Reversible Computing for Beginners

Lecture 3.

Marek Perkowski

Some slides from Hugo De Garis, De Vos, Margolus, Toffoli, Vivek Shende & Aditya Prasad

Reversible Computing

• In the 60s and beyond, CS-physicists considered the **ultimate limits** of computing, e.g.

  • What is the maximum bit processing rate of a cubic centimeter of material?

  • What is the minimum amount of heat generated per bit processed? etc.

• This led to the fundamental developments in computing - reversible logic.

The Most Important aspect of research is Motivation

Why I have motivation to work on Reversible Logic?

Reasons to work on Reversible Logic

- *Build Intelligent Robots*
- *Save Power*
- *Save our Civilization*
  - *and supremacy of advanced nations?*

How to build extremely large finite state machines with small power consumption??

…and this leads us to the second reason…..

Save our Civilization

Quantum Computers will be reversible

Quantum Computers will solve NP-hard problems in polynomial time

If Quantum Computers will be not build, USA and the world will be in trouble

Motivation for this work:

Quantum Logic touches the future of our civilization

- We live in a very exciting time.
- US economy grows, despite crisis
- World economy grows
- Thanks to advances in information-processing technology.
- Usefulness of computers has been enabled primarily by semiconductor-based electronic transistors.

Moore's Law

- In 1965, Gordon Moore observed a trend of increasing performance in the first few generations of integrated-circuit technology.
- He predicted that in fact it would continue to improve at an exponential rate - with the performance per unit cost increasing by a factor of 2 every 18 months or so - for at least the next 10 years.
- The computer industry has followed his prediction throughout for 45 years..
- Increased power of computers created a new civilization:
  - communications, manufacturing, finance, and all manner of products and services.
  - It has affected nearly everything.

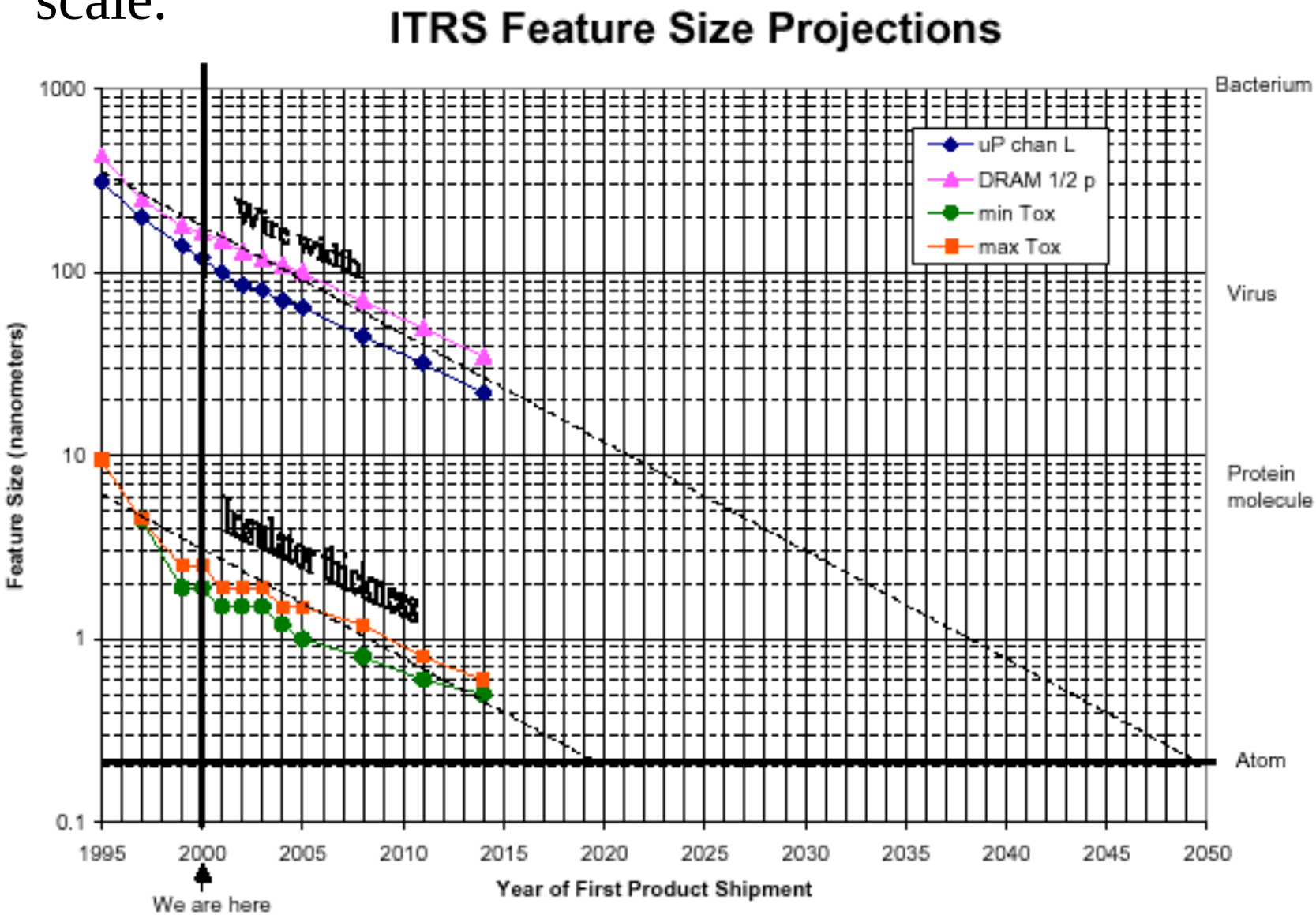
Questions.

- How long can Moore's law continue to hold?
- What are the ultimate limits, if any, to computing technology?
- How will the technology need to change in order to improve as much as possible?
- What will happen to our civilization if the Moore Law will stop to work?

Amazing thing

- A product that seems to get better in every respect as you make it smaller and smaller.
- You *can* make it smaller as you improve the fabrication process in a fairly methodical way.
- But, how long can this trend continue?
- There are a lot of fundamental physical limits that will ultimately come to bear on the shrinkage of transistors, and the improvement of computing technology in general.

- Even beyond this, after not too many decades of Moore's law, the entire transistor itself will approach the atomic scale.



Problems in Reversible/Quantum Computing

- Building physically Reversible Gates in quantum, optical or CMOS technologies (Fredkin, Feynman, Toffoli gates)
  - this is done by physicists and material science people,
  - requires deep understanding of quantum mechanics,
  - big costs, expensive labs.
- Solving NP-hard problems in polynomial time using hypothetical gates and computers (Deutsch)
  - this is done by mathematicians and computer scientists in IBM, Bell Labs, MIT and other top places.
  - Does not require understanding of physics of quantum mechanics but only its mathematics.
  - We can do it if we learn more math !!

Problems in Reversible/Quantum Computing

- Building physically Reversible Computers from CMOS gates (MIT, USC, Seoul Nat. Univ. Korea, De Vos - Belgium).
  - this is done by EE scientists with background in CMOS design.
  - Does not require understanding of quantum mechanics,
  - we can do this.
- Designing new Reversible and Quantum gates . (De Vos, Kerntopf, Picton). This requires elementary knowledge of logic synthesis. Topic of this lecture. We can do this. Relatively easy.
- Designing logic synthesis theory for reversible and Quantum Logic
  - this is done by mathematicians and computer scientists with no background in logic synthesis
  - Picton and other, weak results, few publications
  - This is a virgin field waiting for pioneers.
  - We should do this - see this lecture.

Landauer Principle and What is Reversible Logic Gate?

Landauer's Principle

In 1961, Landauer was considering the smallest amount of heat generated per bit processed in computation.

He made several discoveries and innovations. He introduced the distinction of **logical and physical irreversibility**. He noted that a physical implementation of a logically irreversible process (defined to be one that cannot be logically reversed, i.e. undone by reversing the flow direction of computation) must be physically irreversible (i.e. cannot be undone or reversed to its prior physical state).

A process is **logically reversible** if knowing the binary input to a logic gate, one can deduce the output and **vice versa**. (There is a 1 : 1 mapping between input bit string and output bit string (bijective ,math) e.g. of a logically reversible gate is the NOT gate. An example of a logically irreversible gate is the AND gate.

Landauer's Principle

Knowing one has a 0 at the output of an AND gate does not allow one to deduce what the input was. This gate is logically irreversible.

This is not surprising because there must be loss of information. The AND gate has 2 input bits and only 1 output bit. Information (one bit's worth) has been destroyed!

Landauer discovered (rather surprisingly) that the heat coming from computation was **due to the *destruction* of information** (wiping out bits of information) and *not to the processing of bits*. This is **Landauer's Principle**.

Landauer thought that since computers used logically irreversible gates, (I.e. typically NAND gates) that computing was inherently physically irreversible, and hence inevitably gave off heat.

Classical gates are irreversible

- Conventional computers are built from basic building blocks, such as the **AND, NAND, OR, NOR, and XOR gates.**
- **Such tables are logically irreversible.**
- **This means that, if we forget the value of the input (*A, B*), knowledge of the output *P* is not sufficient to calculate backwards and to recover the value of (*A, B*).**

AB	*P*
0 0	0
0 1	0
1 0	0
1 1	1

(a)

AB	*P*
0 0	1
0 1	1
1 0	1
1 1	0

(b)

AB	*P*
0 0	0
0 1	1
1 0	1
1 1	1

(c)

AB	*P*
0 0	1
0 1	0
1 0	0
1 1	0

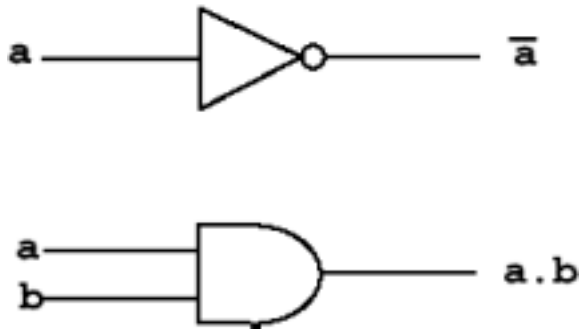
(d)

AB	*P*
0 0	0
0 1	1
1 0	1
1 1	0

(e)

AND **NAND** **OR** **NOR** **XOR**

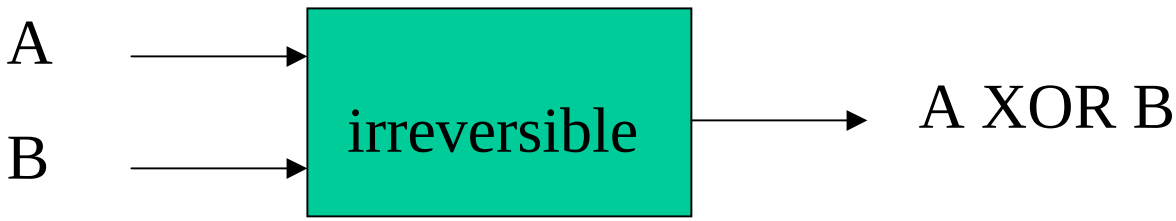
Logical irreversibility = physical irreversibility.



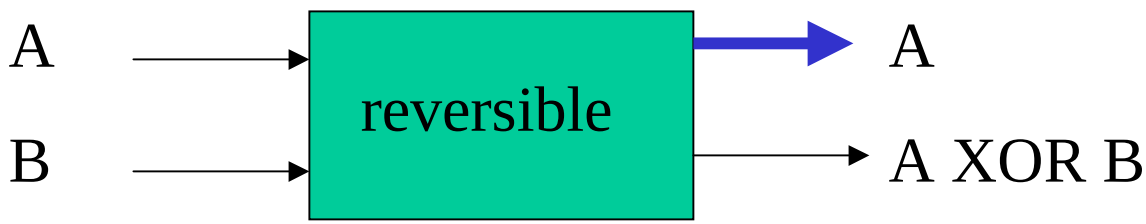
- The NOT gate is reversible
- The AND gate is irreversible
  - the AND gate erases information.
- the AND gate is physically irreversible.

Information is Physical

- Is a minimum amount of energy required per computation step?



- Rolf Landauer, 1970: Whenever we use a logically irreversible gate we dissipate energy into the environment.



A	B	X	Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Repeating input A on output makes this gate reversible

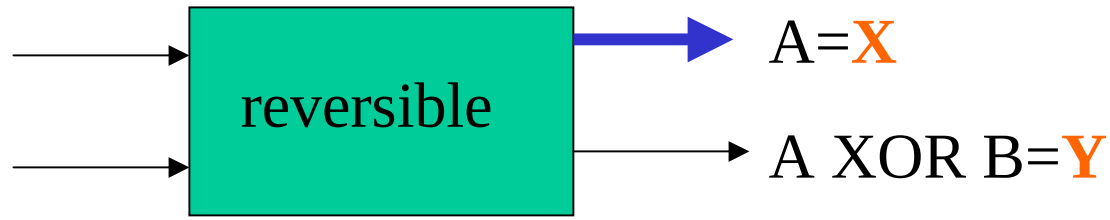
Overview of Reversible Gates

- A reversible gate is one where there is a one-to-one correspondence between the inputs and the outputs (i.e., if in the truth table of the gate there is a distinct output row for each distinct input row).
- In Boolean algebra, such a function is both *one-to-one* (or **injective**) and *onto* (or **surjective**).

