterms for each 1 in the output, or list of maxterms for each 0:

- Disjunctive Normal Form - Minterms joined using disjunctions (OR)
- Conjunctive Normal Form - Maxterms joined using conjunctions (AND)

Both result in the same output, DNF is more comfortable to use in Karnaugh diagrams.

Karnaugh Diagrams

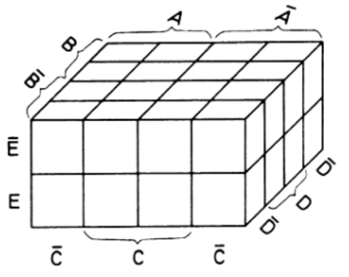
Used to visually and systematically simplify boolean expressions instead of through often complicated boolean algebra manipulation. Furthermore, race conditions (hazards) can be easier identified using this method.

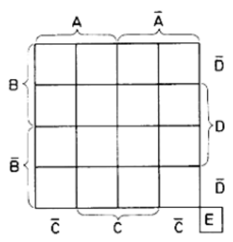
They are analogous to a truth table but represented as a matrix with 2^n elements, where n is the number of inputs.

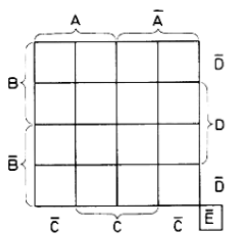
Simply a methodical way of using the neighbour simplification rule: $(\neg A \land \neg B) \lor (A \land \neg B) = \neg B$

Method

- Construct a matrix with 2^n elements, where each side of the matrix representsthe two states of a variable. One of each pair of variables facing opposite one another must be "split" so the neighbour rule can still be applied (See examples).
- Each element contains a 1 for each minterm (DNF) in the truth table based on ANDing the row / column headers
- Create packets (also known as blocks) using the largest possible rectangle with 1s. The packets must contain 2^n cells!
- Packets may overlap and "pacman" over the border (even diagonally!), but not take non rectangle shapes (for example an L shape).
- A minterm is then built from ORing packets together and the neighbour rule is applied: ie. only the variable(s) that remains the same in all cells of the packet remain in the resulting minterm of the packet.







Don't care - Combinations of inputs for which the output doesn't matter, for example additional unneeded numbers in a boolean counting system. Marked with an X in a Karnaugh Diagram. The X's can be treated as 1s when creating packets if it reduces the amount of packets (and therefore OR gates) in the simplified expression.

Static hazards - When the same variable is used in a parent logic gate, changes in the variable can lead to delayed "notches" in the parent's output due to time delays. These can be recognized in Karnaugh diagrams: where two packets are orthogonally next to each other but do not overlap. They can be directly fixed by introducing an extra packet two join the place of the hazard - this results in more gates overall but avoids the hazard.

Number Systems

Base (Radix) - b-adischen Reiehen like in analysis, negative indices of the base for defining decimals

Hexadecimal - Uses digits $0 - 9$ and $A - F$ for 16 possible digits in total. Used to represent binary numbers in a more compact format by splitting a binary number into groups of 4 digits

Octal - Radix 8, can be converted from binary using groups of 3 binary digits.

Converting Decimal to Radix R

1. Perform whole number division of the decimal D by the Radix R: $\frac{D}{R} = Q_0 + r_0$, the remainder is the first digit in the target radix
2. Divide the result of the previous whole-number division Q_0 by the radix R again, this remainder is now the second digit in the target radix and so on
3. Continue until Q_i reaches 0

Converting $0 \leq D_{10} < 1$ to Radix R

This is the same process but the decimal is multiplied by the radix R and the resulting product is used in the next multipliciation. The current digit is the floor of TODO: Coefficients?, starting with the most significant bit.

Only possible for a finite number of decimal digits.

Signed Binary Numbers

- Signing bit
- 1s complement
- 2s complement: Advantageous for performing arithmetic with signed numbers as the sign remains accurate

2s Complement

TODO: Screenshot from script (Konstruktion von 2er-Komplementen) How it can be converted back into decimal, either convert to positive then calculate or use first bit (sign) as $-(2)^n$ and the others as positive binary digits

Fractional numbers still have $\pm 2^0$ as the signing bit, converted in the same way. In the case of a whole + fractional number only the first bit is a signing bit

IMPORTANT: Do not forget signing bit for positive numbers

Binary Arithmetic

Addition of two binary numbers, with maximum n digits has at most $n + 1$ bits in the result

Binary subtraction can be written as the addition of 2s complement numbers

Encoding

Tetraden / Nibble - groups of 4 bits

Many different ways to encode 10 numbers, each has their advantages / disadvantages. Gray / O'Brien useful for counting TODO: Why?

Parity Bit

Additional bit representing if the number of 1s in a word / block is odd / even TODO: Double check

An extra word can be sent with the purpose of checking and also correcting previous words