Final Exam

- You have to solve all problems in group of problems 1 - 10. If you solve them correctly you have a grade of A+
- Any number of additional problems can be solved for additional credit.
- Each problem is assigned certain number of points. The total of points determines the grade.

Problem 1.

- The conditional phase shift that maps |0> to |0> and when j ≠0 | j > to -| j >. Show that the unitary operator corresponding to this is

$$2|0\rangle\langle 0| - I$$

where I is the identity transformation

5 points

Solution to Problem 1

$|0\rangle = [1\ 0]^T$ $\langle 0| = [1\ 0]$

$$|0\rangle\langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} [1 \quad 0] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$2|0\rangle\langle 0| - I = 2\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

You can check from definition that this matrix is unitary, maps |0> to |0>, and maps |j> to -|j>

As an example:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} | j\rangle = j\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = j\begin{bmatrix} 0 \\ -1 \end{bmatrix} = -j\begin{bmatrix} 0 \\ 1 \end{bmatrix} = -| j\rangle$$

In general:

$$2|0\rangle\langle 0| - I = \begin{bmatrix} 2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & -1 \end{bmatrix}$$

Problem 2. Quantum Dot Cellular Automata, Davio Gates and Lattices.

- Be able to observe here the following:
- the majority gate
- the inverter
- the fan-out
- the cross-over
- the way of laying out gates and connecting them that accommodates for inverters in the wires.
- The timing for Cellular Automata model

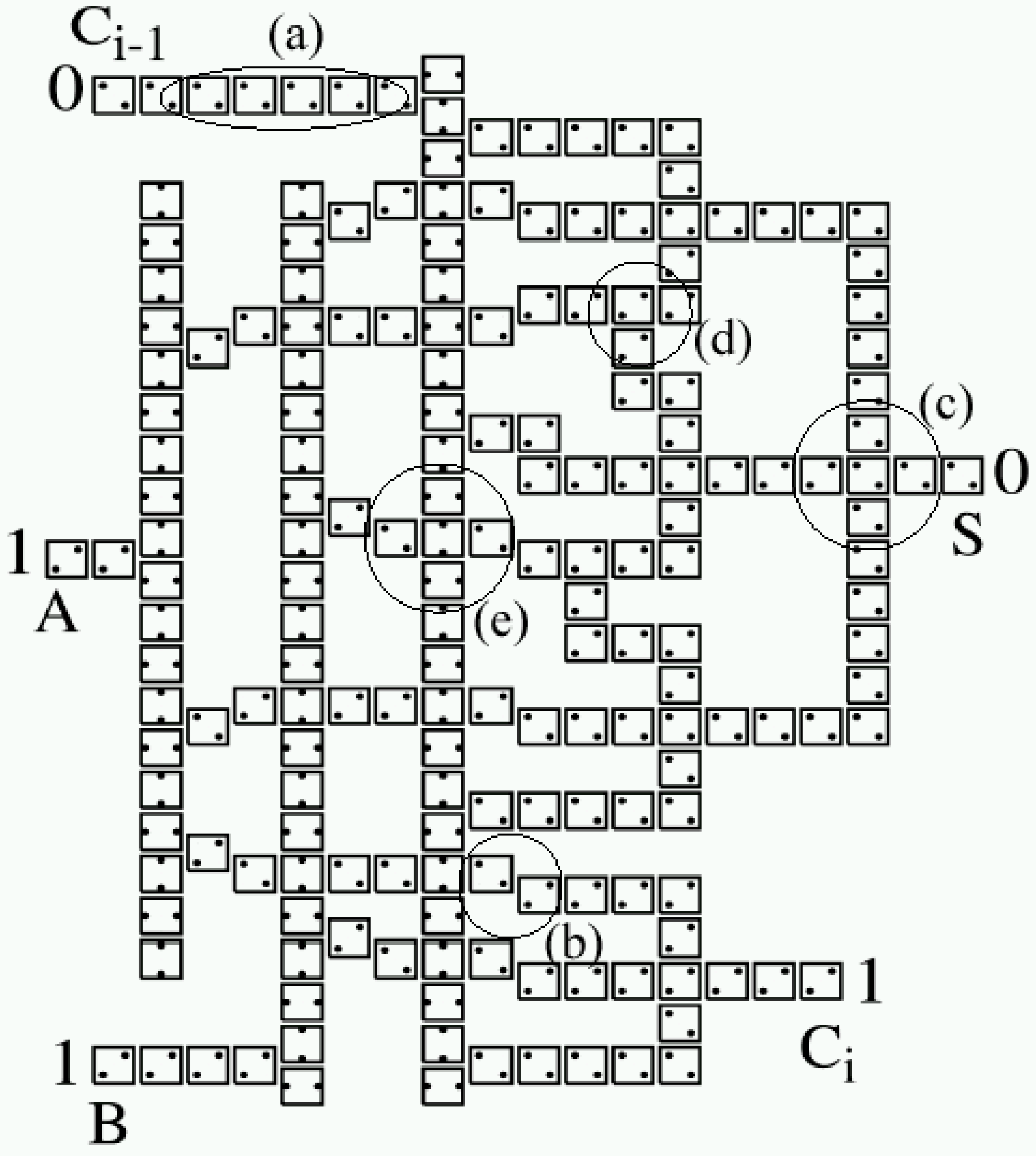


Figure 2. QCA single bit full adder. The following sections are labeled: a) wire, b) inverter, c) majority gate, d) fan-out, and e) cross-over.

Problem 2 (cont)

- 1. Given is a quantum dot circuit from Figure. 2. Draw Karnaugh maps and write the functions for each gate. This way you will explain how function S is created using majorities. (Function C_i is easy).
- 2. Use the synthesis method that you found by an analysis in point 1 to realize the function of Shannon Gate (Shannon Expansion).How many gates you need at the minimum? Remember that the cell should be stackable to regular structures such as lattices
- 3. Show how the Shannon Lattice can be realized in Quantum Dot logic using the Shannon gates as realized in point 2. Use grid paper and be neat to show exact timing. Annotate functions of each gate, including inverters.
- 4. Show how function F = **F=B+B'(AC+A'C')** is realized in this lattice. You can use the block scheme and not a detailed schematics to represent the Shannon gate in this lattice.

20 points

Solution to Problem 2.

Nobody even tried to solve this problem so I am not showing the solution. However, to help you think about this problem, I show below how the Shannon Lattice can be realized. If you then go back to lectures and read about Davio gates you will figure out how to realize Davio gate from majorities. This can be done in several ways.

Please observe the dotting for each type of element, wire, gate or intersection of wires. We will use the same in next figures.

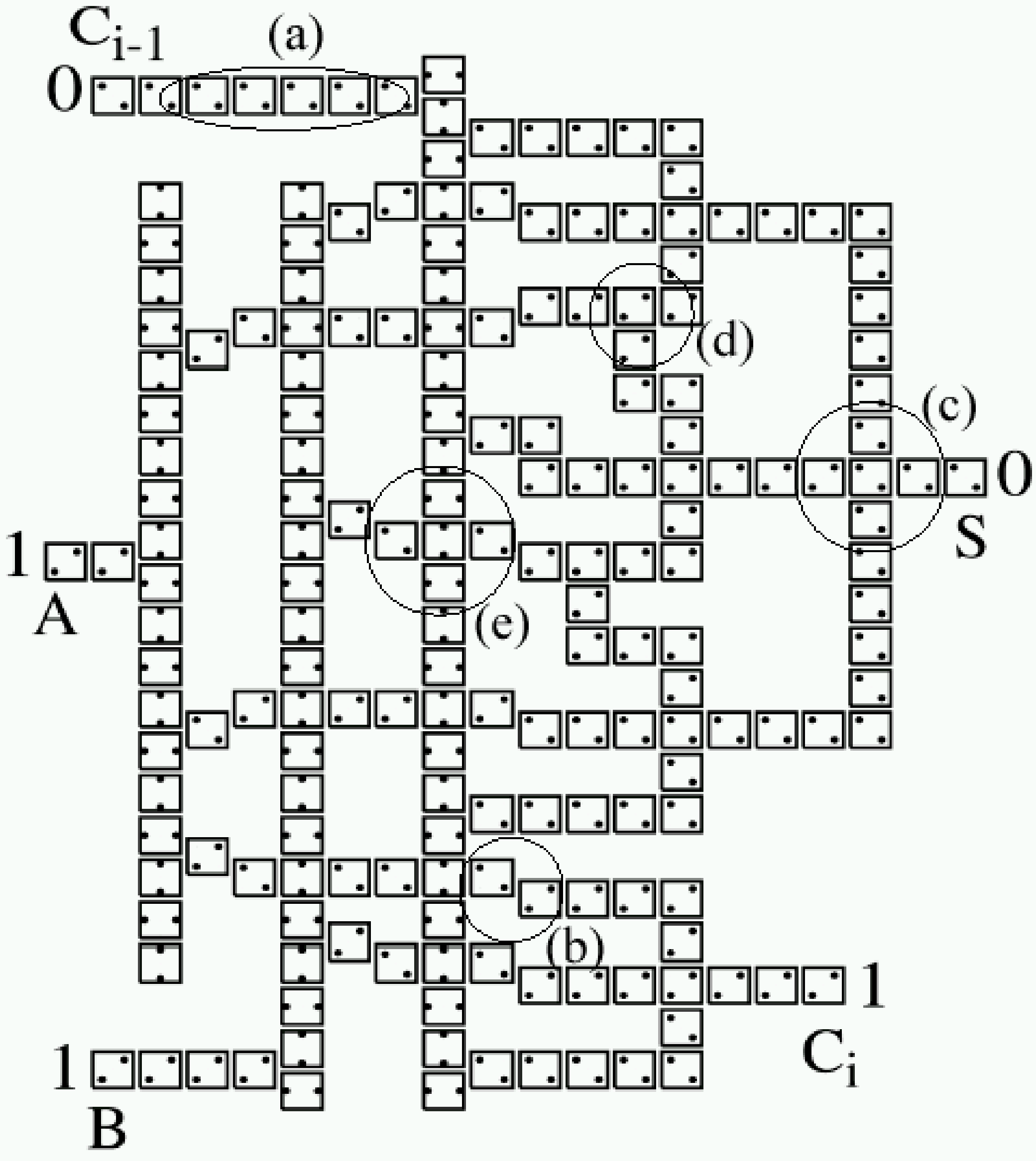
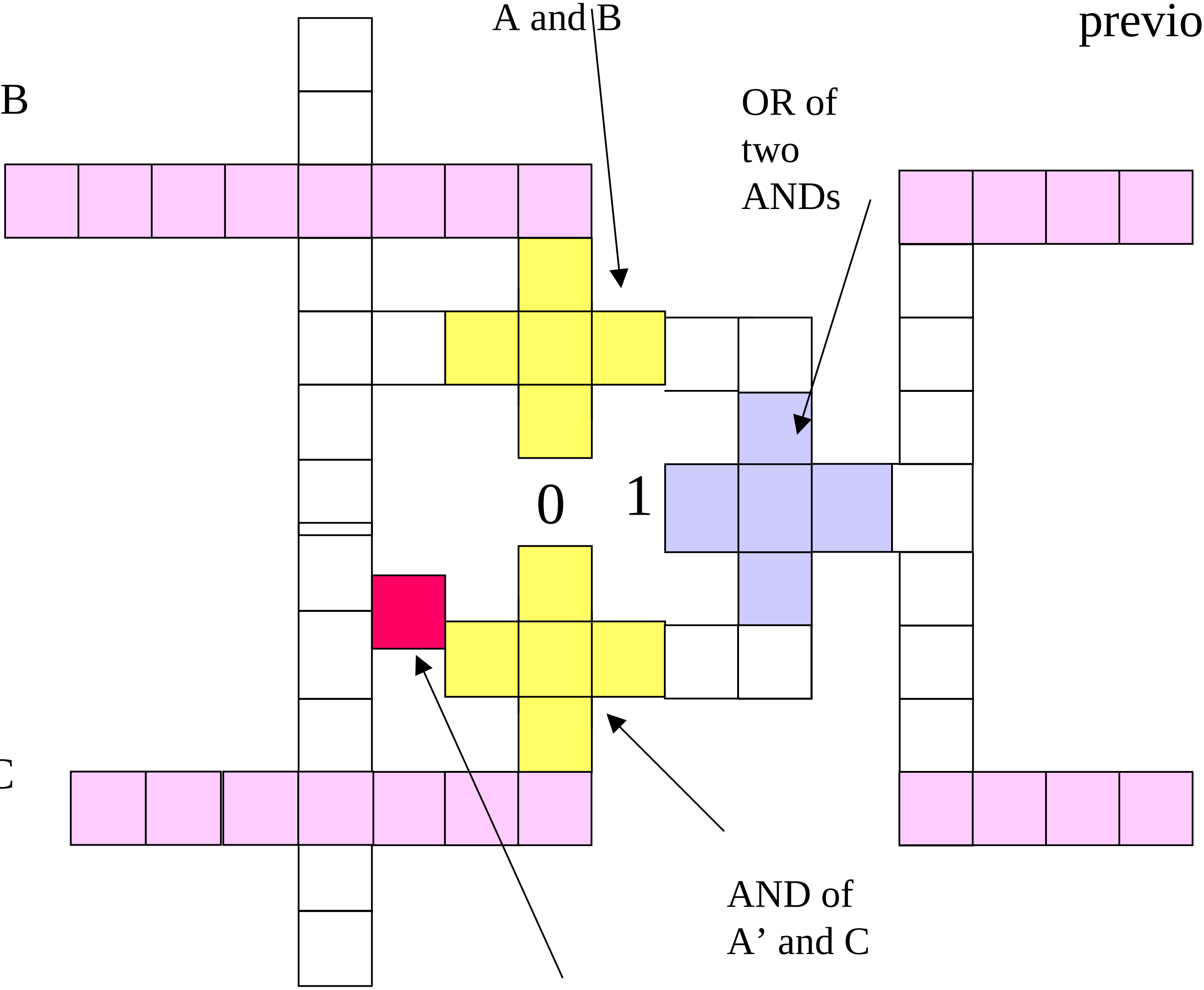