A	B	X	Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

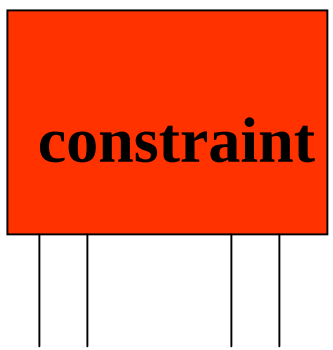
Think about a gate as an input/output *constraint*

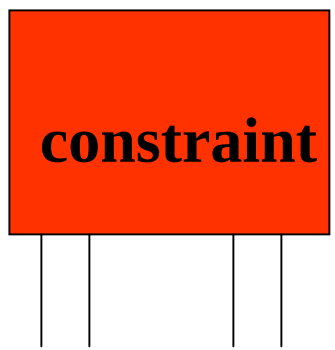
A	B	X	Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

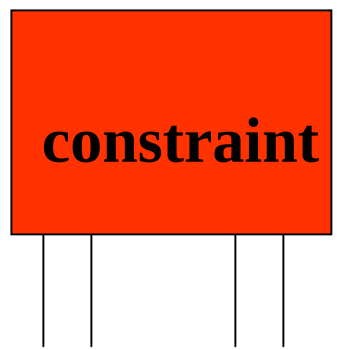


R(input, output) = R(<A,B>,<X,Y>)

R(<A,B>,<X,Y>) = Permute(2,3)







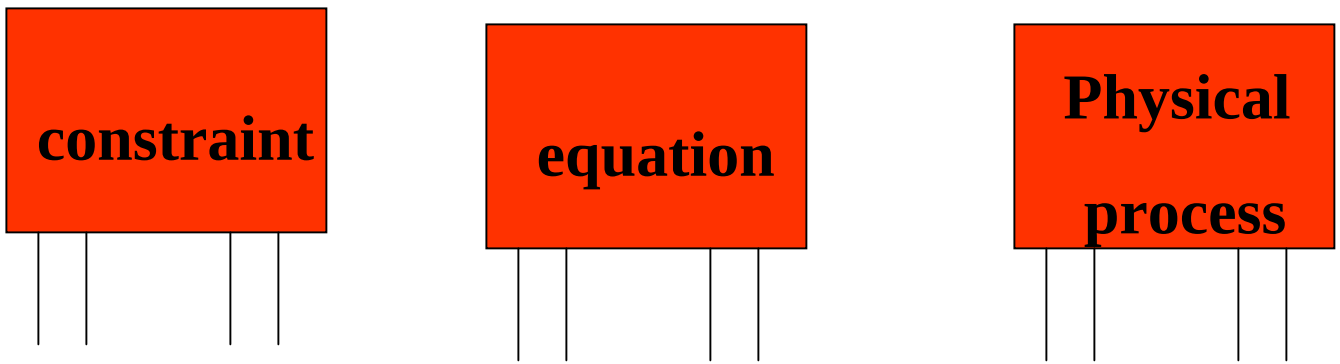
<0,0> ==> <0,0>

<1,0> <== <1,1>

<<1,0> , <1,1>> YES

<<1,1> , <0,1>> NO

Think about a gate as an input/output constraint



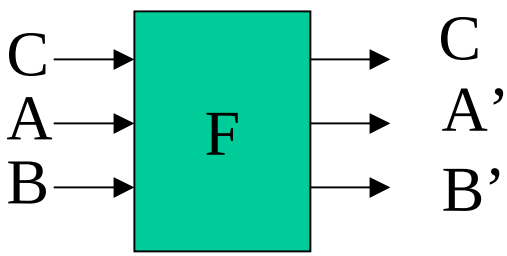
Few ways to think about reversible gates

People in the 1970s started wondering if it might be possible to have logically reversible gates that could be used to make a physically reversible computer.

Logically Reversible Gates

A logically reversible gate must have the same number of inputs as outputs (why?)! Two of the most famous such gates are the **Fredkin Gate** and the **Toffoli Gate**, shown below.

The Fredkin Gate is a *controlled swap gate*, i.e. if the control bit C is 1, then the two input bits A and B are swapped at the output. See the figure and the corresponding truth table.



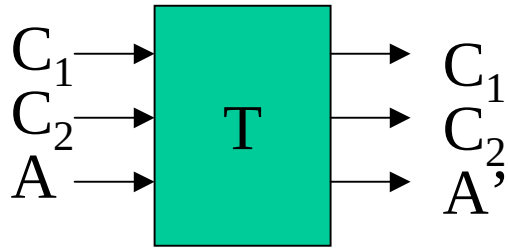
C A B	C A' B'
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 ***1 0***
1 1 0	1 ***0 1***
1 1 1	1 1 1

Truth table for the **Fredkin Gate**. If C=1, A and B swap, otherwise A and B go through unchanged.

The **Toffoli Gate** is often called a **CCNot Gate**, i.e. if the two control input bits are both 1, then the 3rd input is reversed. See the figure and its truth table on the next slide.

The Toffoli Gate is an important gate in quantum computing as you know already.

C_1 C_2 A	C_1 C_2 A’
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 0 1
1 1 0	1 1 ***1***
1 1 1	1 1 ***0***



If the two control input bits C_1 and C_2 of a **Toffoli Gate** are both 1, then the input bit A reverses, else all 3 bits go through unchanged.

Both gates are **computationally universal**, i.e. they can be used to generate AND, OR, NOT gates, which together form a **computationally universal set** of gates (i.e. they can be used to generate any Boolean function (i.e. a function that maps a set of M-bit input strings into a set of N-bit output strings).

Ex. Show that these two gates are computationally universal.

Information is Physical

- **Charles Bennett, 1973:**
  - There are no **unavoidable** energy consumption requirements per step in a computer.
  - Power dissipation of ***reversible*** computer, under ideal physical circumstances, is **zero**.

If reversible logic gates are computationally universal, then one can build computers based on them which should also be reversible, contradicting Landauer's original conclusion.

Bennet of IBM in 1973 discovered a way to make a reversible Turing Machine (by adding a history tape that gets written on and then unwritten (made blank) at the end.

Reversible computing implies no information is wiped out, hence a history of all calculations is kept, then is reversibly restored to its original state.

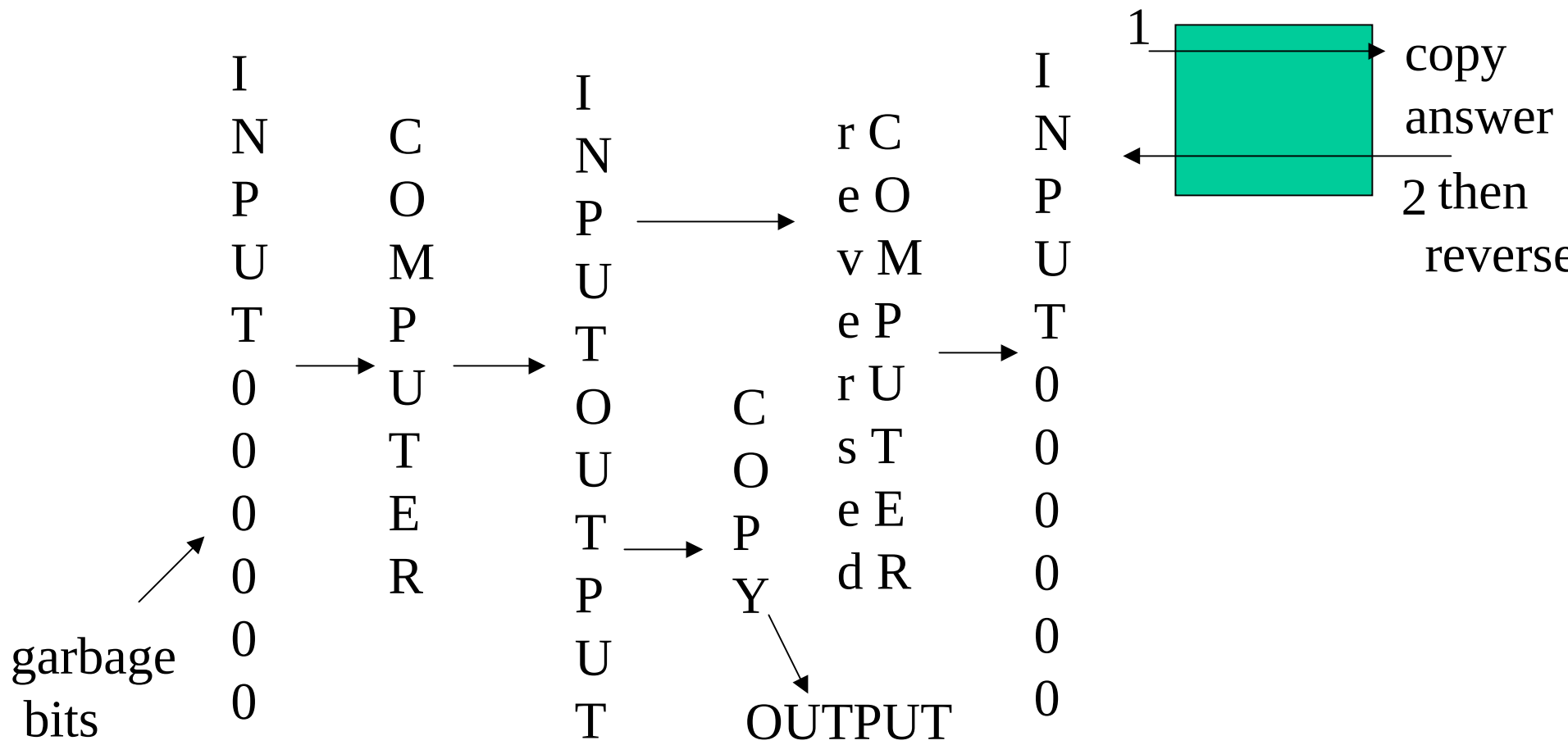
How to Compute Reversibly

1) Design a computer using reversible gates, to make it reversible.

2) Send in a bitstring on the LHS (see figure) to the reversible computer. The answer exits on the RHS.

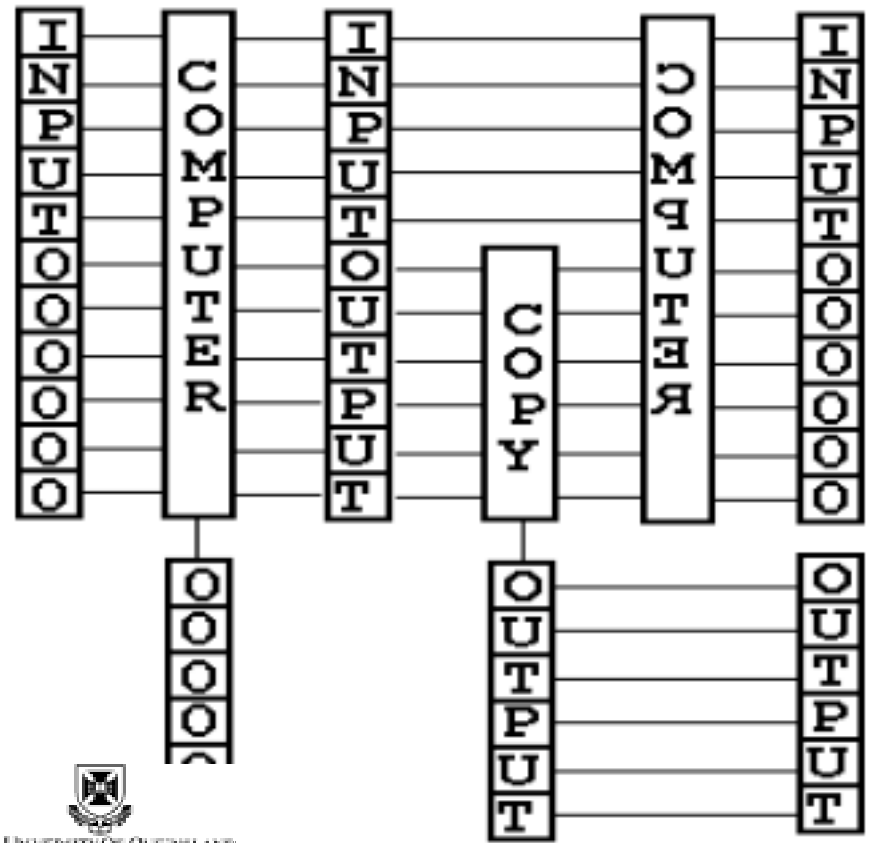
3. Make a copy of the answer, keep it.

4. Send the answer from the RHS back into the computer, resulting in the original input bit string on the LHS. (As would be expected in a reversible computer).



Reversible computation.

- **Charles Bennett, IBM, 1973.**
  - ***Logical reversibility of computation,***
    - **IBM J. Res. Dev. 17, 525 (1973).**



- This principle applies also to combinational circuits that we build.
- But is this a best way?
- No, this is extremely wasteful
- New principles of logic synthesis should be invented

Future of Reversible Computing

• Computer science has no choice.
• It must adapt itself to reversible computing, otherwise molecular scale circuits (a consequence of Moore's law over the next 20 years) will explode with the heat they generate.

• CS will have to use reversible gates, hence compute reversibly.
• This is not easy. All bits have to be stored and reversed. How to do this in practice?

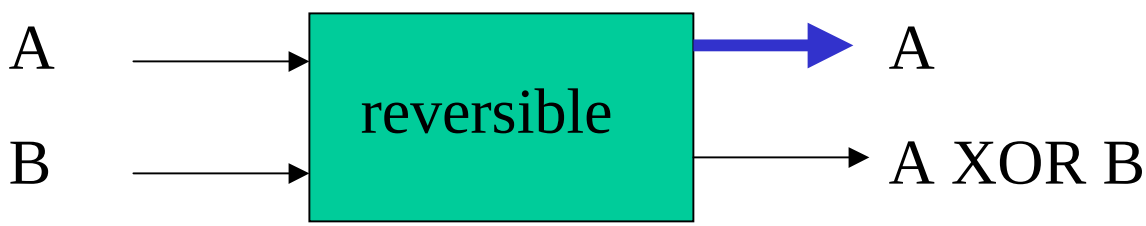
•Ongoing research, e.g. reversible high level computer language "R" at MIT.
• Laptop designers are interested in (quasi)-reversible circuits for laptops. Quasi-reversible circuitry generates less waste heat, hence enhances battery life.

Landauer Theorem

- Whenever we use a logically irreversible gate we dissipate energy into the environment.

- Reversible gate is a necessary but not sufficient condition of losing no power

- In addition, we need at least an ideal switch.
- Can ideal switch be build?



Real-world Applications

- Digital signal processing
- Cryptography
- Computer graphics
- Network congestion

Links to Quantum Computation

- Quantum operations are all reversible
- Every (classical) reversible circuit may be implemented in quantum technology
- Certain quantum algorithms have “pseudo-classical” subroutines, which can be implemented in reversible logic circuits

Theoretical Advantages

- Information, like energy, is conserved under the laws of physics
- Thermodynamics can be used to tie the irreversibility of a system to the amount of heat it dissipates
- An energy-lossless circuit must therefore be information-lossless
- Furthermore, there is evidence to suggest that reversible circuits may be built in an energy-lossless way

Landauer's principle

- ***Landauer's principle:*** **logic computations that are not reversible, necessarily generate heat:**
  - **i.e. *kT*log(2), for every bit of information that is lost.**

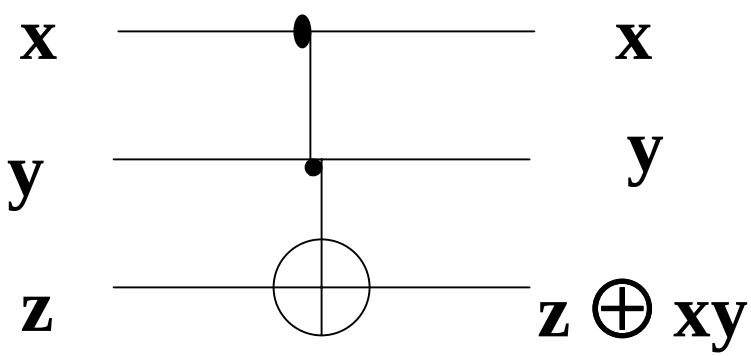
    where ***k*** is Boltzmann's constant and ***T*** the temperature.
- For *T* equal room temperature, this package of heat is small, i.e. 2.9 x 10^{-21} joule, but non-negligible.
- In order to produce zero heat, a computer is only allowed to perform ***reversible computations.***
- Such a logically reversible computation can be `undone': the value of the output suffices to recover what the value of the input `has been'.
- The hardware of a reversible computer **cannot** be constructed from the conventional gates
- On the contrary, it consists ***exclusively of logically reversible*** building blocks.

First Example: Toffoli's Gate

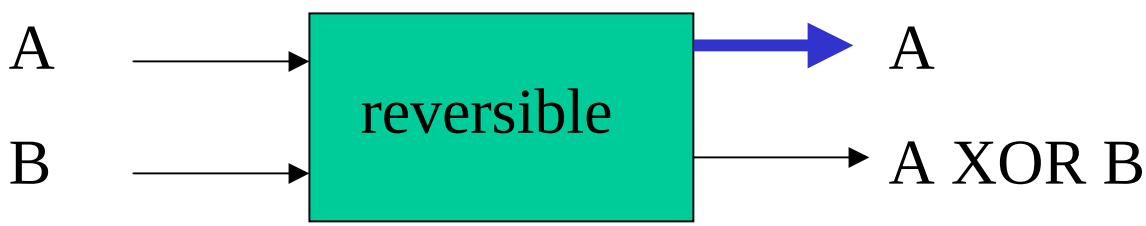
- Tomasso Toffoli, 1980: There exists a reversible gate which could play a role of a universal gate for reversible circuits.

$$Q^{(3)} :(x,y,z)==>(x,y,z \oplus xy)$$

⊕ denotes EXOR



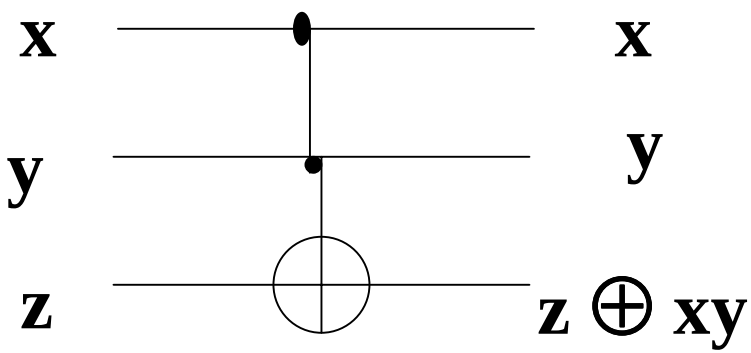
Fredkin and Toffoli created the first (3,3) universal gate



Toffoli Gate

Non-linear Gate: Toffoli

- **Important example of non-linear gate:**
- **3-bit Toffoli gate, (3,3) Toffoli gate**
- **Called also controlled-controlled-NOT**



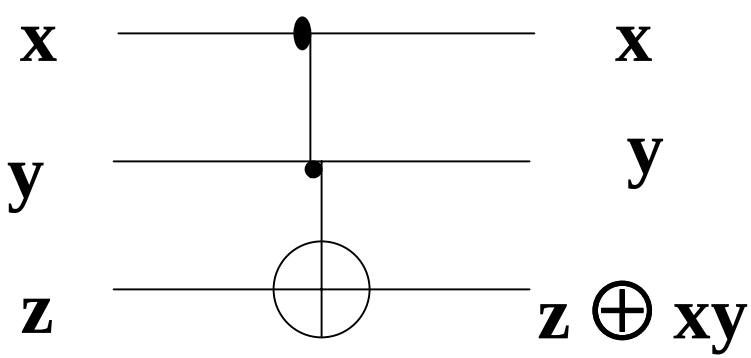
We already introduced this gate earlier, now will be more detail

It flips the third bit if the first two bits are 1 and does nothing else

Like the Toffoli gate, it is its own inverse (please prove these two facts)

- Of course, a computer built of only this type of gate, even using quantum technology, would have the power of only classical computers

But we can in addition to these gates construct gates that are possible only in quantum



Sufficient motivation is also that reversible gates can be realized in many other technologies that already exist

What other Reversible Gates can exist?

Width of Reversible Gate

- The number of output bits of a reversible logic gate necessarily *equals its number of input bits.*
- We will call this number the ***`logic width'*** of the gate.

Quantum Reversible Gates

- **In designing gates for a quantum computer, certain constraints must be satisfied.**
- **In particular, the *matrix of transition amplitudes* must be unitary, which implies, roughly speaking, that it conserves probability:**
- **The sum of the probabilities of all possible outcomes must be exactly 1.**
- **A consequence of this requirement is that any quantum computing operation must be reversible:**

Quantum Reversible Gates

- **quantum computing operation must be reversible:**
  - **You must be able to take the results of an operation and put them back through the machine in the opposite direction to recover the original inputs.**
- **Standard gates do not obey this rule, since information is irretrievably lost when two input bits are condensed into a single output bit.**

Universal Classical and Quantum Reversible Gates

- The study of reversible computing has gotten a lot of attention lately.
- This is because of quantum computing but also because it was found that reversible computation can be done also in CMOS, optically and in nano-technologies.
- It consumes very little power, and power reduction is becoming the most important design objective of computers.
  - a reversible computer can perform any computation and can do so with arbitrarily low energy consumption (Charles H. Bennett and Rolf Landauer of IBM)

Universal Classical and Quantum *Reversible Gates*

- A reversible (3,3) gate devised by Tommaso Toffoli of MIT is a "universal" classical gate:
- A computer could be built out of copies of this gate alone
- **Deutsch has shown that a similar gate is universal for quantum computers**
- Both the Toffoli and the Deutsch gates have three inputs and three outputs, but more recently two-qubit (2,2) gates have also proved universal for **quantum computations**
- **I believe that (2,2) gates can be constructed in MVL and I am working on this.**

Quantum Dots

- Practical ***quantum technologies*** are years or decades away.
- Few implementation schemes are already under discussion.
- The idea closest to existing electronic technology relies on
- ”Quantum dots” are closest to existing electronic technologies
- We will discuss them in one of next lectures
  - They are isolated conductive regions within a semiconductor substrate.
  - Each quantum dot can hold a ***single electron***, whose presence or absence represents one qubit of information. (qubit is quantum bit, it will be much more on it).

Another Quantum Circuits

- ***Hypothetical polymeric molecule*** in which the individual subunits could be toggled between the ground state and an excited state
- **David P. DiVincenzo** of IBM has described a mechanism by which isolated nuclear spins would interact -- and thereby compute -- when they are brought together by the *meshing of microscopic gears*
- ***Others*** will be also discussed

Reversibility in Logic Gates

- A logic gate is reversible if
  - It has as many input as output wires
  - It permutes the set of input values
- Some examples include
  - An inverter (the NOT gate)
  - A two-input, two-output gate which swaps the values on the input wires (the SWAP gate)
  - An (n+1)-input, (n+1)-output gate which leaves the first n wires unchanged, and flips the last if the first n were all 1 (the n-CNOT gate)

Reversibility in Logic Circuits

- A combinational logic circuit is reversible if
  - It contains only reversible gates
  - It has no FANOUT
  - It is acyclic (as a directed multigraph)
- It can be shown that a reversible circuit has as many input wires as output wires, and permutes the set of input values

- **Reversible Logic**

reversible gate

invertible function

Terminologies are not yet consistent, be aware

Linear Reversible Gates

Second Example: Feynman Gate

Quantum XOR

- Controlled NOT
- Quantum XOR
- Reversible XOR
- QCF
- Feynman gate

Quantum XOR

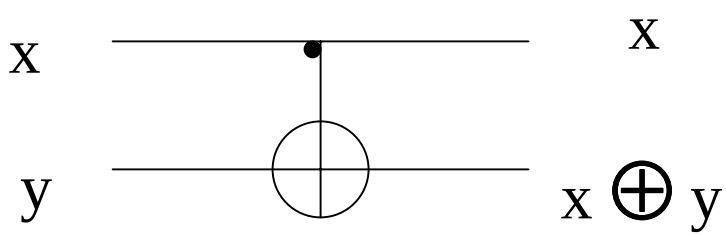
- How do we build up a complicated reversible computation from elementary components -
  - what constitutes a universal gate set?
- We will see that **one-bit and two-bit reversible gates do not suffice**; we will need three-bit gates for universal reversible computation.
- Of the ***four 1-bit --> 1-bit gates***, two are reversible; the trivial gate and the NOT gate.
- Of the $(2^4)^2$ = **256 possible 2-bit--> 2-bit gates**, 4! = 24 are reversible.

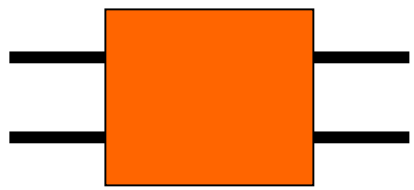
Quantum **XOR**

- Of special interest is the controlled-NOT or **reversible XOR** gate:

$$XOR : (x; y) \Longrightarrow (x, x \oplus y);$$

by $\oplus$ we denote EXOR (modulo-2 sum)

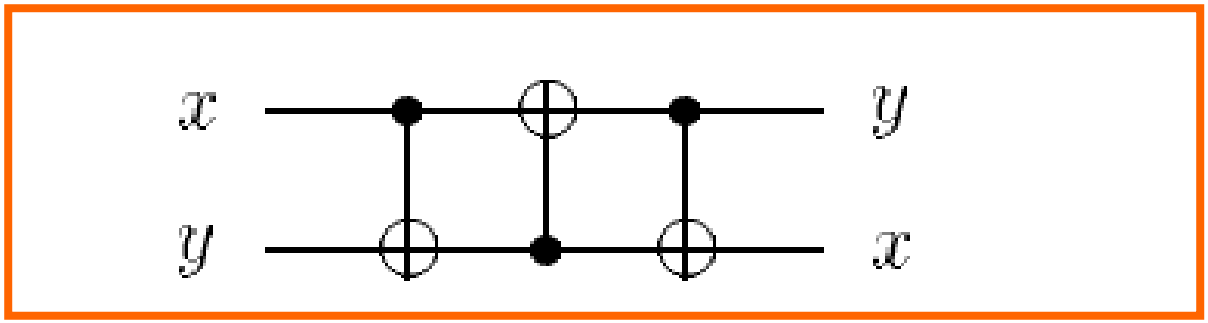




These notations were introduced by physicists and they are inconsistent with standard electrical engineering notations, however it will be convenient for us to use both notations.

Swapping bits using XOR cascade

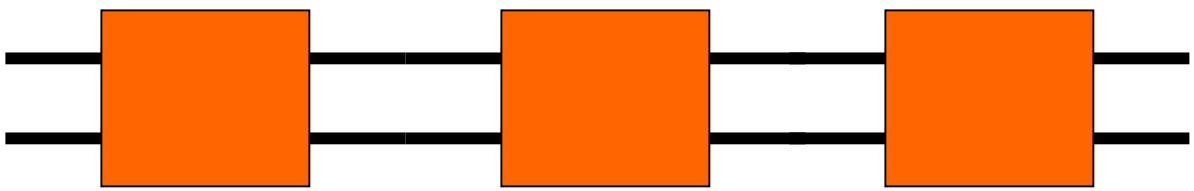
With the circuit



Constructed from three Quantum XORs, we can swap two bits:

(x,y) ==> (x,x ⊕ y) ==> (y,x ⊕ y) ==> (y,x)

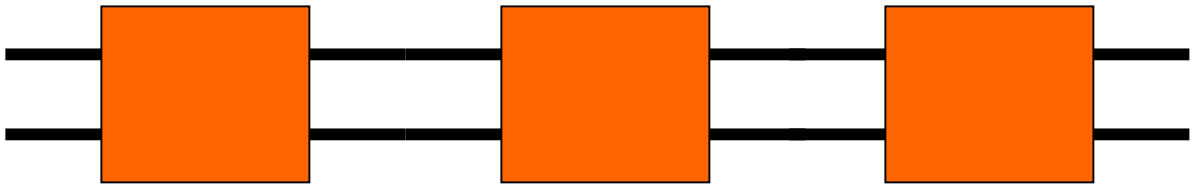
Conclusion: in quantum logic you pay for crossing wires!!!