Figure 2. QCA single bit full adder. The following sections are labeled: a) wire, b) inverter, c) majority gate, d) fan-out, and e) cross-over.

Use dotting like in previous schematics



AND of
A and B

Data
variable B

OR of
two
ANDs

Data
variable C

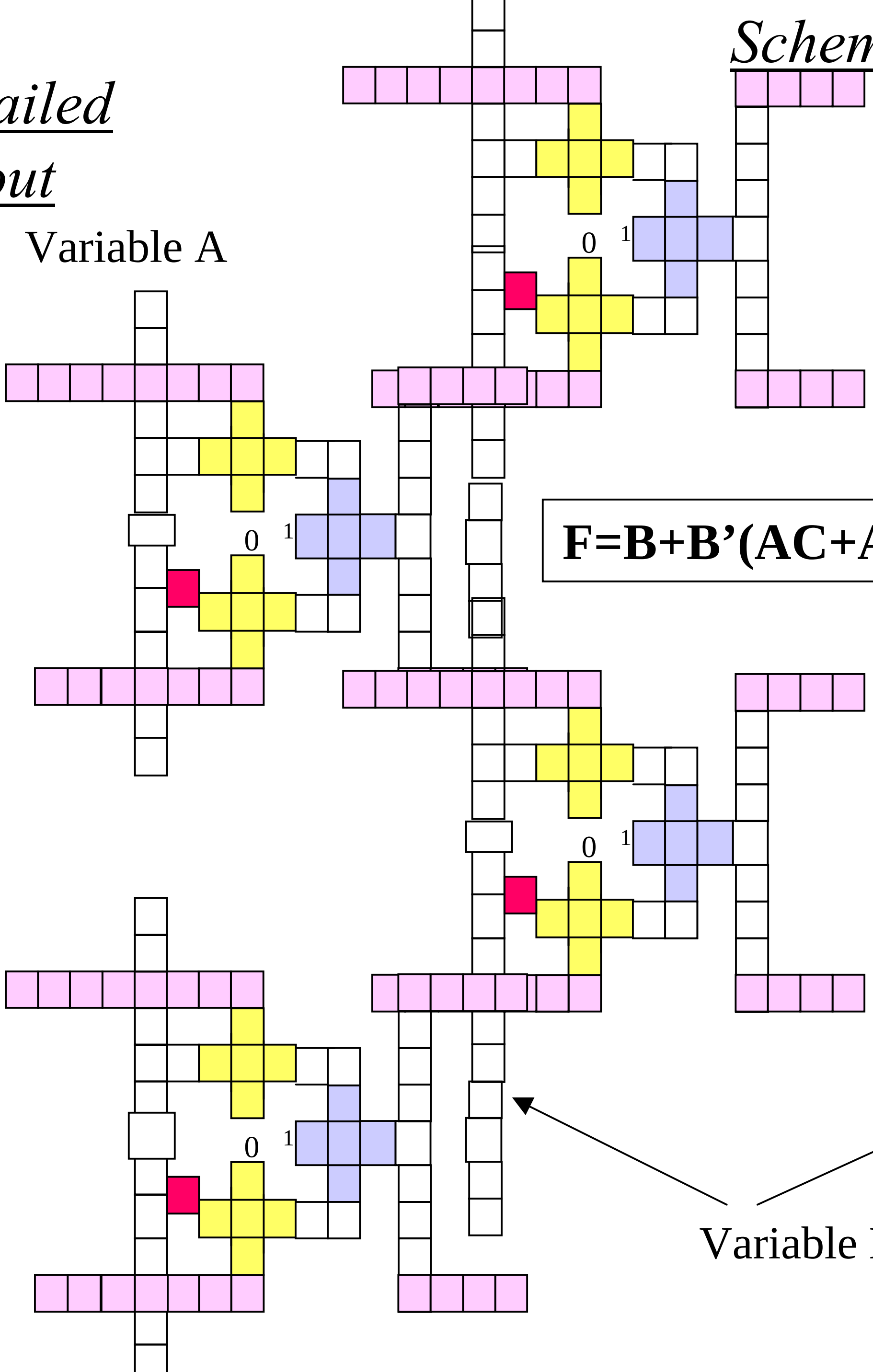
AND of
A’ and C

inverter

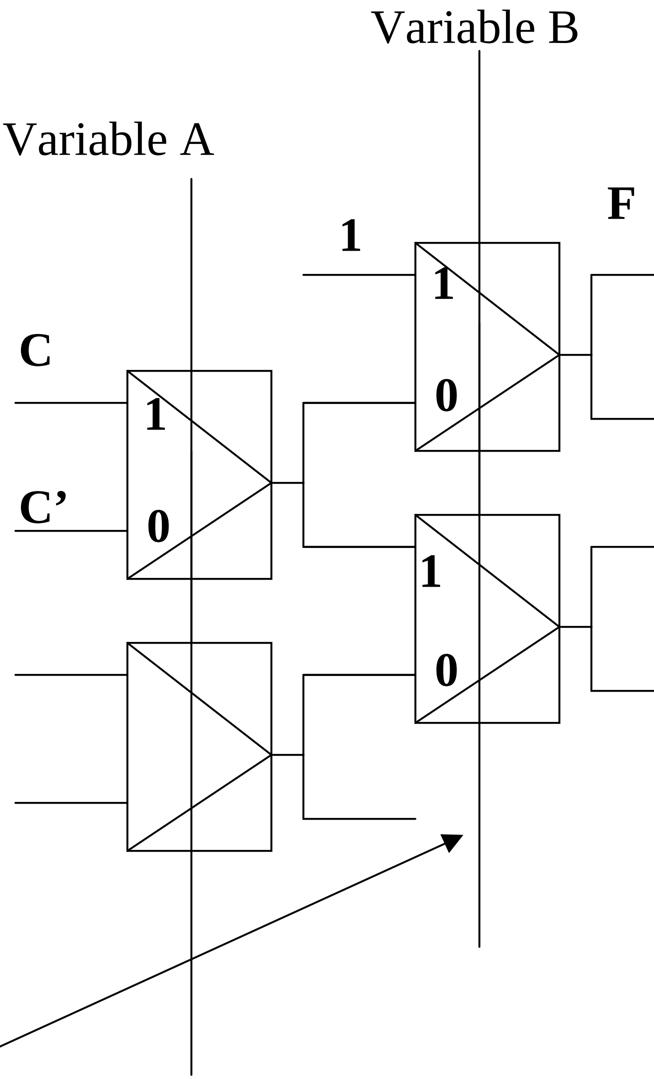
Control
variable A

This diagram shows a multiplexer with vertical control variable, prepared for stacking in a regular structure

Detailed layout



Schematics of this layout



F=B+B’(AC+A’C’)

Variable B

Part of a regular tissue (fabric) for Shannon Lattice using Quantum Dot Cellular Automata

Problem 3.

Calculate the eigenvalues and eigenvectors associated with the following matrix

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

10 points

Solution to Problem 3

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Solution

$$B = \begin{bmatrix} 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix}$$

Eigenvalues given by $\det[\lambda I - B] = 0$

$$\begin{bmatrix} \lambda - 1 & 0 & -2 & 0 \\ 0 & \lambda - 1 & 0 & -2 \\ -2 & 0 & \lambda - 1 & 0 \\ 0 & -2 & 0 & \lambda - 1 \end{bmatrix} = 0$$

Eigenvalues given by

$$\begin{bmatrix} \lambda-1 & 0 & -2 & 0 \\ 0 & \lambda-1 & 0 & -2 \\ -2 & 0 & \lambda-1 & 0 \\ 0 & -2 & 0 & \lambda-1 \end{bmatrix} = (\lambda-1)\det\begin{bmatrix} \lambda-1 & 0 & -2 \\ 0 & \lambda-1 & 0 \\ -2 & 0 & \lambda-1 \end{bmatrix} - 2\det\begin{bmatrix} 0 & \lambda-1 & -2 \\ -2 & 0 & 0 \\ 0 & -2 & \lambda-1 \end{bmatrix}$$

$$= (\lambda-1)\left\{(\lambda-1)(\lambda-1)^2 - 2\left(2(\lambda-1)\right)\right\} - 2\left\{-(\lambda-1)\det\begin{bmatrix} -2 & 0 \\ 0 & \lambda-1 \end{bmatrix} - 2\det\begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}\right\}$$

$$= (\lambda-1)^4 - 8(\lambda-1)^2 + 16$$

$$= \left[(\lambda-1)^2 - 4\right]^2 = 0$$

$$\boxed{\lambda = -1,\ 3}$$

-1 **3**

Eigenvectors given by

$$\begin{bmatrix}1 & 0 & 2 & 0\\ 0 & 1 & 0 & 2\\ 2 & 0 & 1 & 0\\ 0 & 2 & 0 & 1\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\\ v_4\end{bmatrix}=-\begin{bmatrix}v_1\\ v_2\\ v_3\\ v_4\end{bmatrix} \qquad \begin{bmatrix}1 & 0 & 2 & 0\\ 0 & 1 & 0 & 2\\ 2 & 0 & 1 & 0\\ 0 & 2 & 0 & 1\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\\ v_4\end{bmatrix}=3\begin{bmatrix}v_1\\ v_2\\ v_3\\ v_4\end{bmatrix}$$

From above we get:

$$\begin{bmatrix}v_1+2v_3\\ v_2+2v_4\\ 2v_1+v_3\\ 2v_2+v_4\end{bmatrix}=\begin{bmatrix}-v_1\\ -v_2\\ -v_3\\ -v_4\end{bmatrix} \qquad \begin{bmatrix}v_1+2v_3\\ v_2+2v_4\\ 2v_1+v_3\\ 2v_2+v_4\end{bmatrix}=\begin{bmatrix}3v_1\\ 3v_2\\ 3v_3\\ 3v_4\end{bmatrix}$$

Eigenbasis

$$\begin{bmatrix}0\\ 1\\ 0\\ -1\end{bmatrix},\begin{bmatrix}1\\ 0\\ -1\\ 0\end{bmatrix} \qquad \begin{bmatrix}0\\ 1\\ 0\\ 1\end{bmatrix},\begin{bmatrix}1\\ 0\\ 1\\ 0\end{bmatrix}$$

Eigenbasis is found from the above matrix equation

Another approach

Eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$

$$\det\begin{bmatrix} \lambda-1 & -2 \\ -2 & \lambda-1 \end{bmatrix} = 0 \quad \lambda = -1,\ 3$$

Eigenvectors

$$\lambda = -1:\ v_2 = -v_1 \qquad \lambda = 3:\ v_2 = v_1$$

Eigenbasis

$$\lambda = -1: \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\lambda = \ 3: \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\lambda = 1$$

$$\lambda = 1:\ v_2, v_1$$

$$\lambda = 1: \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\lambda = 1: \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Eigenbasis of B, eigenvalue λ= -1

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Eigenbasis of B, eigenvalue λ= 3

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Problem 4. Set finding problems like Petrick Function, Solving Boolean Equations, Graph Coloring, Set covering

- These problems are finding a subset of certain set. We illustrated pipelined processors for some of them. We showed also a binary tree architecture for one of these problems.
- Discuss a pipelined architecture for solving any of the above problems.
- Draw the pipeline stage, the registers and logic blocks in it.
- Detailed specification of control on logic level is not required. Show example of operation. Discuss the efficiency of your approach. Compare to purely software approach.