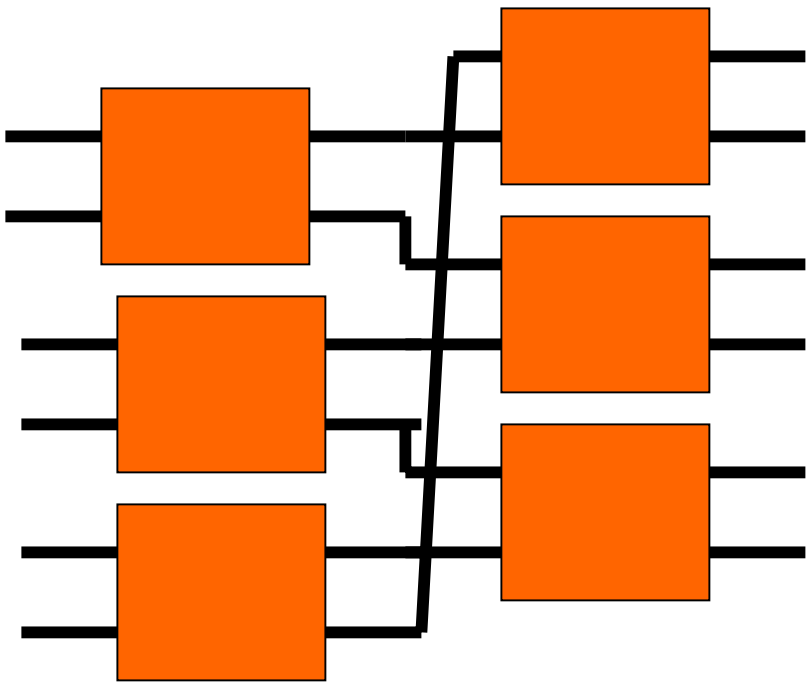
cascade

Of importance in quantum, quantum dot, but not CMOS

Structures



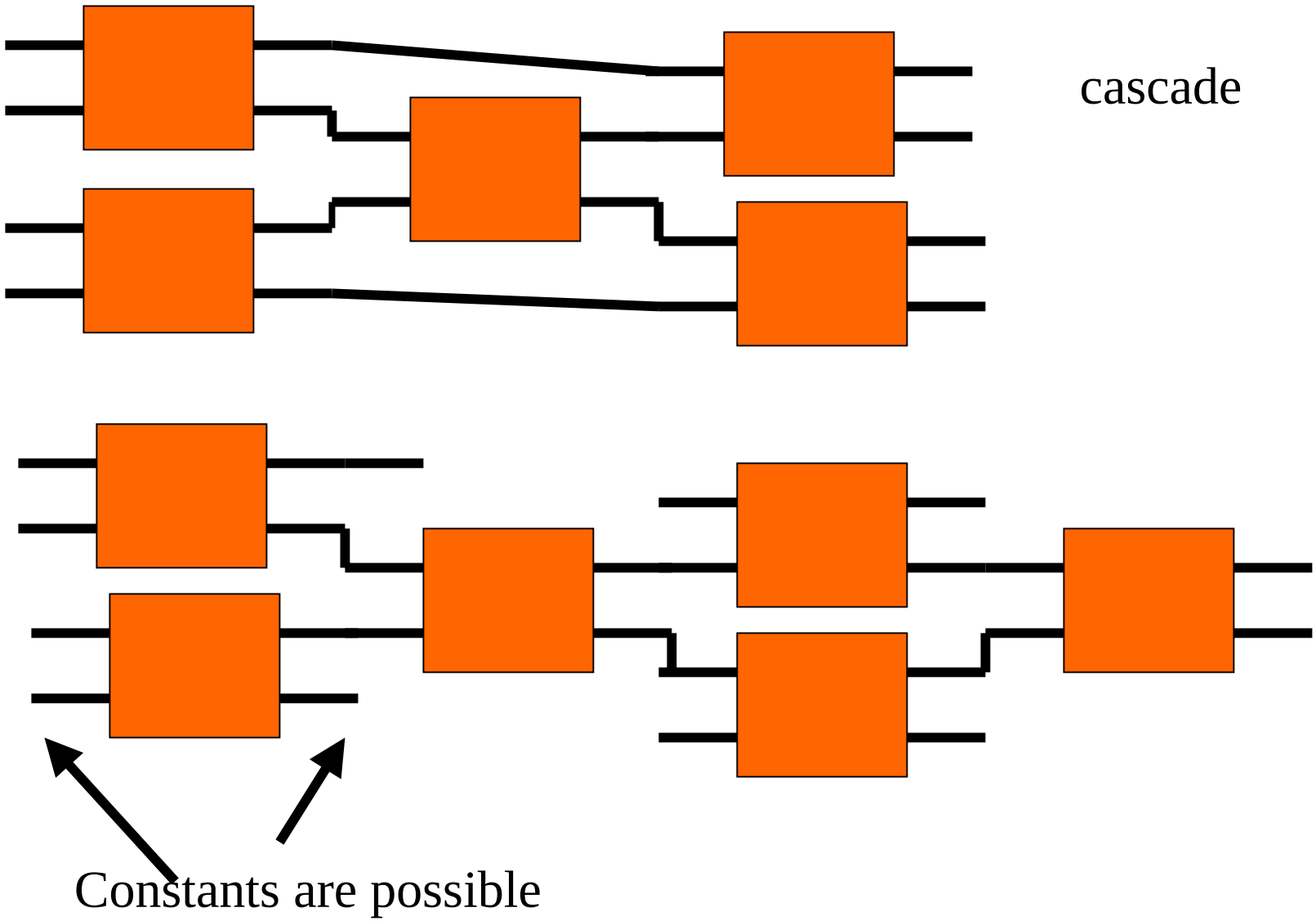
cascade



Only linear circuits with linear gates

No branchings!!

Structures



Rules for creating Structures

- **Do not generate the waste of outputs**
- **Create constants rather than functions on outputs**
- **Create re-usable functions (repeating inputs by branching may be useful)**

The QCF gate is not complete

- The QCF gate is not enough to build a complete quantum computer
  - like NOT gates are not enough to build a classical computer.
- Performing useful calculations requires gates that process more than one bit (or qubit in case of quantum logic) at a time.
- For example, AND gates in conventional computers.

Linear Gates

- To prove that one-bit and two-bit gates are not universal, it can be observed that they are linear.
- Composition of linear gates is always linear, and there exist of course non-linear functions (by linear we mean a circuit that satisfies superposition, as in control theory)
- Each reversible two-bit gate has an action of the form:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

Linear Gates

Where the constant $\begin{pmatrix} a \\ b \end{pmatrix}$

takes one of four possible values, and the matrix M is one of the six invertible matrices

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

All additions are performed in modulo-2 (EXOR)

Combining the six choices for M with the four possible constants,

we obtain 24 distinct gates, which exhausts all the reversible

2==> 2 gates

Linear Gates

- Since the linear transformations are closed under composition, any circuit composed from reversible 2 ==> 2 and 1 ==> 1 gates will compute a linear function **x ==> Mx+a**
- **As shown by Toffoli gate (controlled-controlled-Not , $Q^{(3)}$) there are (3,3) gates that are non-linear.**
- **We will investigate such gates for n>= 3**
- **Linear gates will still remain very important because of their special properties.**
- **EXOR forests used in BIST are linear circuits, they can be modified to reversible logic.**

Gates of logic width 3

- ***Fredkin*** and ***Toffoli***:
  - demonstrated that three inputs and three outputs is necessary and sufficient in order to construct a *reversible implementation* of an arbitrary boolean function of a finite number of logic variables.
- Thus, from the fundamental point of view, **reversible logic gates with a width equal to three have a privileged position**.

How many reversible gates exists?

- The truth table of a logic gate of width ***w*** consists of **2^w** lines, each containing two ***w*-bit** numbers: the *w*-bit input (*A*, *B*, *C*, ...) and the *w*-bit output (*P*, *Q*, *R*, ...).
  - For convenience, all possible inputs, ranging from (0, 0, 0, ...) to (1, 1, 1, ...), are ordered arithmetically.
- Such a gate is reversible if-and-only-if all **2^w** output numbers **form a permutation of the 2^w input numbers.**
- Hence, there exist **exactly $(2^w)!$ different reversible gates of width *w***.

How many reversible (3,3) gates exists?

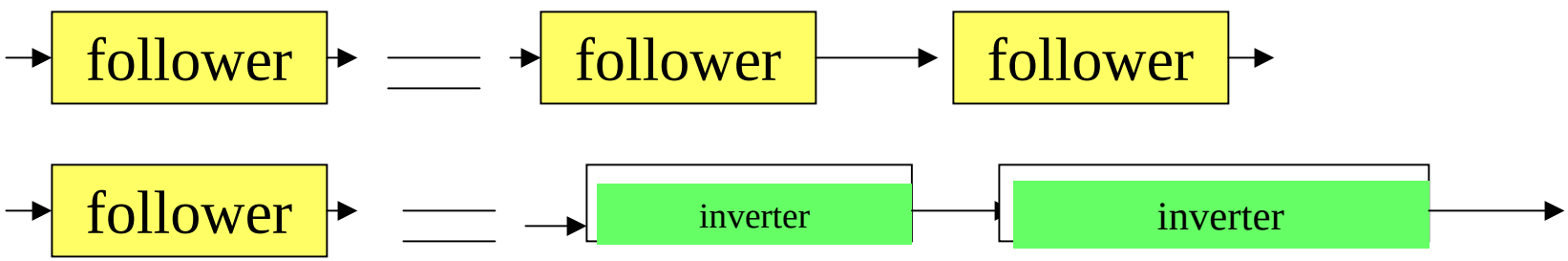
- In particular, there are **8! = 40,320** reversible gates with ***3-bit width***.
- We will investigate which of these 40,320 gates fulfil the role of **universal building block**, and which fulfil this job **more efficiently** than the others.
- In order to tackle the problem, we will successively study the reversible gates with ***w* = 1, *w* = 2, and *w* = 3.**

Single bit Reversible Gates

- There exist only four different truth tables with one bit input and one bit output.
- Two of them are logically irreversible: the **resetter** ($P = 0$) and the **setter** ($P = 1$).
- The two others are reversible: the follower ($P = A$) and the inverter ($P =$ NOT A).
  - If, for example, we have `forgotten' the value of A, knowledge of the value of the inverter's output P suffices to recover it.

Single bit Reversible Gates

- Note that among the 1-bit reversible gates, the NOT gate is a `generator'.
  - This means we can make any reversible gate of width 1 by combining a finite number of this particular gate.
- Indeed, a follower can be fabricated by the sequence of two inverters.
- The opposite is not true: one cannot fabricate an inverter by cascading followers.



Reversible Gates with two bits

- There are $4^4 = 256$ different truth tables with two inputs (A, B) and two outputs (P, Q).
- Among them, only $4! = 24$ are reversible.
- However, some of these twenty-four truth tables fall apart into two separate 1-bit reversible tables.
  - **Table a** decomposes into one follower $Q = A$ (Table 2b) and one inverter P = NOT B **(Table c)**.

AB	PQ
0 0	1 0
0 1	0 0
1 0	1 1
1 1	0 1

(a)

=

A	Q
0	0
1	1

(b)

&

B	P
0	1
1	0

(c)

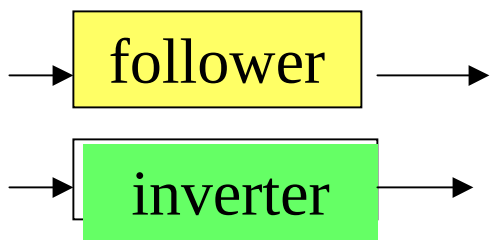


Table : Falling apart of a truth table.

Calculation with two bits

- On the contrary, truth Table GATES b is an example of a 2-bit reversible table that cannot be reduced to two separate 1-bit reversible tables, and therefore is called a true two-bit reversible gate.
- Among the 24 reversible 2-bit tables, only 16 are true 2-bit tables.

P=A

Q=A ⊕ B

A	*P*
0	1
1	0

(a)

AB	*PQ*
0 0	0 0
0 1	0 1
1 0	1 1
1 1	1 0

(b)

ABC	*PQR*
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 0 1
1 1 0	1 1 1
1 1 1	1 1 0

(c)

Table GATES: Feynman's truth tables: (a) NOT, (b) CONTROLLED NOT, (c) CONTROLLED CONTROLLED NOT.

CONTROLLED NOT by Feynman

- **All reversible true 2-bit gates can be fabricated from the same building block, combined with an inverter before and/or an inverter after.**
- **Indeed, Table 3b together with the inverter (Table 3a) forms a set of two building blocks with which we can synthetize an arbitrary reversible 2-bit gate.**
- **Truth Table 3b is called the `CONTROLLED NOT` by Feynman**

- **Its logic operation looks like this:**

$$P = A$$

$$Q = A \oplus B ,$$

where XOR is the abbreviation of the EXCLUSIVE OR function.

- **The gate is the reversible form of the classical (irreversible) XOR gate.**

A	P
0	1
1	0

(a)

AB	PQ
0 0	0 0
0 1	0 1
1 0	1 1
1 1	1 0

(b)

ABC	PQR
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 0 1
1 1 0	1 1 1
1 1 1	1 1 0

(c)

Table GATES: Feynman's truth tables: (a) `NOT`, (b) `CONTROLLED NOT`, (c) `CONTROLLED CONTROLLED NOT`.

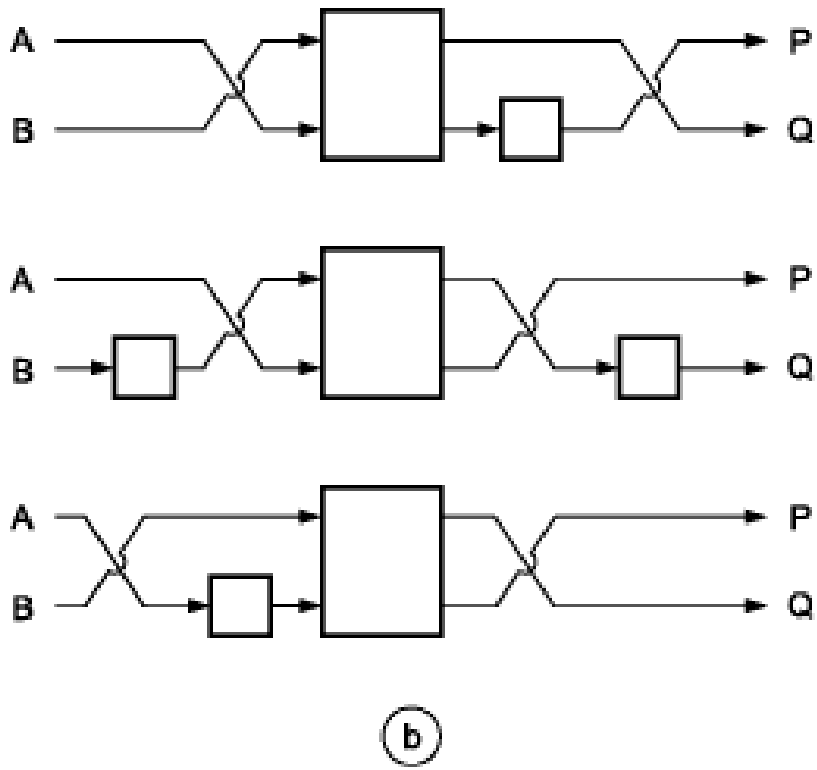
The synthesis of a 2-bit reversible gate: (a) truth table, (b) three different implementations combining NOTs and CONTROLLED NOT.

- Figure gives a representative example of a 2-bit reversible gate, realized by combining NOT and CONTROLLED NOT gates.

- Whereas output *Q* simply equals input *B*, output *P* can be described in three different ways:
- ***P* = NOT (*A* ⊕ *B*)**
- ***P* = *A* ⊕ (NOT *B*)**
- ***P* = (NOT *A*) ⊕ *B* .**

AB	*PQ*
0 0	1 0
0 1	0 1
1 0	0 0
1 1	1 1

(a)



(b)

Controlled NOT gate

- These three boolean expressions are identical, but lead to different physical realizations.
- We note, however, that these implementations not only make use of the 1-bit NOT function and the 2-bit CONTROLLED NOT function, but also of the 2-bit exchanger, i.e. the gate that interchanges two logic variables (realizing $P = B$ as well as $Q = A$).
- This is an example of a general property: the EXCHANGER, the NOT, and the CONTROLLED NOT form a natural `generating set' for the twenty-four 2-bit reversible gates.
- More precisely, each reversible 2-bit gate can be synthesized by taking one or zero CONTROLLED NOTs and adding one or zero EXCHANGERs and one or zero NOTs to the left and to the right of it.

- Note that:
  - neither the EXCHANGER nor the NOT is a true 2-bit gate,
  - but the CONTROLLED NOT is one.
- See Table 4

AB	PQ
0 0	0 0
0 1	1 0
1 0	0 1
1 1	1 1

(a)

AB	PQ
0 0	0 1
0 1	0 0
1 0	1 1
1 1	1 0

(b)

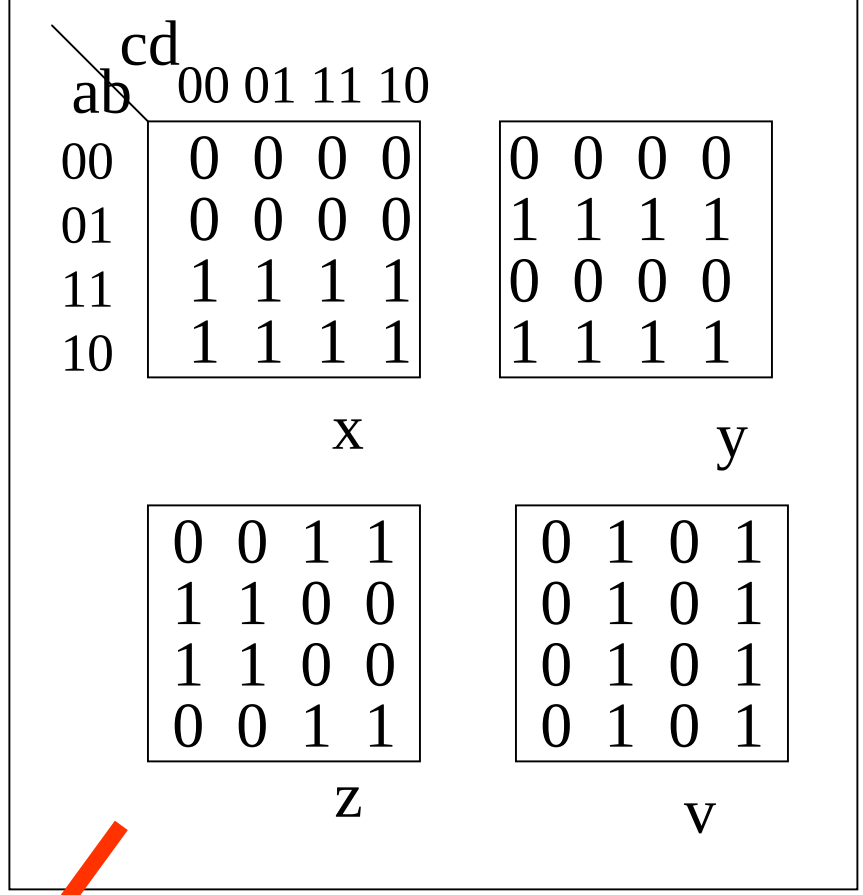
AB	PQ
0 0	0 0
0 1	0 1
1 0	1 1
1 1	1 0

(c)

Table 4: The three generators of the 2-bit reversible gates: (a) EXCHANGER, (b) NOT, (c) CONTROLLED NOT.

Converter of binary to Gray code

abcd	xyzv
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

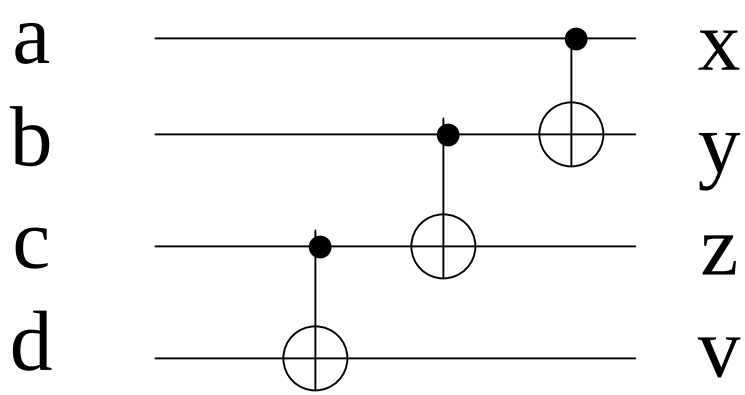


$x = a$

$y = a \oplus b$

$z = b \oplus c$

$v = c \oplus d$



Example from Literature

- Reversible half and full adders as given by De Vos (1999). We use the following gates:
  - The **CONTROLLED NOT** described by:

    P = A and Q=A $\oplus$ B

  - The **CONTROLLED CONTROLLED NOT** described by:

    P = A, Q = B, and R = (A AND B) $\oplus$ C

Implications of Reversibility: Additional Inputs

- Reversibility criteria may require us to add inputs to a given function. Take a 2-input OR function:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

A	B	C	F	G	H
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	1	0
1	1	0	1	1	1
+	+	+	+	+	+(?)

  - Note that we have some degrees of freedom in how we fill out the columns C, G, and H

Implications of Reversibility: Additional Inputs

- Intuitively, the need for extra inputs has to do with the number of original inputs as well as the degree of balance of the output function(s) (ratio of 0s and 1s).
- It would be good to derive the mathematical basis of this property.

Implications of Reversibility: Additional Inputs

- For example, a 2-input XOR can be made reversible by simply adding an output to the original gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

A	B	F	G
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Implications of Reversibility: Additional Inputs

- All n-input XOR functions can be made reversible by adding n-1 outputs.
- Linearly Independent gates such as Galois Addition or MODSUM have the same property
- Similarly, it seems that reversible versions of OR and AND gates always require us to add one input to the original list of inputs. But more must be done to create such reversible gates
- Negations of inputs or outputs simply permute the rows (so analysis for NAND gates is same as for AND gates)

Implications of Reversibility: Additional Inputs

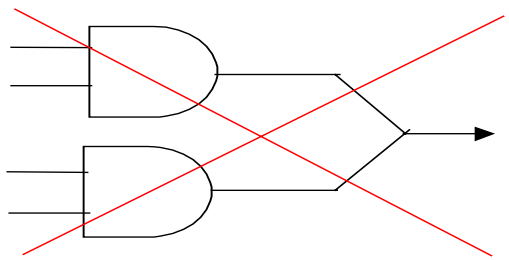
- Conclusion:
  - We may need to add a number of inputs and outputs to a given specification of a function to make it reversible.
  - I would like to be able to characterize the conditions under which this is needed and the nature of what we're adding.
  - This problem may already have been tackled elsewhere in Mathematics (how to turn a non-invertible function into an invertible one)

Implications of Reversibility: Additional Inputs

- **Completing** a given specification for a function with additional inputs and/or outputs may constitute the first step of our quest to synthesize an arbitrary function into a reversible circuit.

Implications of Reversibility: Fan-out operation

- In regular binary logic, we cannot "combine" two wires into one.



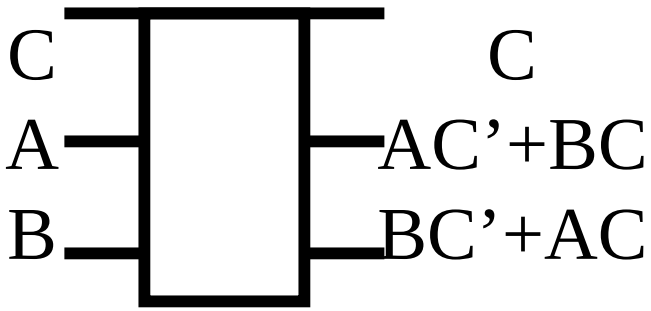
- Since we want our circuits to be reversible, any branching of a signal looked at in reverse will appear as combining signals.

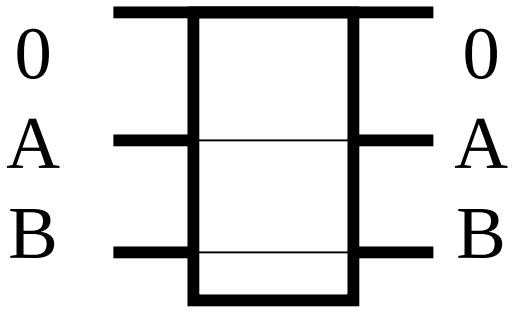
Implications of Reversibility: Fan-out operation

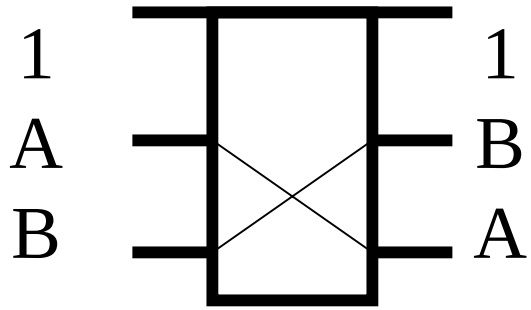
- Moreover, as Fredkin and Tofolli (1981) point out, duplication of signals from a physical viewpoint is far from trivial.
- We will therefore need to use gates anytime we wish to duplicate a signal.
- In minimizing a given function reversibly, we cannot assume free fan-out.

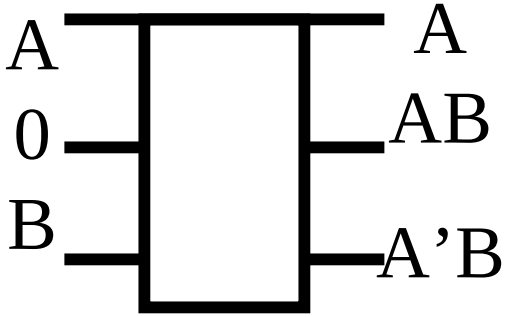
Third Example: Fredkin's Gate

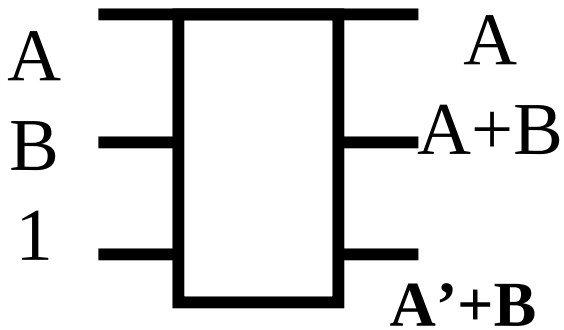
Operation of the Fredkin gate

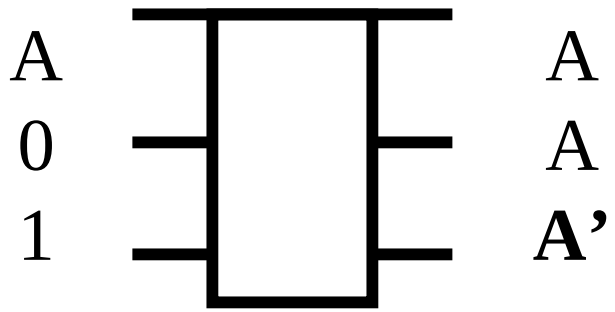








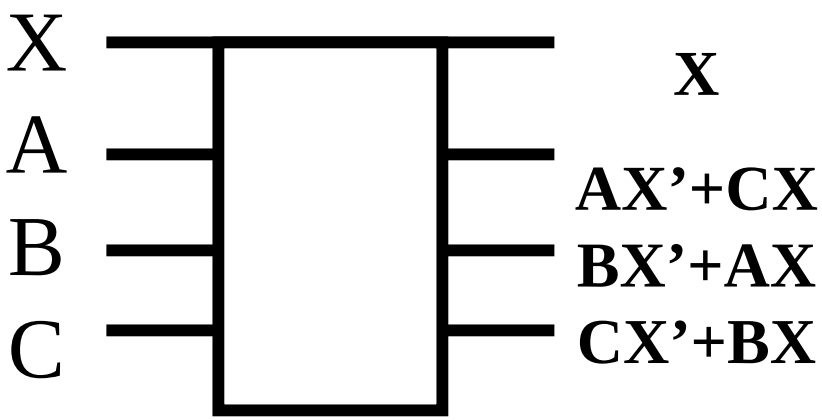


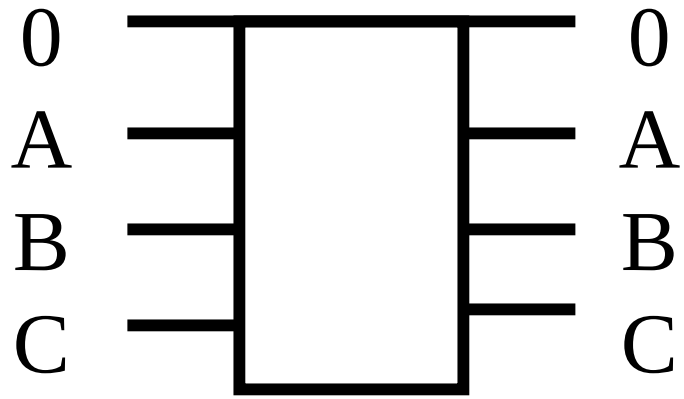


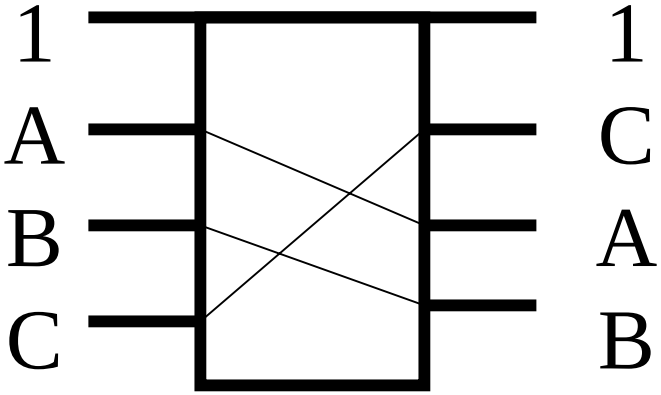
Note new notation for Fredkin gate.