20 points

Solution to Problem 4. SAT problem

- **Draw the pipeline stage, the registers and logic blocks in it.**
- **Detailed specification of control on logic level is not required. Show example of operation. Discuss the efficiency of your approach. Compare to purely software approach.**

We want to find a product term of literals that satisfies the product of sums of literals satisfiability formula.

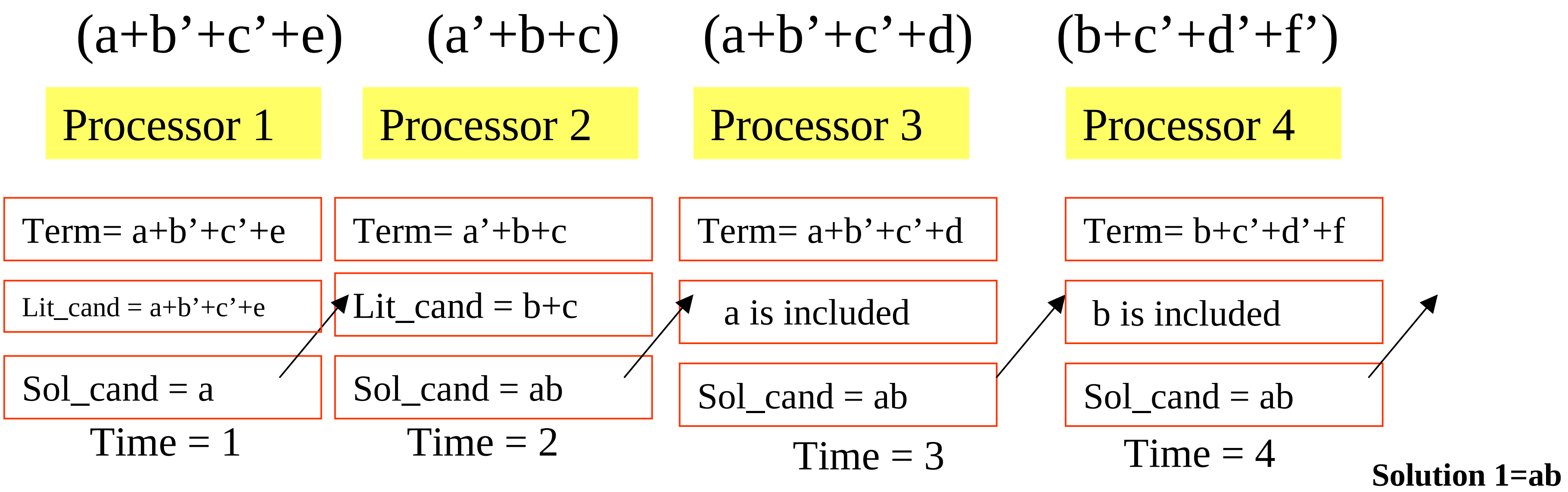
Example: term1 term2 term3

$$(a+b'+c'+e)(a'+b+c)(a+b'+c'+d)(b+c'+d'+f') = 1$$

The above formula is satisfied by a product b’a’c’.

The idea of our pipelined processor is a simplification of the processor to solve satisfiability with least product term cost. Each processor has one term. The solution candidate is a literal product term that is forwarded from left to right. Suppose a is generated randomly to satisfy the first sum term. It is send to the right to processor that includes term 2. In this term the variable a is negated so it must be removed from the set of candidates. The candidates are b and c. Randomly one of them, say b, is generated and added to the solution candidate. Now, after leaving term 2, the solution candidate is ab. In term 3 a is already present, so no literal is added to the solution candidate when passing this processor. The same situation is in term 4, so ab is generated as a solution when it leaves processor 4. In some big problems there is no satisfiable product or it is very hard to find. So we use pipelining. When the first solution candidate leaves processor 3, the second solution candidate leaves processor 2 and the third solution candidate leaves processor 1.

Solution to Problem 4. SAT problem



(a+b’+c’+e) (a’+b+c) (a+b’+c’+d) (b+c’+d’+f’)

Processor 1

Term= a+b’+c’+e

Lit_cand = a+b’+c’+e

Sol_cand = a

Time = 1

Processor 2

Term= a’+b+c

Lit_cand = b+c

Sol_cand = ab

Time = 2

Processor 3

Term= a+b’+c’+d

a is included

Sol_cand = ab

Time = 3

Processor 4

Term= b+c’+d’+f

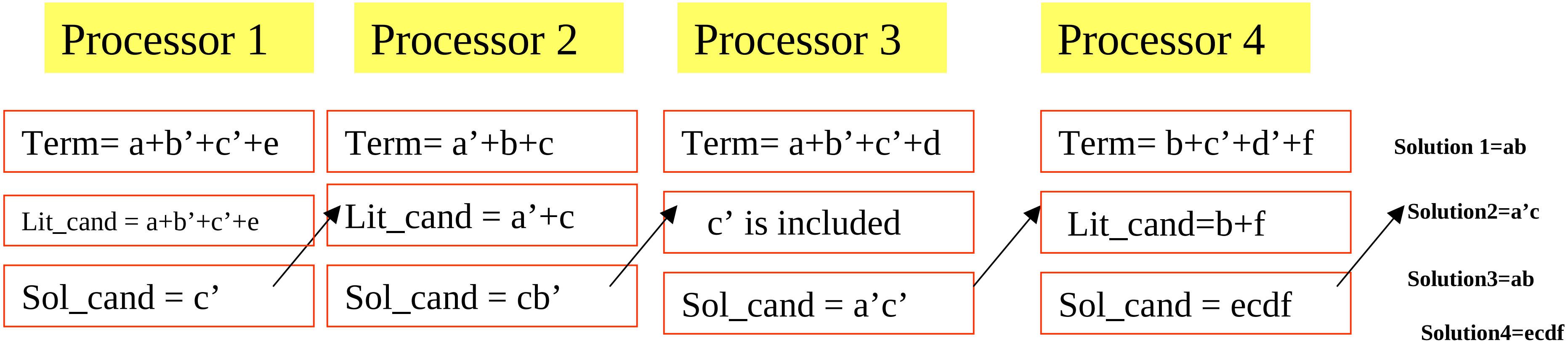
b is included

Sol_cand = ab

Time = 4

Solution 1=ab

This figure explains the main idea of operation



Processor 1

Term= a+b’+c’+e

Lit_cand = a+b’+c’+e

Sol_cand = c’

Processor 2

Term= a’+b+c

Lit_cand = a’+c

Sol_cand = cb’

Processor 3

Term= a+b’+c’+d

c’ is included

Sol_cand = a’c’

Processor 4

Term= b+c’+d’+f

Lit_cand=b+f

Sol_cand = ecdf

Solution 1=ab

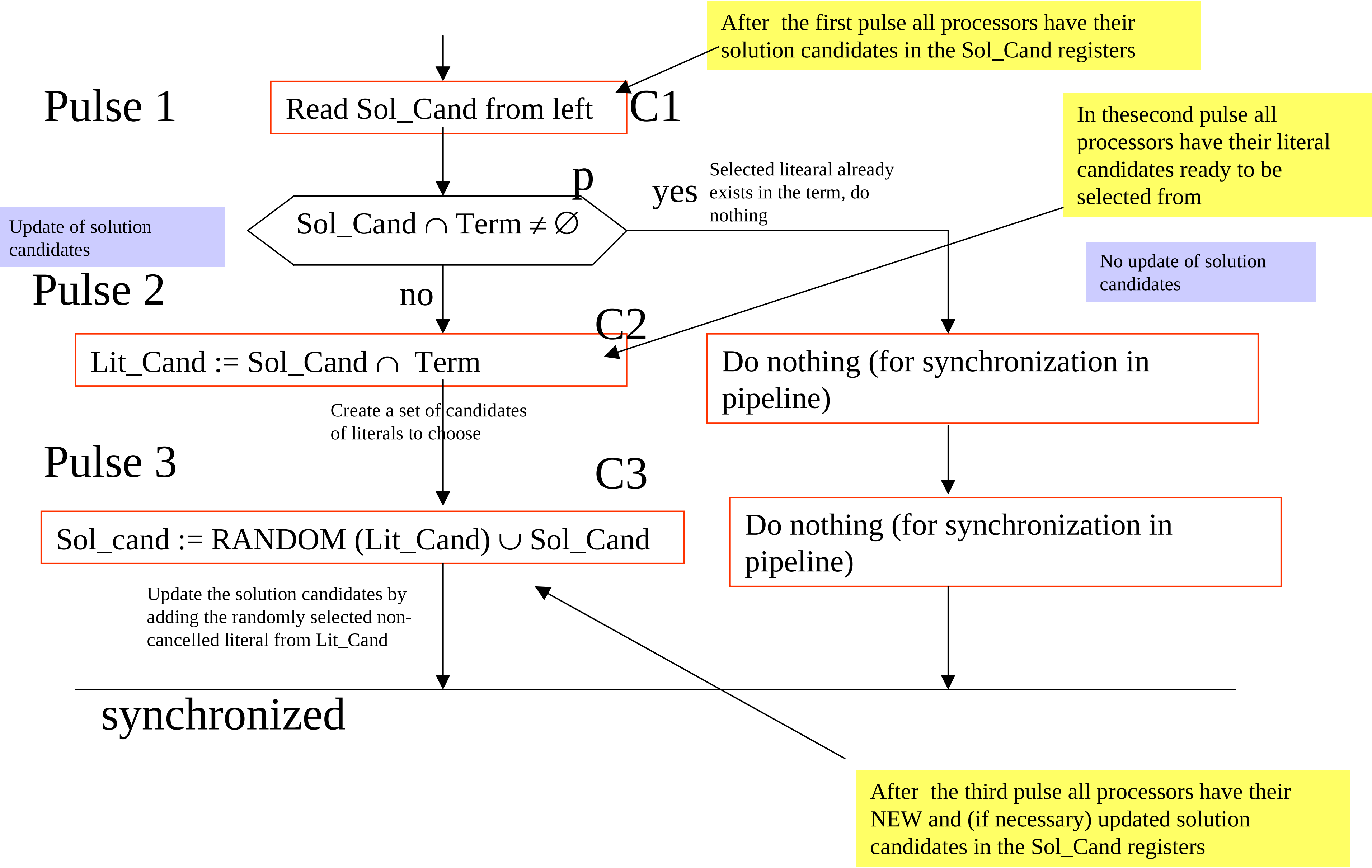
Solution2=a’c

Solution3=ab

Solution4=ecdf

This figure shows snapshot of pipelining at the end of time moment 4

Now that we understand the principle and the pipelining we can write the control and create the data path for each processor.