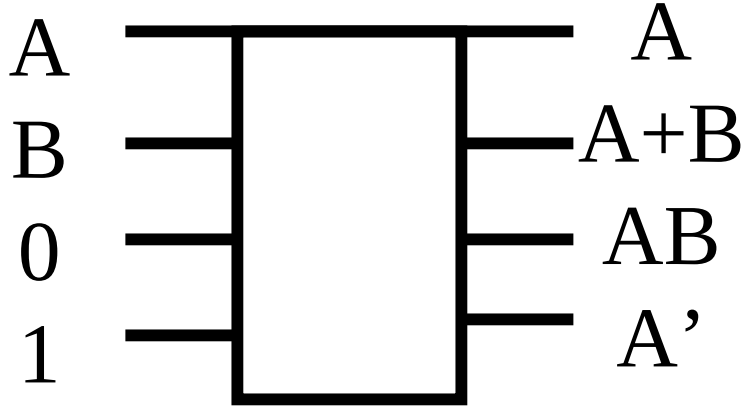
Note that it is a controlled swap gate.

A 4-input Fredkin gate





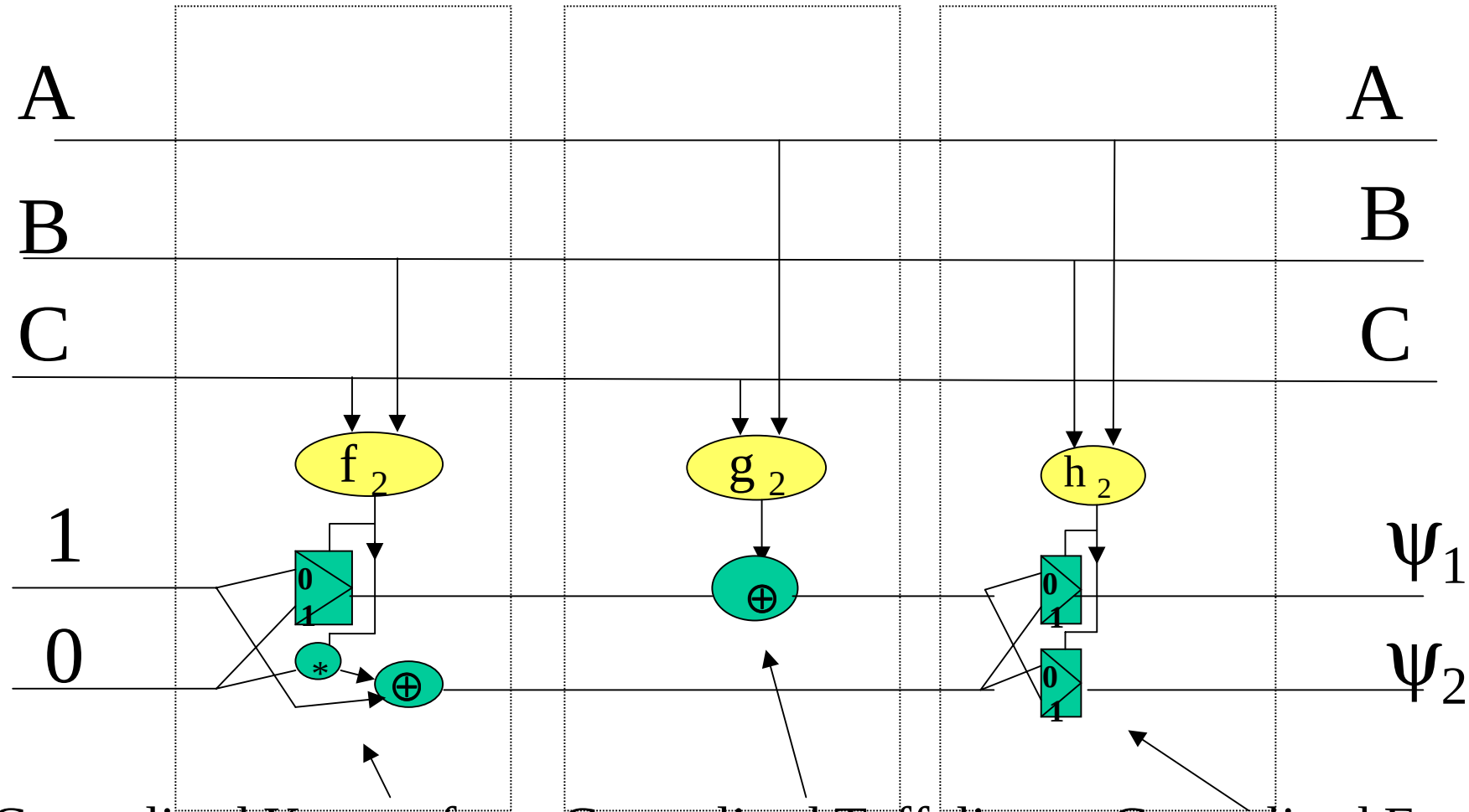




Concept: Cascades

Width of cascade. Length of cascade.
Garbage. Input constants.

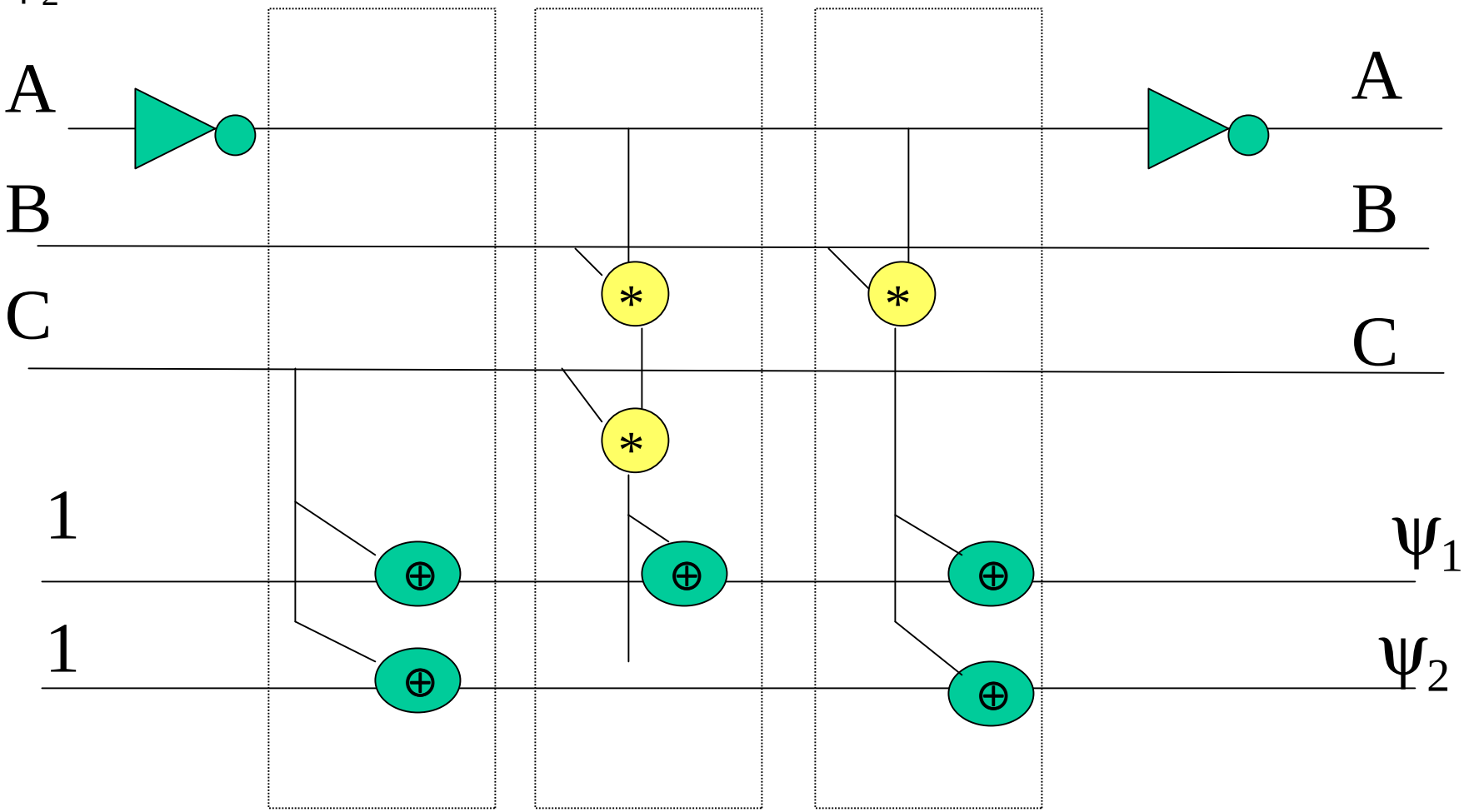
General Cascade of Kerntopf, Toffoli and Fredkin Family Gates



Example of multi-output FPRM cascade of Toffoli family gates

$\psi_1 = 1 \oplus C \oplus A'BC \oplus A' B$

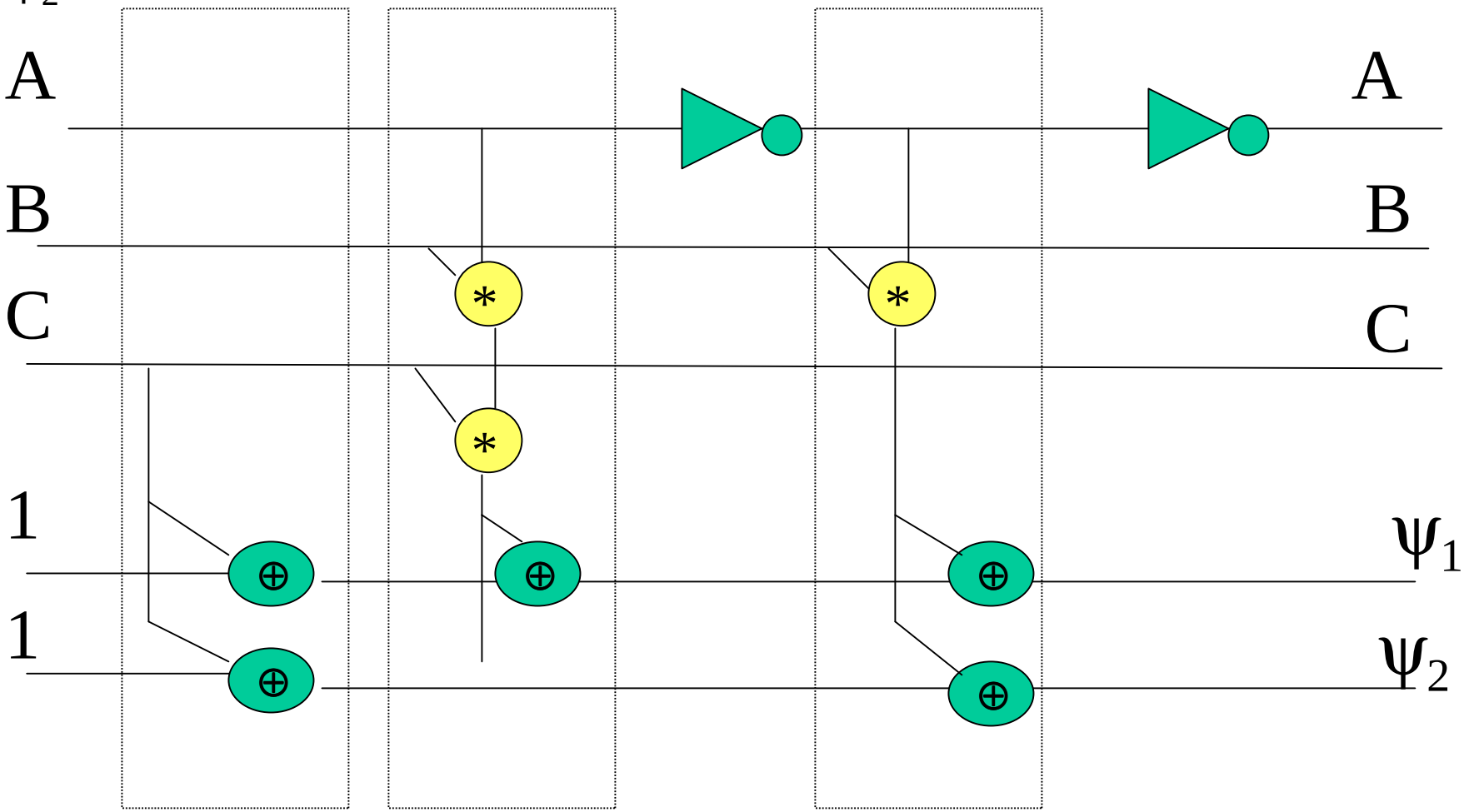
$\psi_2 = 1 \oplus C \oplus A' B$



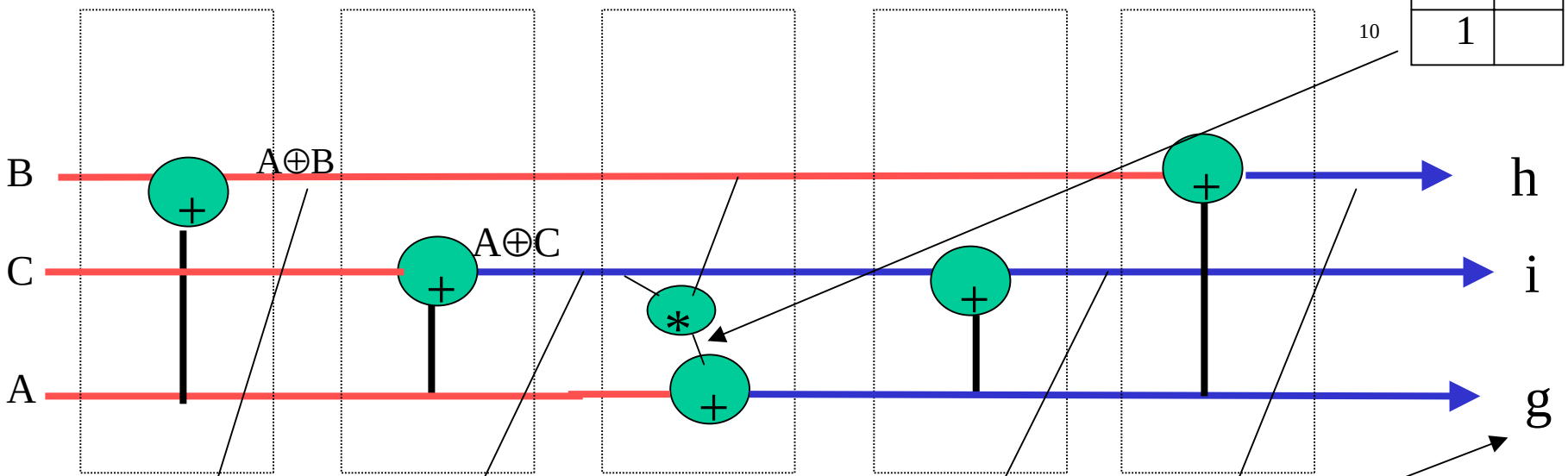
Example of multi-output ESOP cascade of Toffoli family gates