Pulse 1

Read Sol_Cand from left

C1

After the first pulse all processors have their solution candidates in the Sol_Cand registers

p

Sol_Cand $\cap$ Term $\neq \emptyset$

yes

Selected litearal already exists in the term, do nothing

Update of solution candidates

In thesecond pulse all processors have their literal candidates ready to be selected from

No update of solution candidates

Pulse 2

no

C2

Lit_Cand := Sol_Cand $\cap$ Term

Do nothing (for synchronization in pipeline)

Create a set of candidates of literals to choose

Pulse 3

C3

Sol_cand := RANDOM (Lit_Cand) $\cup$ Sol_Cand

Do nothing (for synchronization in pipeline)

Update the solution candidates by adding the randomly selected non-cancelled literal from Lit_Cand

synchronized

After the third pulse all processors have their NEW and (if necessary) updated solution candidates in the Sol_Cand registers

Details of the design

- Encoding of the sets in positional notation:

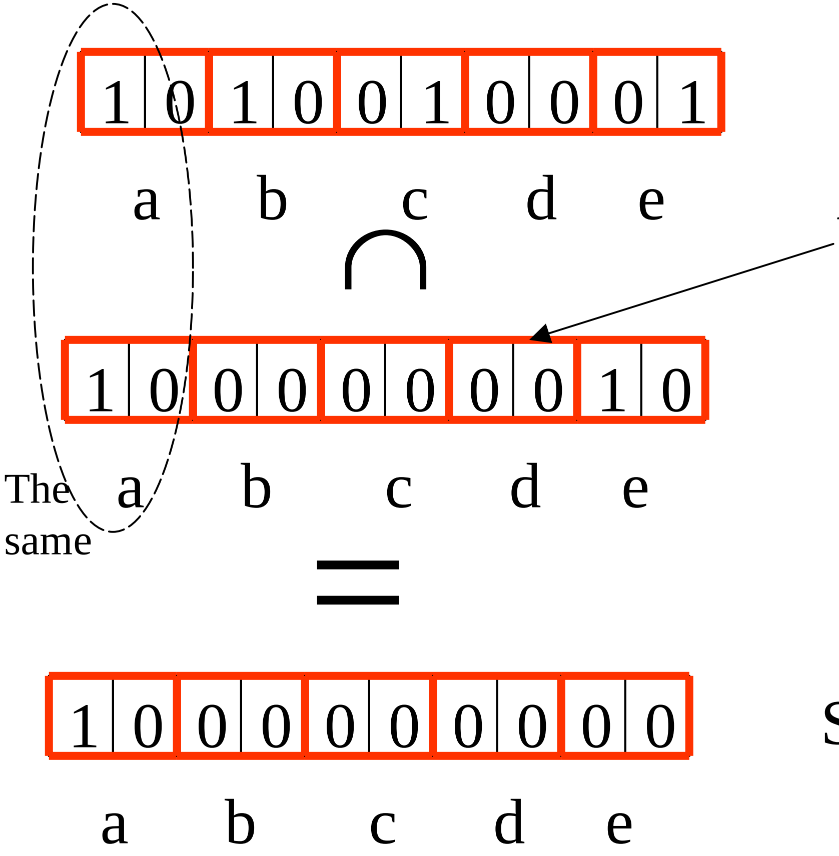
  10 encodes literal a’

  01 encodes literal a

  00 or 11 encodes literal absent

Encoding of a’b’c e

1 0	1 0	0 1	0 0	0 1
a	b	c	d	e



1 0	1 0	0 1	0 0	0 1
a	b	c	d	e

Term a’ + b’ + c + e

$\cap$

Absent represented as zeros

1 0	0 0	0 0	0 0	1 0
a	b	c	d	e

Sol_Cand = a’ + e’

=

1 0	0 0	0 0	0 0	0 0
a	b	c	d	e

Sol_Cand $\cap$ Term $\neq \varnothing$

This slide explains how to create intersection in one pulse using special encoding of boolean data

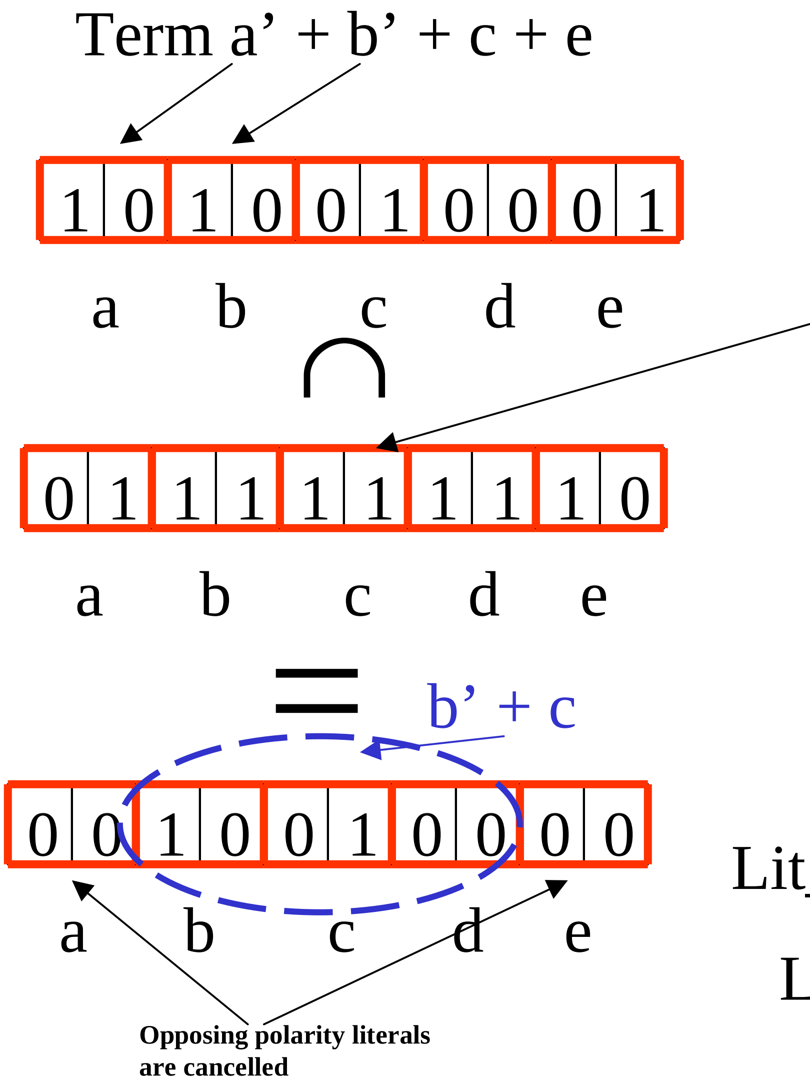
Logic operation 1

Details of the design

Logic operation 2

Encoding of a’b’c e

1	0	1	0	0	1	0	0	0	1
a		b		c		d		e	



Term a’ + b’ + c + e

1	0	1	0	0	1	0	0	0	1
a		b		c		d		e	

∩

0	1	1	1	1	1	1	1	1	0
a		b		c		d		e	

Absent represented as ones

=

b’ + c

0	0	1	0	0	1	0	0	0	0
a		b		c		d		e	

Opposing polarity literals are cancelled

Term a’ + b’ + c + e

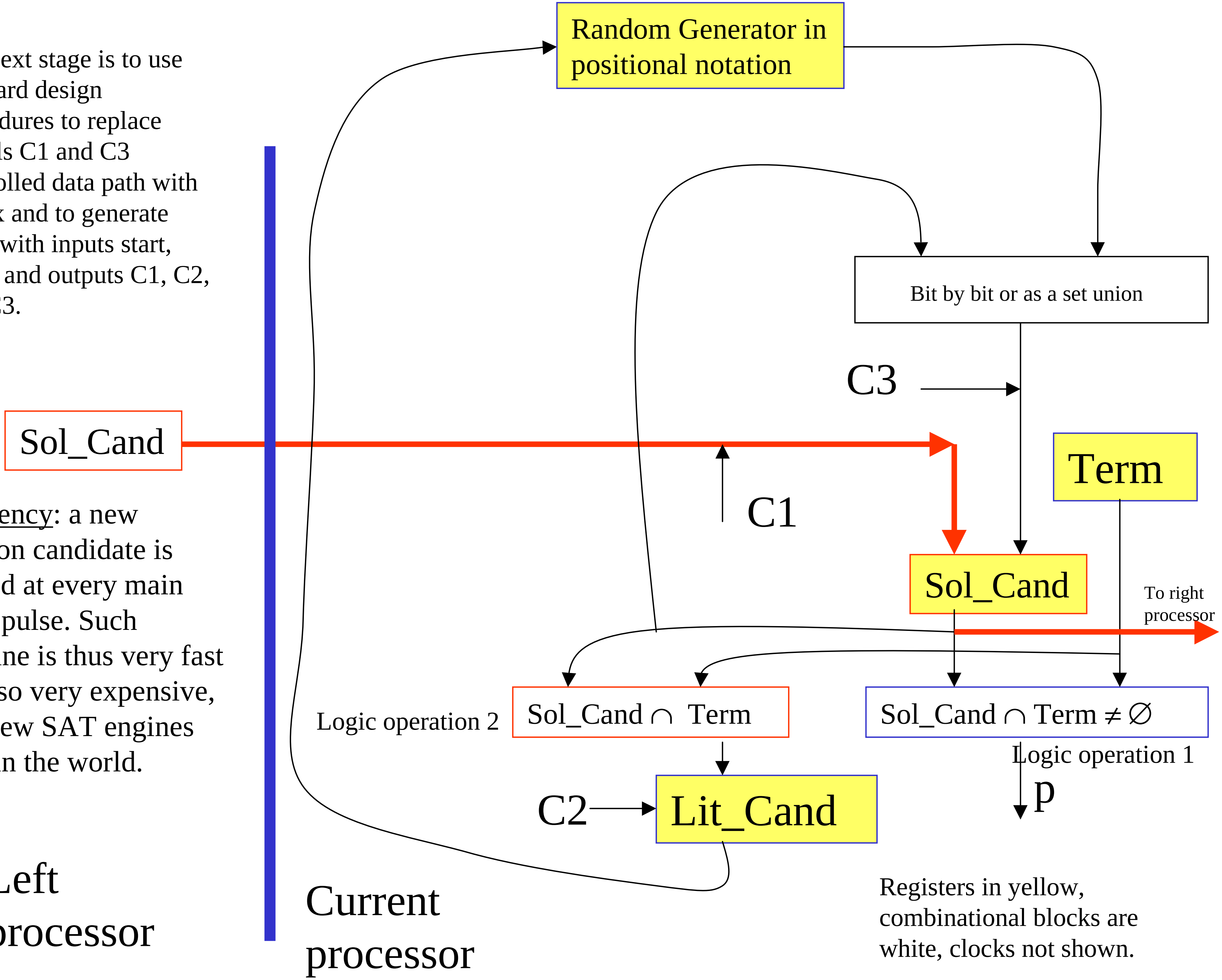
Sol_Cand = a + e’

Lit_Cand := Sol_Cand ∩ Term

Lit_Cand = b’ + c

This slide explains how to create set of literal candidates

The next stage is to use standard design procedures to replace signals C1 and C3 controlled data path with a mux and to generate FSM with inputs start, and p and outputs C1, C2, and C3.



Efficiency: a new solution candidate is created at every main clock pulse. Such machine is thus very fast but also very expensive, only few SAT engines exist in the world.

Registers in yellow, combinational blocks are white, clocks not shown.

Problem 5. Discuss Systolic or other architecture to solve the Graph Closure problem.

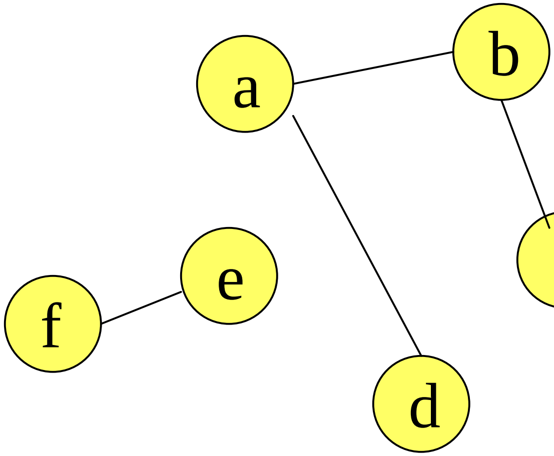
- Graph is represented as an incidence matrix. Find its closure.
- Discuss a pipelined, systolic, cellular, DNA or any other architecture for solving any of the above problems. Any type of architecture discussed in the class can be used, except of standard Von Neumann Processor
- Draw the data flow, the blocks, the registers and logic blocks.
- Explain how the control works, draw diagram or pseudocode or state machine.
- Detailed specification of control on logic level is not required.
- Show example of operation.
- Discuss the efficiency of your approach. Compare to purely software approach.

20 points

Solution to Problem 5.

Multiplying the incidence matrix by itself we get:

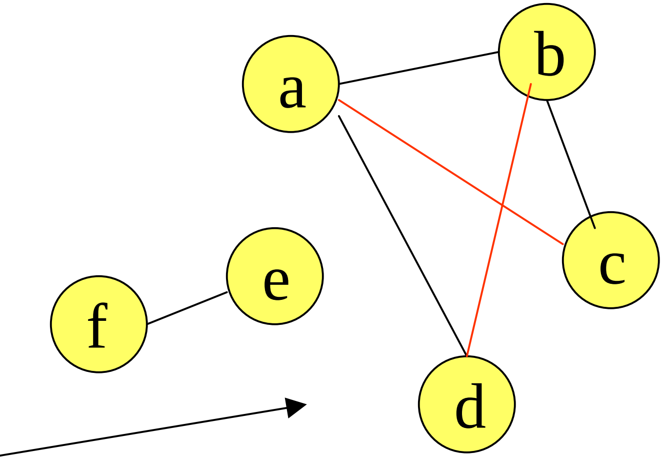
$$\begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline a & 1 & 1 & 1 & 1 & 0 & 0 \\ b & 1 & 1 & 1 & 1 & 0 & 0 \\ c & 1 & 1 & 1 & 0 & 0 & 0 \\ d & 1 & 1 & 0 & 1 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 1 & 1 \\ f & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$



graph

$$\begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline a & 1 & 1 & 0 & 1 & 0 & 0 \\ b & 1 & 1 & 1 & 0 & 0 & 0 \\ c & 0 & 1 & 1 & 0 & 0 & 0 \\ d & 1 & 0 & 0 & 1 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 1 & 1 \\ f & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Its incidence matrix M

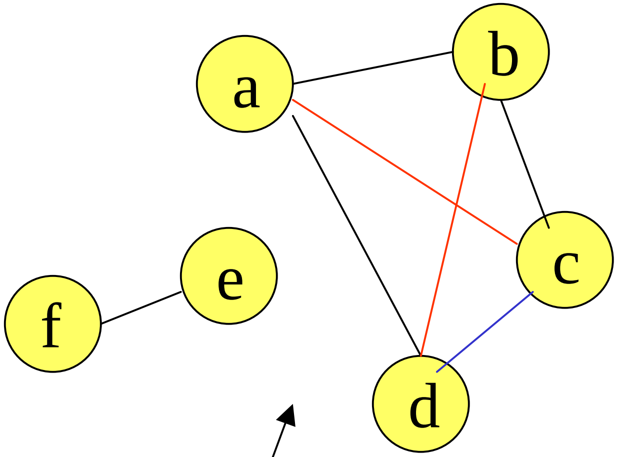


Graph of M*M

This matrix corresponds to this graph. New edges that result as closures are shown in red both in graph and matrix

We repeat multiplication.

Multiplying the incidence matrix M*M by M again we get:

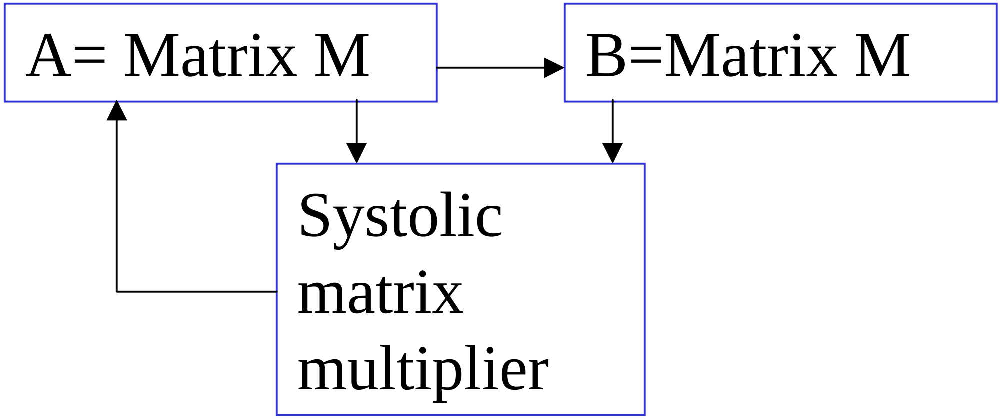


Graph of M*M*M

$$
\begin{array}{c|cccccc}
 & a & b & c & d & e & f \\
\hline
a & 1 & 1 & \mathbf{1} & 1 & 0 & 0 \\
b & 1 & 1 & 1 & \mathbf{1} & 0 & 0 \\
c & \mathbf{1} & 1 & 1 & 1 & 0 & 0 \\
d & 1 & \mathbf{1} & 1 & 1 & 0 & 0 \\
e & 0 & 0 & 0 & 0 & 1 & 1 \\
f & 0 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

This matrix corresponds to this graph. New edges that result as closures are shown in blue both in graph and matrix

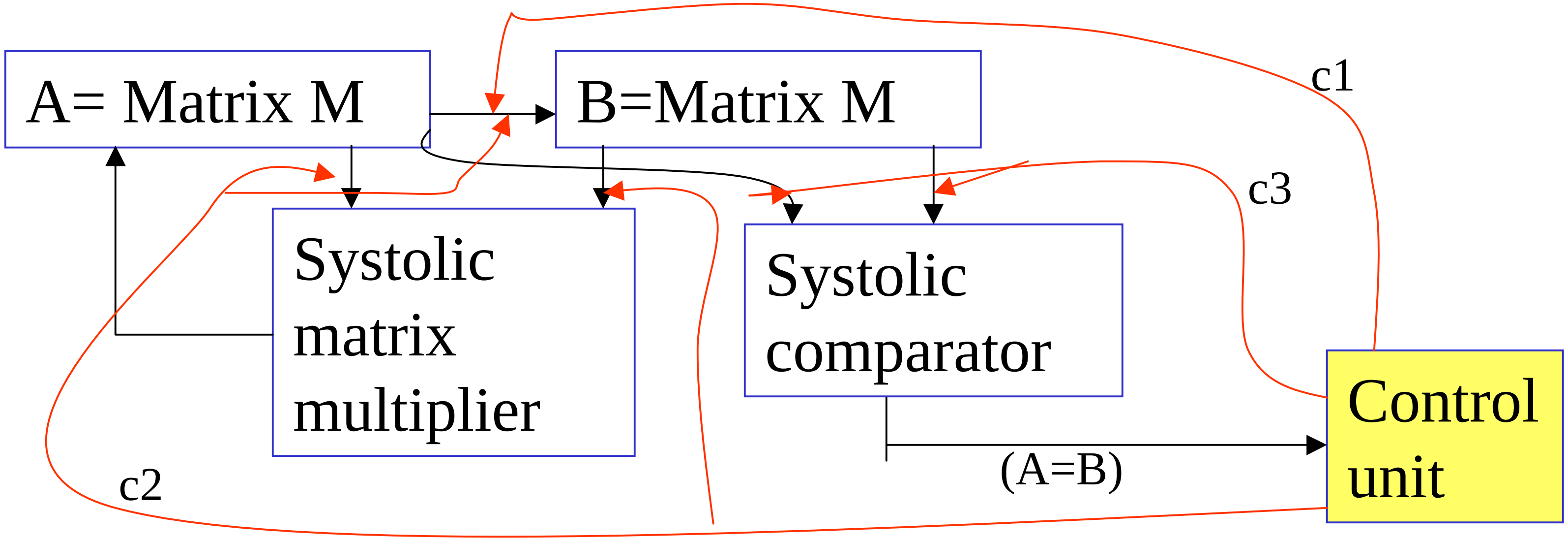
We repeat multiplication but the new matrix M*M*M*M is the same as the old matrix M*M*M. Thus we found the closure.



A= Matrix M

B=Matrix M

Systolic matrix multiplier



A= Matrix M

B=Matrix M

Systolic matrix multiplier

Systolic comparator

Control unit

c1

c2

c3

(A=B)

Registers A	Registers B	
M	M	
M*M	M	Not equal. Continue
M*M*M	M*M	Not equal. Continue
M*M*M*M	M*M*M	Equal. Stop

Detailed clocking not shown.

Algorithm

c1. Copy Matrix M from registers A to registers B

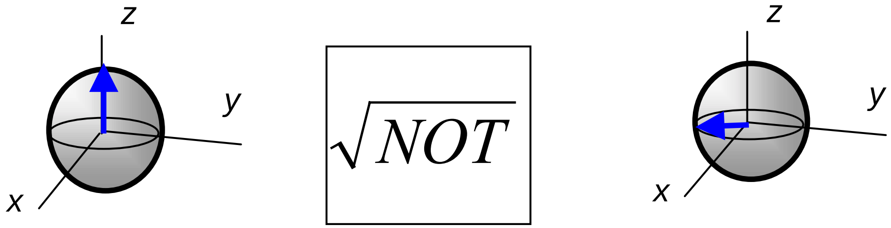
c2. Multiply A and B, send result of multiplication to A. Concurrently send the contents of registers A to registers B.

c3. If A=B then stop else go to 2.

- It remains to discuss how the subcircuits are designed.
- For matrix multiplication you can take the matrix multiplier from the class but the operations of algebraic multiplication are replaced by Boolean ANDing and operations of algebraic addition are replaced with Boolean ORing. This was the creative part of your task. This also explains the principle why matrix multiplication can be used in graph theoretical problems.
- Copying is just a register transfer.
- Operations in step 2 of algorithm can be pipelined, so multiplication and transfer are done at the same time. Of course , the number of pulses depends on the size of matrix M. See discussion in class material.
- The time to multiply matrices of size k is 3k-1 if the result remains in place.

Problem 6.

- In lecture 1 on QC we saw that the action of the square-root-of-NOT gate was to transform the |0> qubit to one pointing anti-parallel with the y-axis (see below). 1.
- 1. Explain this. Use matrices and vectors.
- 2. Explain what happens if two such gates are connected in series.



10 points

Solution to Problem 6

- Recall that

$$\sqrt{NOT}|0\rangle = \begin{bmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1+i \\ 1-i \end{bmatrix}$$

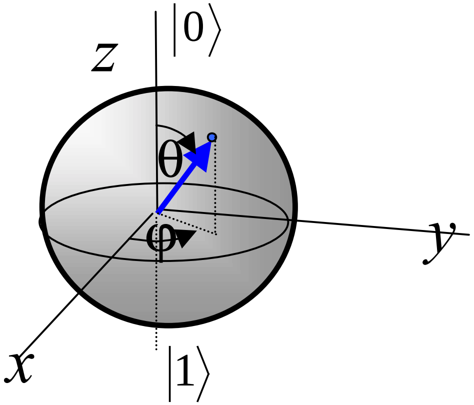
- This can be re-written

$$\sqrt{NOT}|0\rangle = \left(\frac{1+i}{2}\right)|0\rangle + \left(\frac{1-i}{2}\right)|1\rangle = \left(\frac{1+i}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1-i}{1+i}\right)|1\rangle\right)$$

$$\equiv \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

- So $\frac{\theta}{2} = \frac{\pi}{4}$ and $e^{i\varphi} = \frac{1-i}{1+i} = -i$

$$\theta = \frac{\pi}{2}, \ \varphi = \frac{3\pi}{2}$$



Problem 7.

- Explain why $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is entangled

5 points

Solution to Problem 7

$$\begin{aligned} |x\rangle \otimes |y\rangle &= (x_0|0\rangle + x_1|1\rangle) \otimes (y_0|0\rangle + y_1|1\rangle) \\ &= x_0 y_0|00\rangle + x_0 y_1|01\rangle + x_1 y_0|10\rangle + x_1 y_1|11\rangle \end{aligned}$$

Now for this to be decomposable the coefficients of |01> and |10> must be zero so x_0y_1 and x_1y_0 are zero, but this implies that either x_0y_0 or x_1y_1 are zero so that the given state is not produced.