$\psi_1 = 1 \oplus C \oplus ABC \oplus A' B$

$\psi_2 = 1 \oplus C \oplus A' B$



Optimal Solution to Miller Function



AB \ C	0	1
00		
01		1
11		
10	1	

AC ⊕ BC ⊕AB

AB \ C	0	1
00		
01	1	1
11		
10	1	1

AB \ C	0	1
00		1
01		1
11	1	
10	1	

AB \ C	0	1
00		1
01		
11		1
10	1	1

AB \ C	0	1
00		
01	1	
11	1	1
10	1	

AB \ C	0	1
00		
01		1
11	1	1
10		1

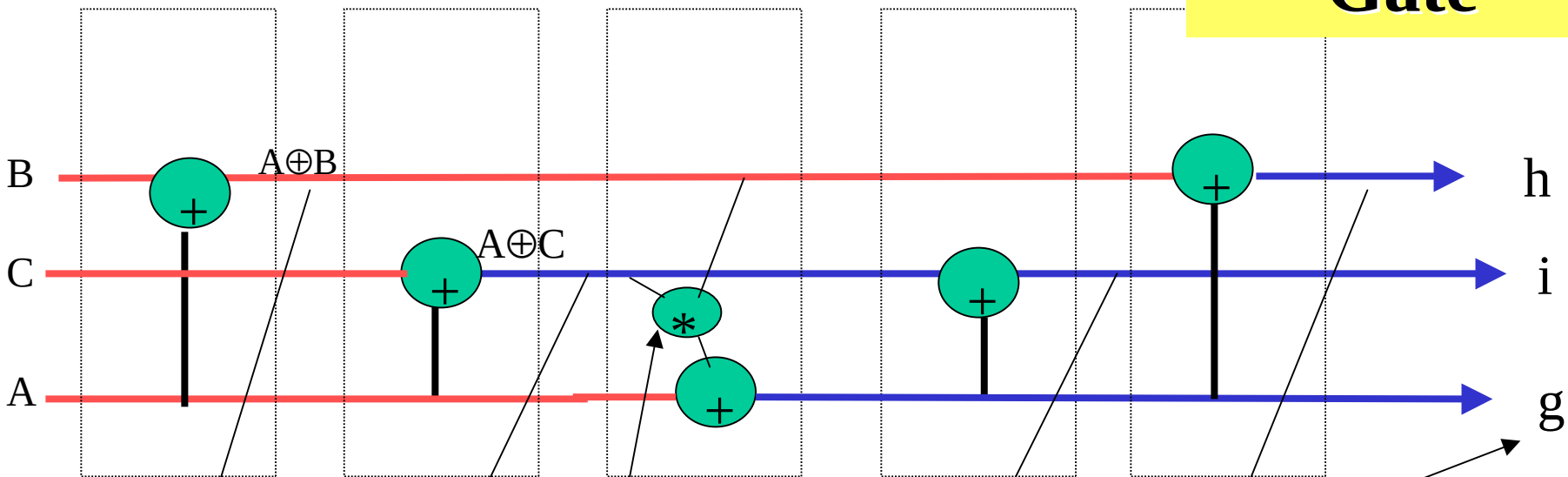
Equations for Miller Gate

$g = AC \oplus BC \oplus AB = \text{majority}(A,B,C)$

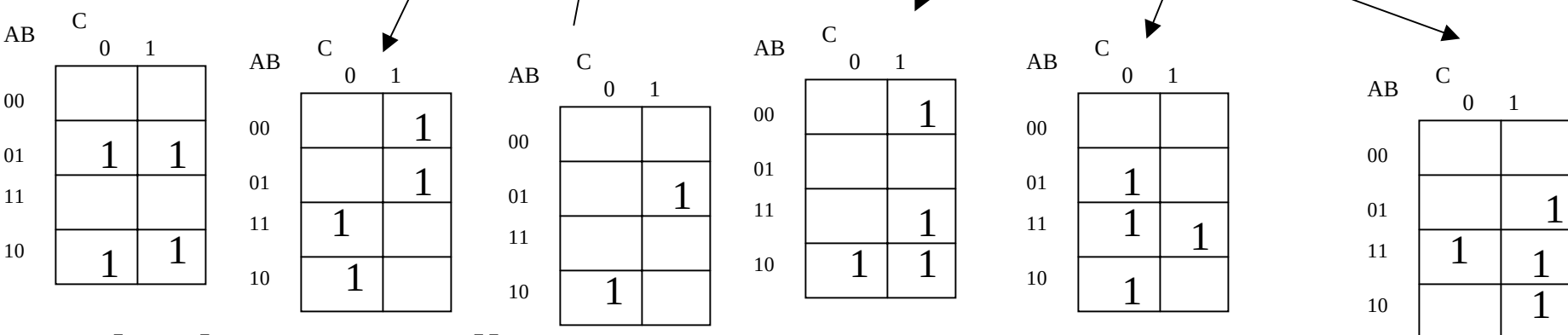
$i = AC \oplus B'C \oplus AB' = \text{majority}(A,B',C)$

$h = AC' \oplus BC' \oplus AB = \text{majority}(A,B,C')$

Exors in these equations can be replaced by ORs

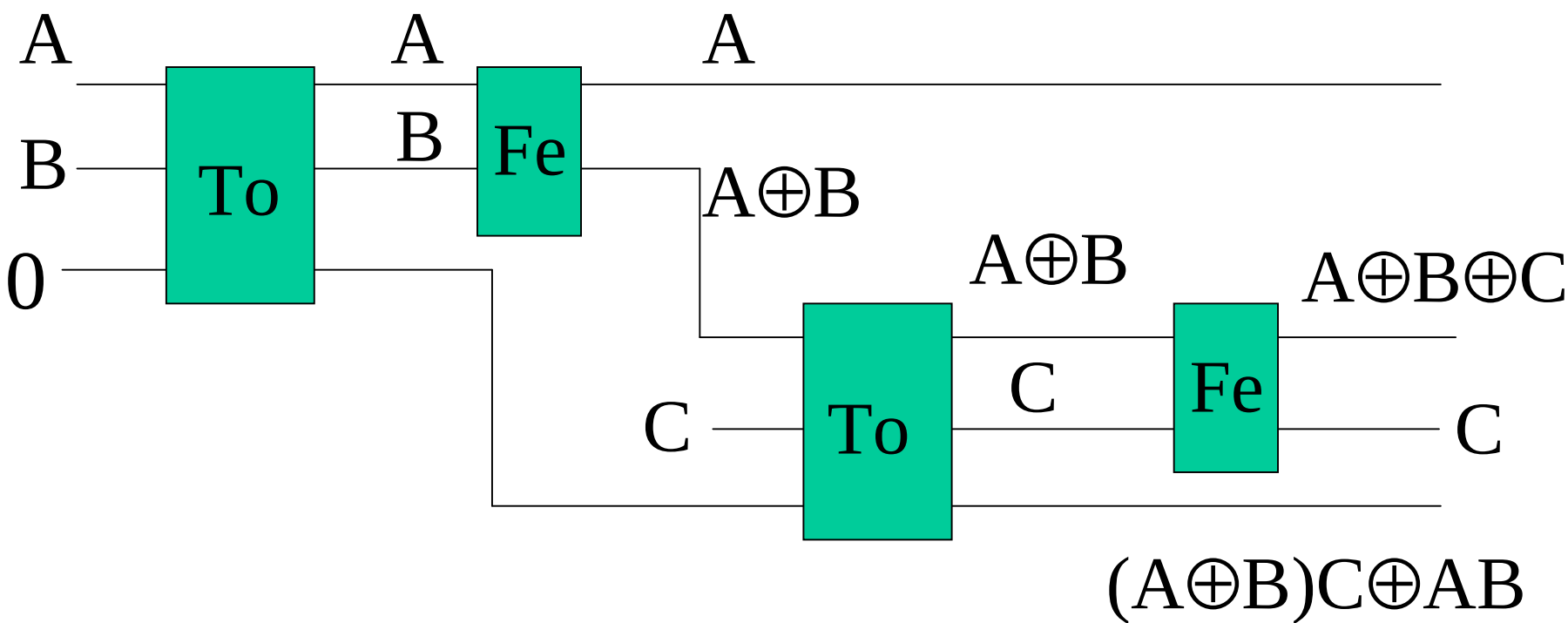


AC ⊕ BC ⊕AB



Optimal Solution to Miller Function

CMOS Notation for Full Adder realized using Composition

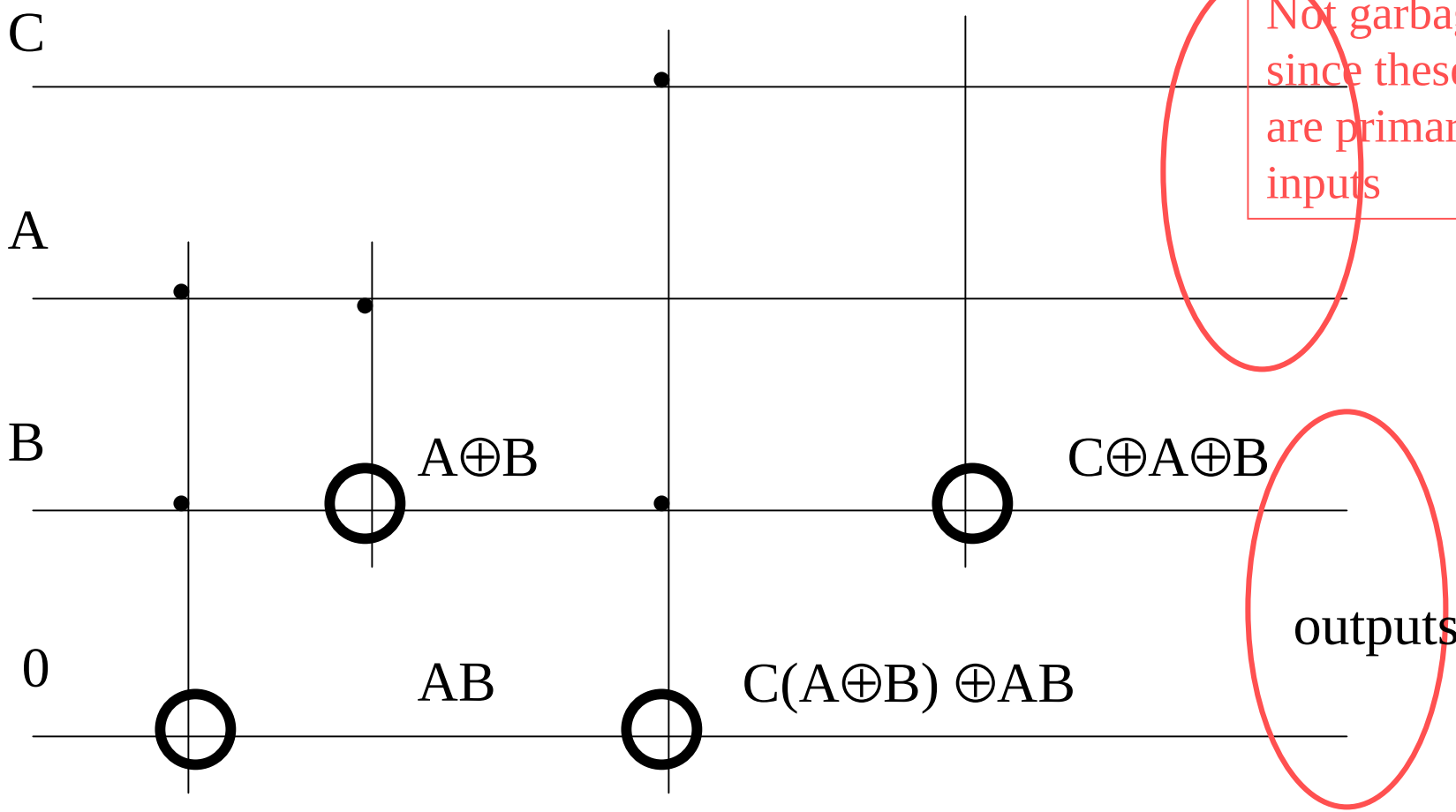


No garbage

one input constant

4 gates, optimum solution

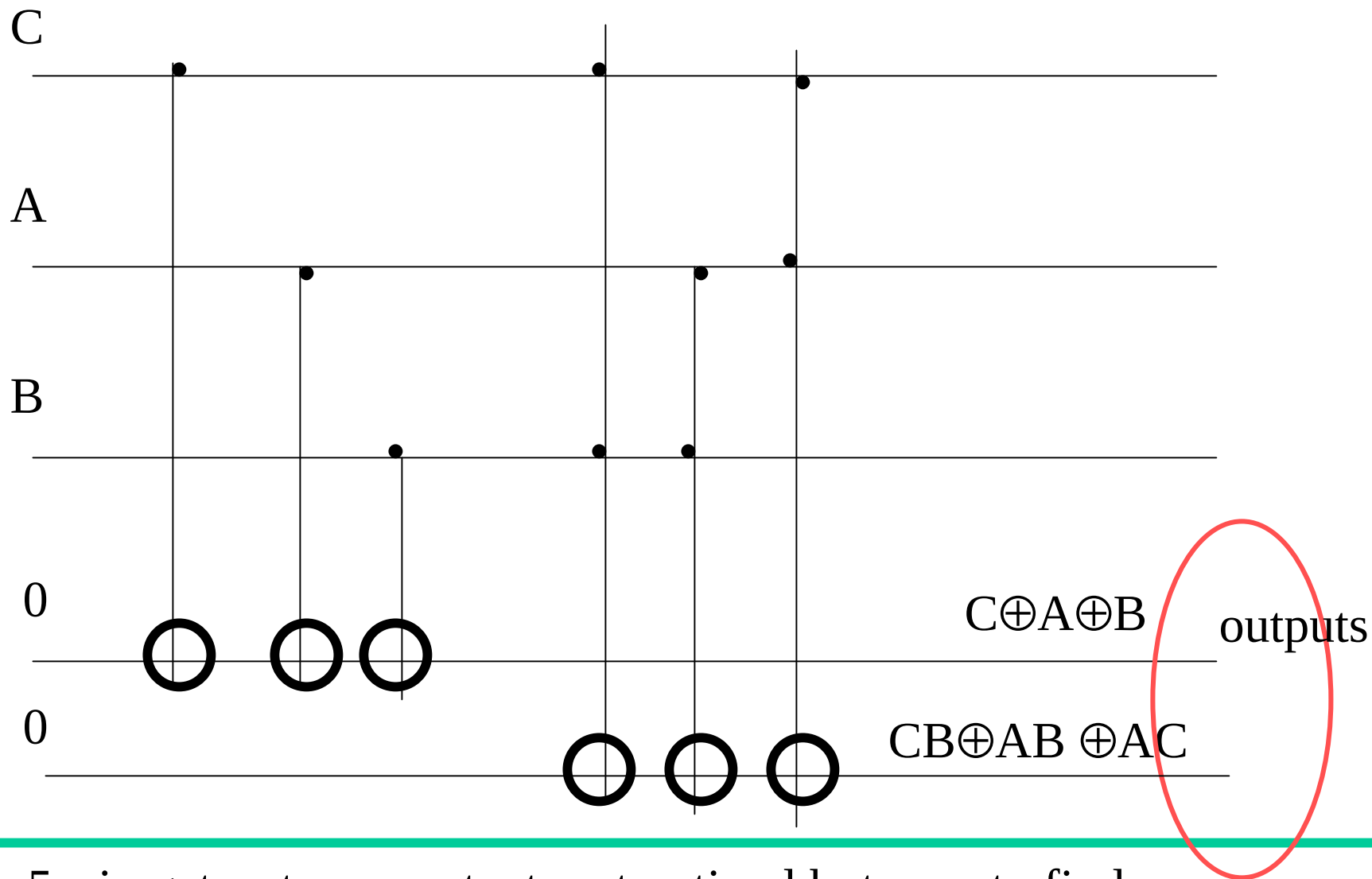
Quantum Notation for Full Adder realized using composition