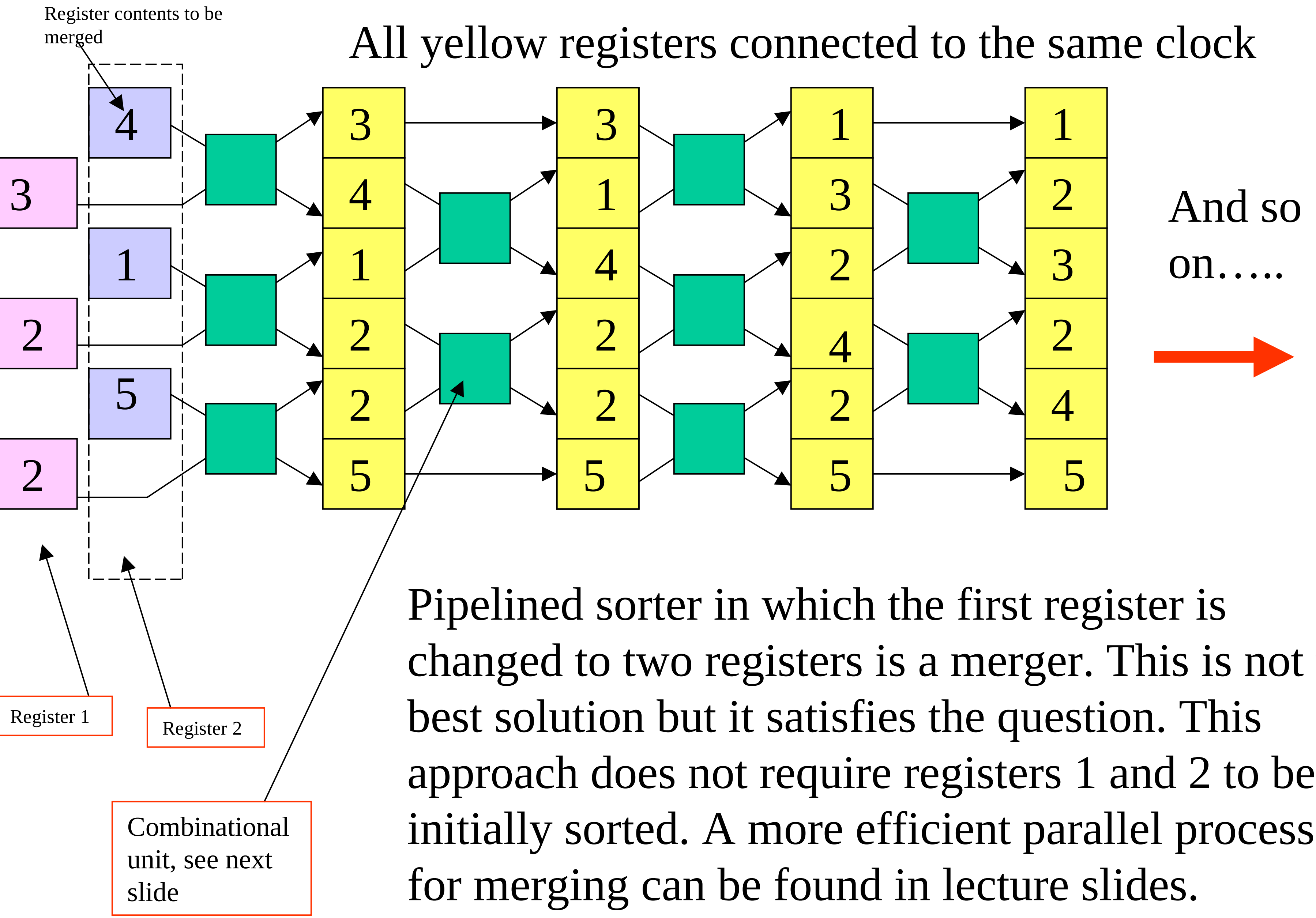
Thus the entangled state cannot be written as a tensor product of single-qubit states.

Problem 8. A processor for merging.

- In class we designed several massively parallel processors for sorting. Merging and sorting are similar operations.
- Design a highly parallel processor for merging two sets of numbers that removes repeated numbers from the final solution. It can be a systolic, combinational, Cellular or any other architecture, but must be highly parallel.

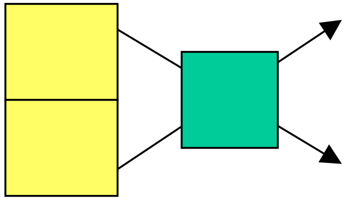
20 points

Solution to Problem 8. Adaptation of one of solutions from the class.



Register contents to be merged
All yellow registers connected to the same clock
4
1
5
3
2
2
3
4
1
2
2
5
3
1
4
2
2
5
1
3
2
4
2
5
1
2
3
2
4
5
And so on…..
Register 1
Register 2
Combinational unit, see next slide

Pipelined sorter in which the first register is changed to two registers is a merger. This is not the best solution but it satisfies the question. This approach does not require registers 1 and 2 to be initially sorted. A more efficient parallel processor for merging can be found in lecture slides.



Top = Min(a,b) when a≠b , Top = a when a=b

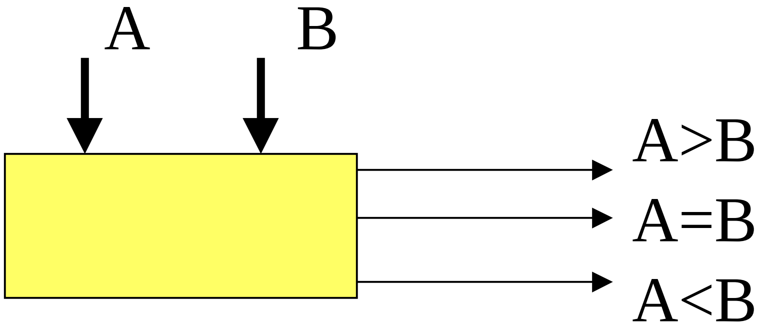
Bottom = Max(a,b) when a≠b, Bottom = E when a=b

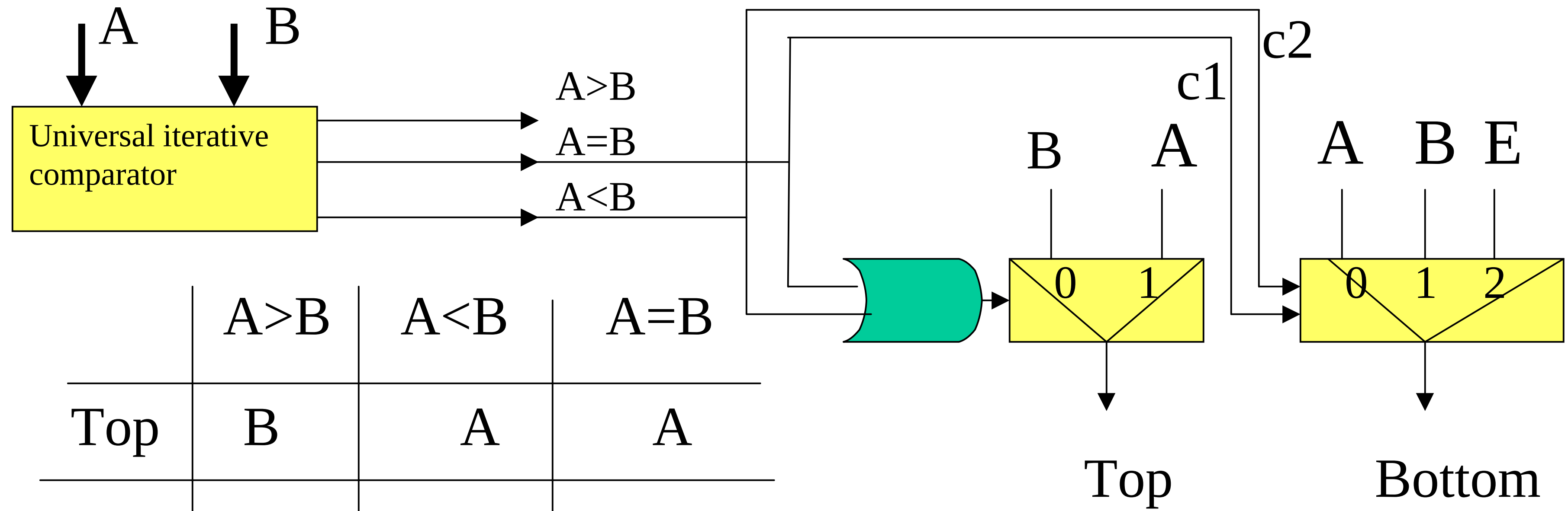
In systematic design, we can create the following table:

	A>B	A<B	A=B
Top	B	A	A
Bottom	A	B	E

Some special string not used in numbers

In class we designed the comparator of order and equality as a combinational unit like this:





	A>B	A<B	A=B
Top	B	A	A
Bottom	A	B	E

Thus we can derive the following:

Top = A when (A=B or A<B) , B otherwise

Bottom = A when A>B , B when A<B, E when A=B

c1 c2	Data	what selected
0 0	0	A>B
0 1	1	A<B
1 0	2	A=B

Problem 9.

Consider the two-qubit state below:

$$|z\rangle = z_0|00\rangle + z_1|01\rangle + z_2|10\rangle + z_3|11\rangle$$

Determine the probabilities of measuring (in the first qubit) 0 and 1 when this state is measured with:

(a) Bell basis

$$|00\rangle \to \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |01\rangle \to \frac{|00\rangle - |11\rangle}{\sqrt{2}}, |10\rangle \to \frac{|01\rangle + |10\rangle}{\sqrt{2}}, |11\rangle \to \frac{|01\rangle - |10}{\sqrt{2}}$$

(b) Magic basis

$$|00\rangle \to \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |01\rangle \to \frac{i(|00\rangle - |11\rangle)}{\sqrt{2}}, |10\rangle \to \frac{i(|01\rangle + |10\rangle)}{\sqrt{2}}, |11\rangle \to \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

10 points

Solution to Problem 9

Bell basis

$$|z\rangle = z_0\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) + z_1\left(\frac{|00\rangle - |11\rangle}{\sqrt{2}}\right) + z_2\left(\frac{|01\rangle + |10\rangle}{\sqrt{2}}\right) + z_3\left(\frac{|01\rangle - |10\rangle}{\sqrt{2}}\right)$$

$$|z\rangle = \left(\frac{z_0 + z_1}{\sqrt{2}}\right)|00\rangle + \left(\frac{z_2 + z_3}{\sqrt{2}}\right)|01\rangle + \left(\frac{z_2 - z_3}{\sqrt{2}}\right)|10\rangle + \left(\frac{z_0 - z_1}{\sqrt{2}}\right)|11\rangle$$

Probability of measuring 0 is

$$p_0 = \sqrt{\left|\frac{z_0 + z_1}{2}\right|^2 + \left|\frac{z_2 + z_3}{2}\right|^2}$$

Probability of measuring 1 is

$$p_1 = \sqrt{\left|\frac{z_2 + z_3}{2}\right|^2 + \left|\frac{z_0 - z_1}{2}\right|^2}$$

Magic basis

$$\left|z\right\rangle = z_0\left(\frac{\left|00\right\rangle + \left|11\right\rangle}{\sqrt{2}}\right) + z_1\left(\frac{i\left|00\right\rangle - i\left|11\right\rangle}{\sqrt{2}}\right) + z_2\left(\frac{i\left|01\right\rangle + i\left|10\right\rangle}{\sqrt{2}}\right) + z_3\left(\frac{\left|01\right\rangle - \left|10\right\rangle}{\sqrt{2}}\right)$$

$$\left|z\right\rangle = \left(\frac{z_0 + iz_1}{\sqrt{2}}\right)\left|00\right\rangle + \left(\frac{z_3 + iz_2}{\sqrt{2}}\right)\left|01\right\rangle + \left(\frac{iz_2 - z_3}{\sqrt{2}}\right)\left|10\right\rangle + \left(\frac{z_0 - iz_1}{\sqrt{2}}\right)\left|11\right\rangle$$

Probability of measuring 0 is

$$p_0 = \sqrt{\left|\frac{z_0 + iz_1}{2}\right|^2 + \left|\frac{iz_2 + z_3}{2}\right|^2}$$

Probability of measuring 1 is

$$p_1 = \sqrt{\left|\frac{iz_2 - z_3}{2}\right|^2 + \left|\frac{z_0 - iz_1}{2}\right|^2}$$

Problem 10.

Prove that the eigenvalues of a Hermitian operator are real

10 points

Solution to Problem 10

Prove that the eigenvalues of a Hermitian operator are real

Proof:

By definition

$$H|z\rangle = \lambda|z\rangle \qquad \langle z|H^\dagger = \lambda^*\langle z|$$

H is Hermitian. So $H^\dagger = H$

$$\langle z|H^\dagger H|z\rangle = \lambda^*\lambda\langle z|z\rangle = \langle z|H^2|z\rangle = \lambda^2\langle z|z\rangle$$

$$\lambda^* = \lambda$$

Additional Problems.

Problem 11.