Width 4

Optimum quantum solution?

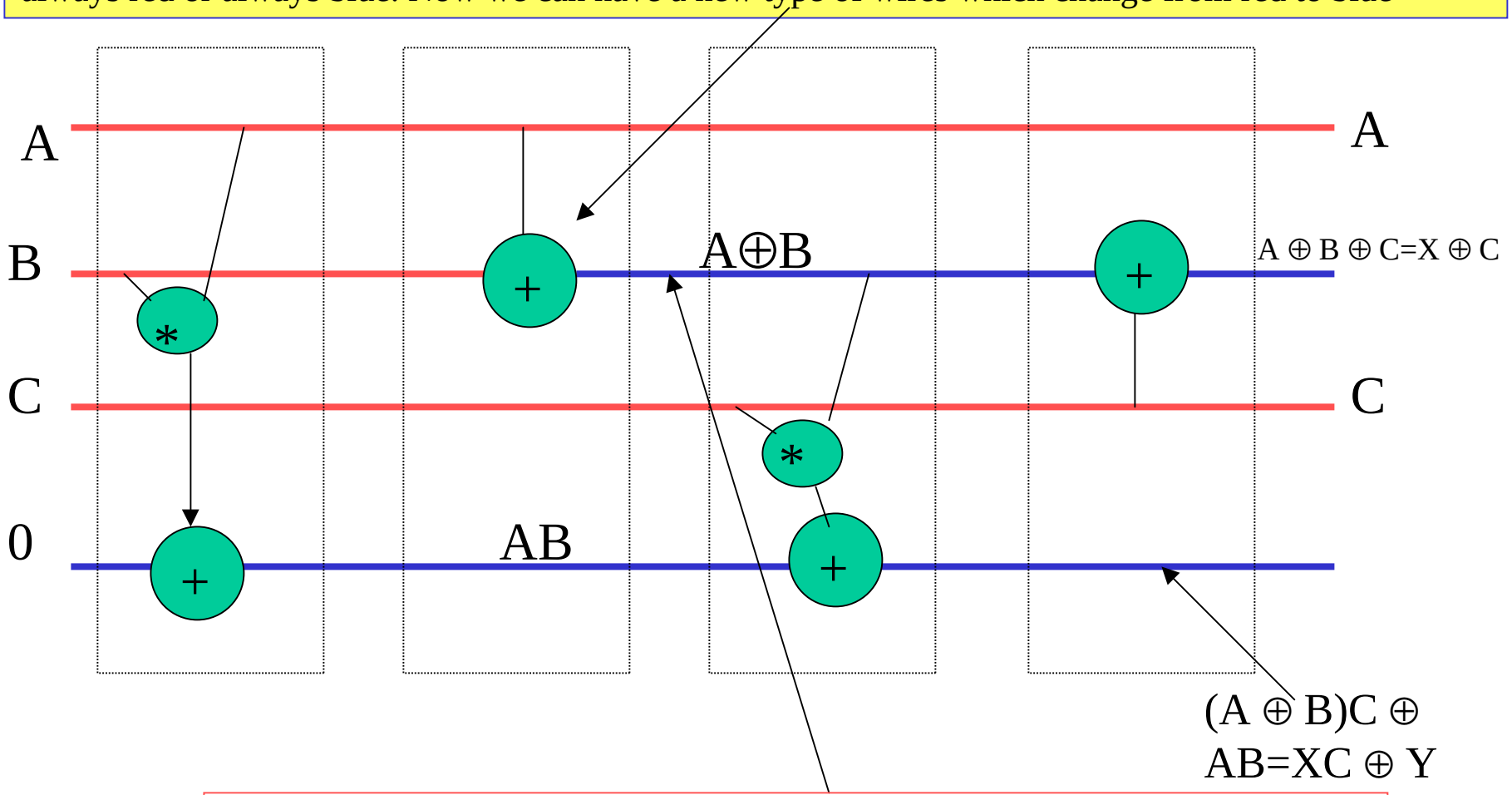
Quantum Notation for Full Adder realized using ESOP



Width 5, six gates, two constant, not optimal but easy to find

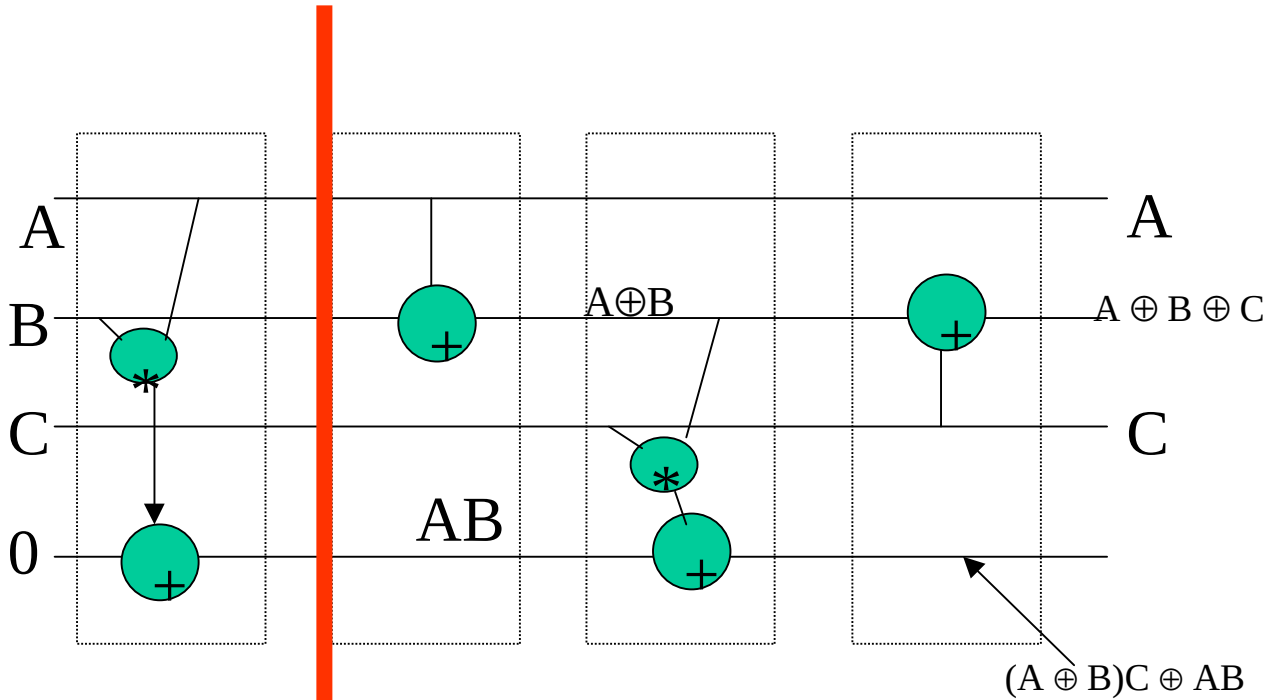
New Backtracking idea for Adder Realization

Red are input wires and blue are output wires. Normally in the past you assumed that every wire is always red or always blue. Now we can have a new type of wires which change from red to blue



Observe that variable B is no longer useful since output functions can be expressed only in terms of functions X=A ⊕ B and Y= AB

New Backtracking idea for Adder Realization



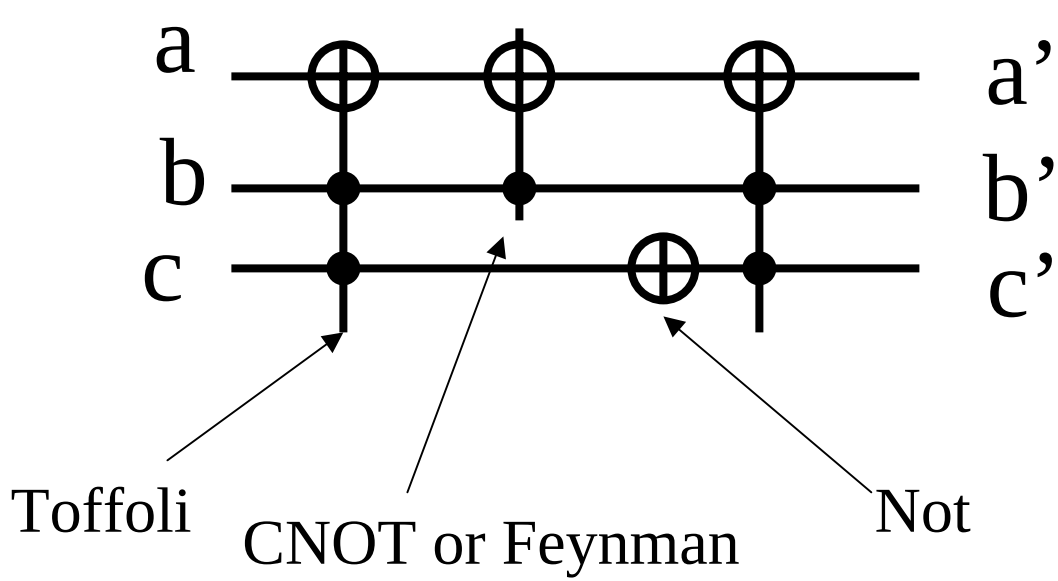
Add gates starting from input level

Count how much of the circuit is already realized (usually only a prediction)

Backtrack to previous part of circuit if you are not successful

A Reversible Circuit & Truth Table

a	b	c	a’	b	c’
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	0



This is called quantum notation

Reversible Circuits & Permutations

- A reversible gate (or circuit) with n inputs and n outputs has 2^n possible input values, and the 2^n possible output values
- The function it computes on this set must, by definition, be a permutation
- The set of such permutations is called $\mathbf{S_{2n}}$

Basic Facts About Permutations

- Permutations are multiplied by first doing one, then the other
- Every permutation can be written as the product of “transpositions”, that is, permutations which switch two indices and leave the rest fixed

Even Permutations

- For a fixed permutation, the parity of the number of transpositions in such a product is constant
- The permutations which may be written as the product of an even number of transpositions are called even

Known Facts

- Any reversible circuit with n+1 inputs and n+1 outputs, built from gates which have at most n inputs and n outputs, must compute an even permutation
- Any permutation may be computed in a circuit using the CNOT, NOT, and TOFFOLI gates, and a sufficient amount of temporary storage

Zero-Storage Circuits

- We can show that every even permutation can be computed in a circuit composed of CNOT, NOT, and TOFFOLI gates which uses no temporary storage
- For an arbitrary permutation, at most one line of temporary storage is necessary

Zero-Storage Circuits

- Roughly, the proof proceeds as follows
  - Pick an even permutation, and write it as the product of an even number of transpositions
  - Pair these up
  - Explicitly construct a circuit to compute an arbitrary transposition pair
- The proof is constructive, and may be used as a synthesis heuristic

Reversible De Morgan's Laws

- De Morgan's Laws allow all inverters in an irreversible circuit to be pushed to the inputs
- The same may be done for a reversible circuit containing only CNOT, NOT, and TOFFOLI gates

Reversible De Morgan's Laws

- Similar rules exist for interchanging TOFFOLI and CNOT gates
- However, it is not always possible to push all the CNOT gates to the inputs
- Oddly enough, using different methods, it is possible to **push all CNOT gates to the middle of the circuit**!

Optimality

- The cost of a circuit is its gate count
  - first approximation
- A reversible circuit is optimal if no circuit with fewer gates computes the same permutation
- Any sub-circuit of an optimal circuit must be optimal;
  - otherwise, the sub-optimal sub-circuit could be replaced with a smaller one

IDA* Search

- Checks all possible circuits of cost 1, then all possible circuits of cost 2, &c.
- Avoids the memory blowup of a BFS
- Still finds optimal solutions
- Checking circuits of cost less than n takes a small amount of time relative to that spent checking cost n circuits

IDA* Search

- Must provide a subroutine to check all circuits of cost n, for arbitrary n
- Only need to check locally optimal circuits

Circuit Libraries

- Store small, locally optimal circuits
  - Index by permutation
  - Use STL hash_map
- Recursively build larger circuits

Grover's Search

- A quantum search algorithm
- Runs in time O(sqrt(N))
- Requires a subroutine that changes the phase of any basis states which match the search criteria

Pseudo-classical Synthesis

- Adding a qubit initialized to |0> - |1> changes this into a problem in classical reversible circuit synthesis
- This can be solved by our methods

Towards Designing New Reversible Gates