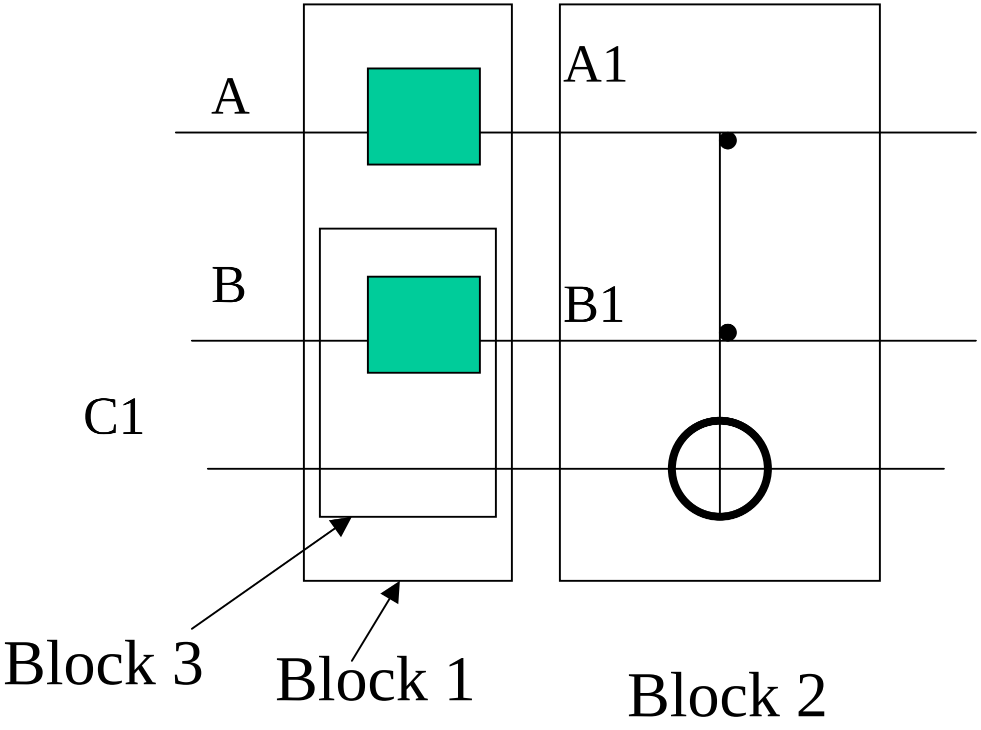
- Design a processor using standard registers and binary technology that simulates (with some accuracy) the operation of a quantum Toffoli gate (with outputs A1. B1, A1*B1 XOR C1) that has quantum Hadamard gates connected to its inputs A and B (A1=Hadamard (A), B1 = Hadamard (B).

Think about the concept of Kronecker Product and Matrix Product. Draw the block diagram only, detailed design is not expected. Operate only on integer approximations of quantum bits in Hilbert space, do not design circuits approximating measurement. Recall butterflies. The circuit may be pipelined or only combinational.

20 points

Solution to Problem 11.

According to the method from the class, we represent the circuit as a pipelined circuit with serial connection of processors corresponding to blocks. To build each processor for a block, we need to find the Kronecker Product of matrices of its all gates (sub-blocks).



Our circuit is a serial connection of Block 1 and Block 2.

Block 1 is a parallel connection of Hadamard, Hadamard and wire

Block 2 is a Toffoli gate

To find the Unitary Matrix of block 1 we need to find a Kronecker Product of matrices H, H and W

Calculate
for block 3

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

Calculate
for block 1

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{pmatrix}$$

- Controlled CNOT (C^2NOT or Toffoli gate)

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

The next slide show how to build the data paths of the processor and illustrates the tremendous parallelism of quantum operations

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

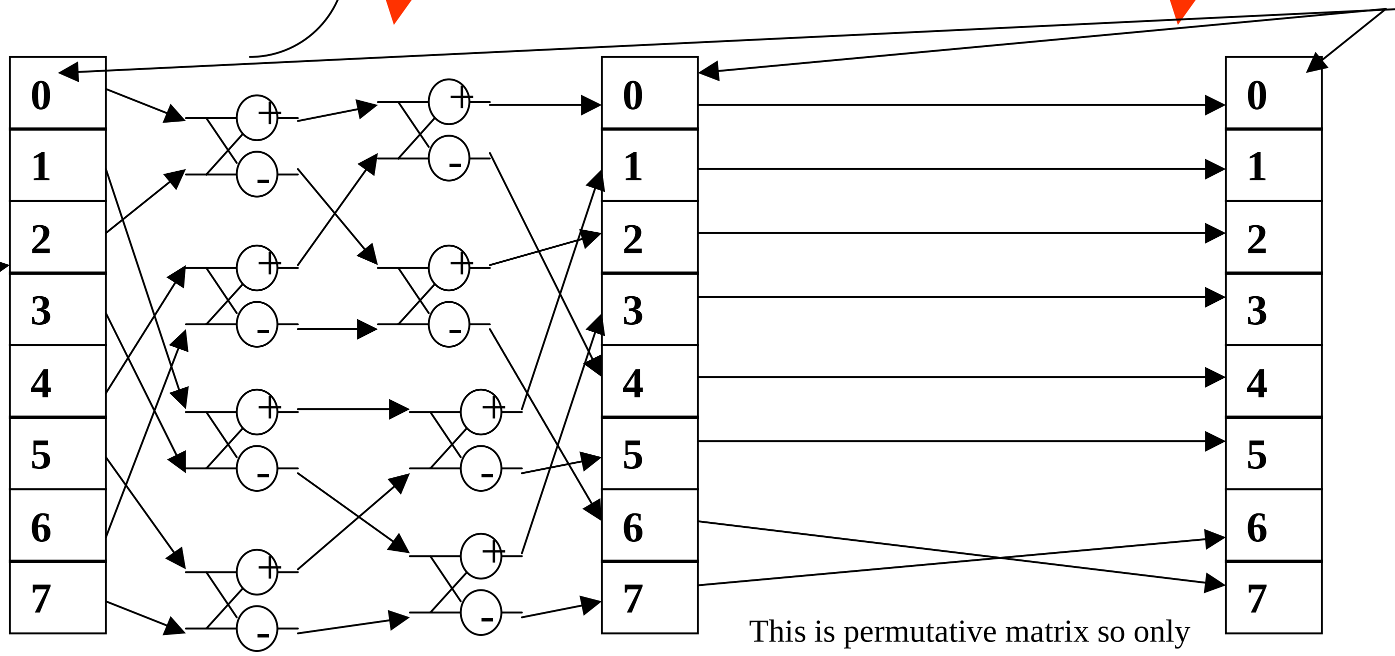
By finding equations and factorizing to Butterfly

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

Now we can draw data path registers, logic between them from the matrices. All values are integers, not complex which simplifies additions and subtractions.

Pipelined registers

Registers correspond to quantum states of a 3-qubit system



Various butterfly types are possible and the data flow structure should be optimized

Combinational adders and subtractors - Walsh Kernels as in DSP

This is permutative matrix so only permutations of register transfers exist.

Problem 12. Generalized Hadamard gates.

- You remember the unitary matrix of a Hadamard gate. Assuming that you can have entries +1,-1,+i and -i , generate all new types of gates that generalize Hadamard gate and can be realized (in principle) in a quantum circuit. You do not have to generate them all.
- Discuss serial and parallel connection of such gates. What will be their unitary matrices?
- What are potential applications of such new gates?

20 points

Solution to Problem 12.

Standard Hadamard Gate has this unitary Matrix (I will not write 1/SQ_ROOT(2) for simplifications

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Let us first investigate small changes. Generalized Hadamard Gates with various positions of -1 are:

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$$

All three new gates are unitary and here you have their inverses

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

We see that multiplying gates by themselves we do not get anything interesting. But we can try a gate by another gate from this family.

Solution to Problem 12.

Thus let us verify some serial connections of these gates:

$$\boxed{\begin{matrix} 1 & 1 \\ 1 & -1 \end{matrix}} * \boxed{\begin{matrix} -1 & 1 \\ 1 & 1 \end{matrix}} = \boxed{\begin{matrix} 0 & 2 \\ -2 & 0 \end{matrix}}$$

Thus we obtained a new gate that is like an inverter for |0> but additionally changes sign for |1>

$$\boxed{\begin{matrix} -1 & 1 \\ 1 & 1 \end{matrix}} * \boxed{\begin{matrix} 1 & 1 \\ -1 & 1 \end{matrix}} = \boxed{\begin{matrix} -2 & 0 \\ 0 & 2 \end{matrix}}$$

A similar gate was obtained

Now let us calculate the Kronecker Product of new matrices.

$$\boxed{\begin{matrix} 1 & 1 \\ -1 & 1 \end{matrix}} \otimes \boxed{\begin{matrix} 1 & -1 \\ 1 & 1 \end{matrix}} = \boxed{\begin{matrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{matrix}}$$

As we see, we generate the same Walsh functions but in different orders. We can also generate negations of Walsh functions.

Solution to Problem 12.

Let us check now how introducing imaginary unit i helps. We multiply rows or columns by i

$$\boxed{\begin{matrix} i & i \\ 1 & -1 \end{matrix}} \qquad \boxed{\begin{matrix} i & 1 \\ i & -1 \end{matrix}} \qquad \boxed{\begin{matrix} i & i \\ i & -i \end{matrix}}$$

We can check that from definition such matrices are unitary. Next we can multiply these new matrices serially and find out that we get new matrices = gates.

$$\boxed{\begin{matrix} 1 & 1 \\ i & -i \end{matrix}} * \boxed{\begin{matrix} i & i \\ 1 & -1 \end{matrix}} = \boxed{\begin{matrix} i+1 & i-1 \\ -1-i & -1+i \end{matrix}}$$

We obtained an interesting new gate

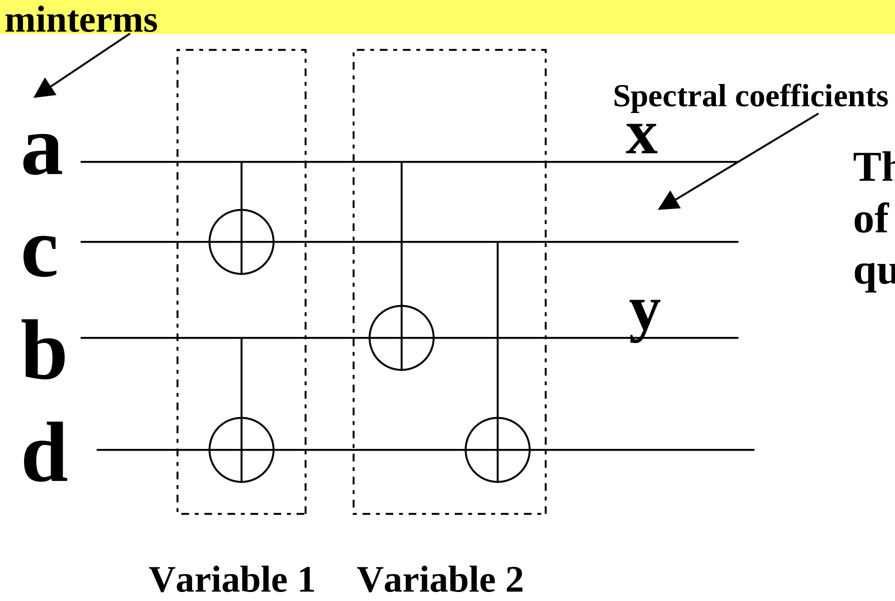
Similarly you can check that Kronecker product of new gates lead to interesting Hadamard-like matrices that have in addition to subtractions and additions also phase changes. The new gates can be used to create generalizations of Hadamard-Walsh transform, which is very important because of the central point of transforms in quantum computing.

Problem 13. RM pipeline butterfly of all FPRM polarities, store cost compare store solution – polarity- reconfigurable pipeline

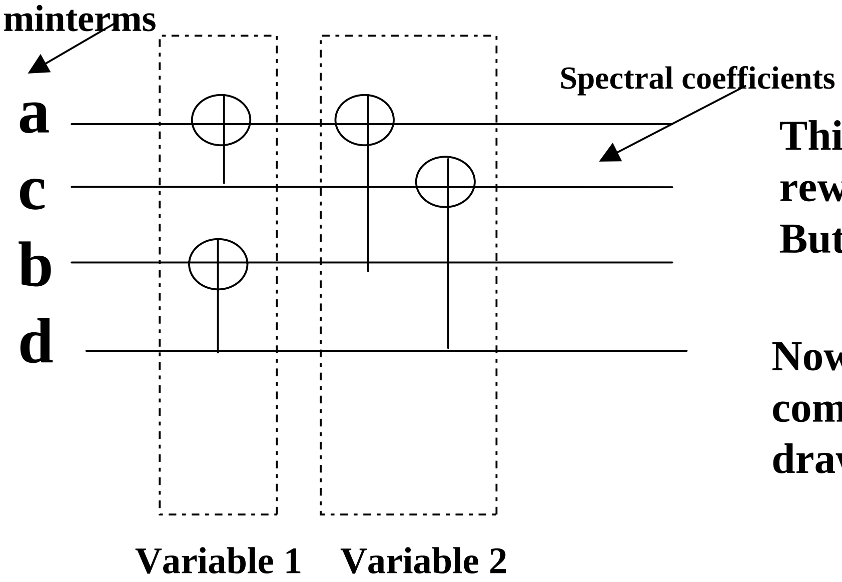
- We discussed the Reed-Muller Transform and circuits. Next we generalized it to Fixed Polarity Reed-Muller (FPRM) and we showed how to expand a function in a form of vector of minterms to any of 2^n polarities of FPRM.
- Draw a quantum circuit for controlled Fixed-Polarity Reed-Muller Transform for two variables.
- Assuming that you can arbitrarily change input and output orders, can you realize the FPRM circuit without swap gates and without garbages?
- Can you realize a circuit with swaps but without garbages?

20 points

Solution to Problem 13.



This quantum circuit is obtained by direct rewriting of the Positive Polarity Reed-Muller Butterfly to quantum notation.



This quantum circuit is obtained by direct rewriting of the Negative Polarity Reed-Muller Butterfly to quantum notation.

Now we jest need to add control variables to design a combined circuit. This method is much simpler than by drawing Kmaps.

Solution to Problem 13.

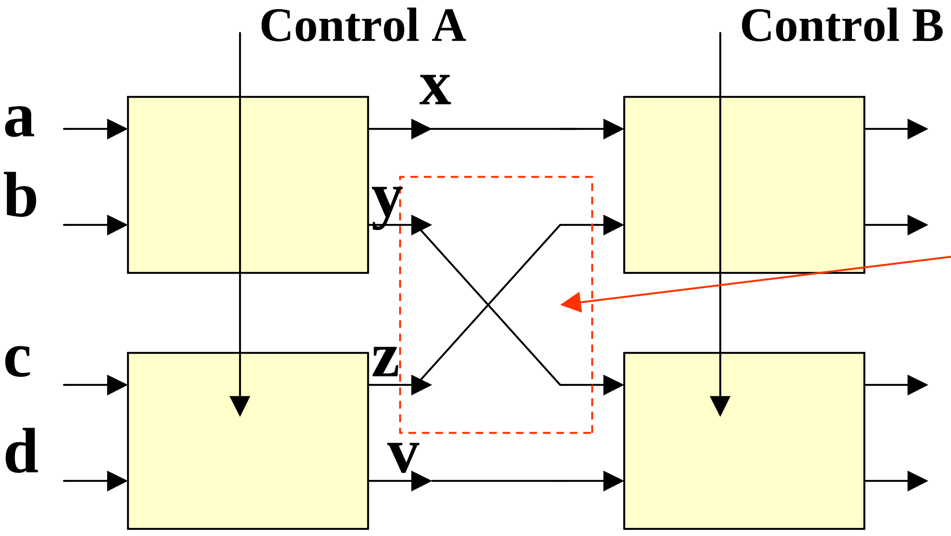
Let us denote the control variables by capital letters. Let us design the gate with inputs a,b and outputs x,y

$$x= A'a \oplus A(a\oplus b)=A'a \oplus Aa \oplus Ab = a \oplus Ab$$

$$y= Ab \oplus A'(a\oplus b)=Ab \oplus A'a \oplus A'b = b \oplus A'a$$

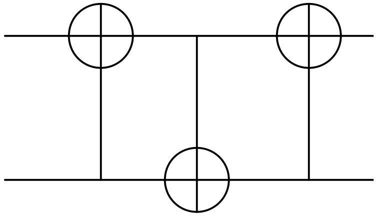
Hence we create a circuit for kernel, which we next repeat in vertical and horizontal directions in standard way to create a controlled butterfly.

A controls polarity of variable 1 in transform, B controls polarity of variable 2. Control lines are shown in red.



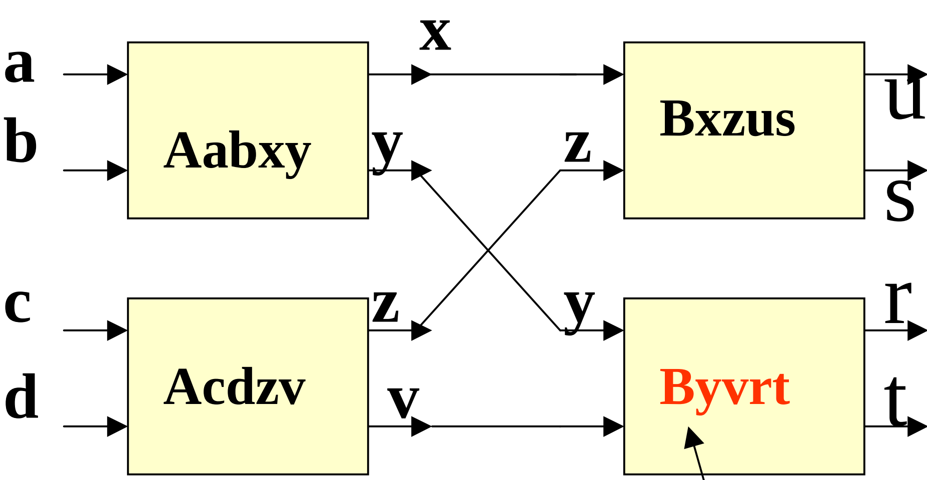
In this solution, each gate is the same and one needs a swap gate

This figure shows that garbages and input constants are not necessary if youallow for swap gates without changing the order.

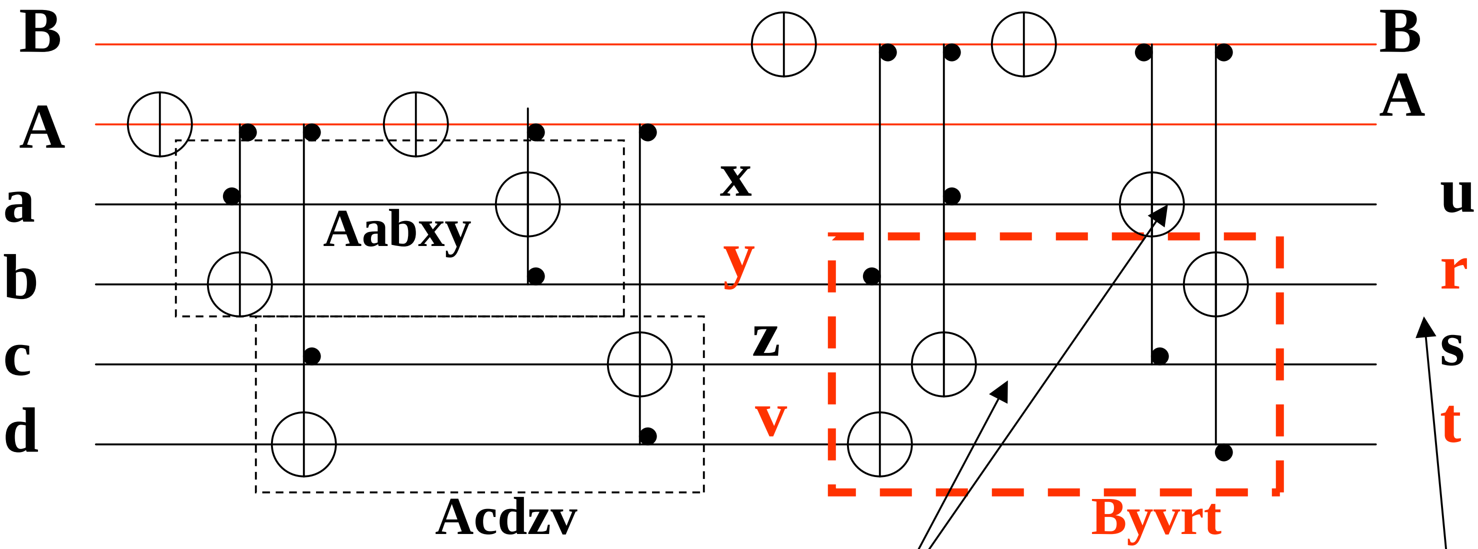


Realization of swap gate

To avoid swapping, you can change the order of inputs or outputs, but this is is not a general solution always. Sometimes order is important and you cannot change it. Here we change output order.



Names of variables in a pattern of gate



This figure shows that garbages and input constants are not necessary if you change order.

This belongs to gate Bxzus

Observe order change

Problem 14. State Moore's Law. What impact will it have on Quantum Computing?

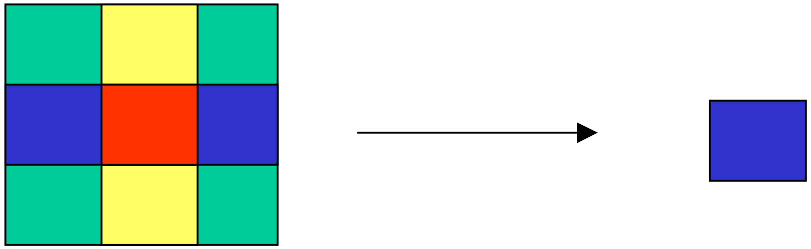
10 points

Solution to Problem 14.

- The computer power for constant cost doubles every 18 months. It cannot continue forever since the atomic limits will be reached between 2012-2020. The fabrication costs cannot keep increasing as until now and the generated heat per size unit also must be reduced. So many fundamental changes are expected and new paradigms such as quantum computing must be used. Concluding, quantum computing is a necessity if we want Moore’s Law to continue to hold.

Problem 15. Cellular Automaton for arbitrary rules in neighborhood 3*3 window with colors

- Given is a set of 4 rules in a 3*3 neighborhood. Each cell has a color, one of four. Each rule has a precedent being a pattern of colors in the neighborhood. It decides the color of the middle block, for instance, Rule 1 is the following:



Encode red by 00, blue by 01, green by 10 and yellow by 11.

This rule for color blue must be satisfied, but there may be more transitions to blue. Formulate three other rules for transitions to other colors and design a logic for a single cell processor.

The behavior for all other situations than the one shown above is arbitrary, but for every color the must be a situation to transit to it.

There can be many rules to transit to any color. Design a simple circuit.

15 points

Solution to Problem 15.

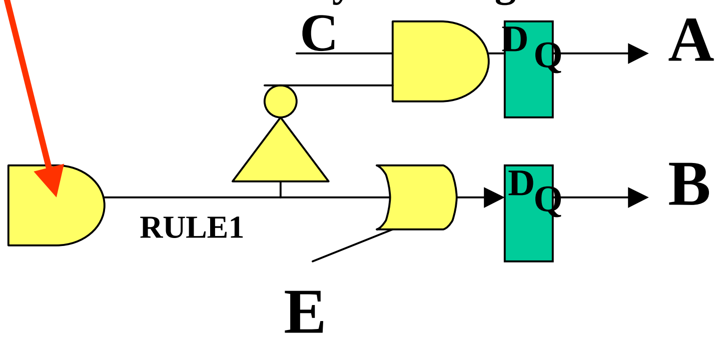
This problem is a little tricky. It has very many solutions so you want to find an easy one.

1	2	3
4	5	6
7	8	9

We encode the processors in neighborhood like this. Each processor has two flip-flops to store the color. The ffs are A and B. The wires from the processors coming to our processor 5 are then the following: 1A,1B, 2A, 2B, 3A, 3B,…. ,9A,9B. Thus the function exciting FF B in our processor is :

RULE1=1A (1B)’ 2A 2B 3A (3B)’ (4A)’ 4B (5A)’(5B)’(6A)’6B 7A (7B)’ 8A 8B 9A (9B)’

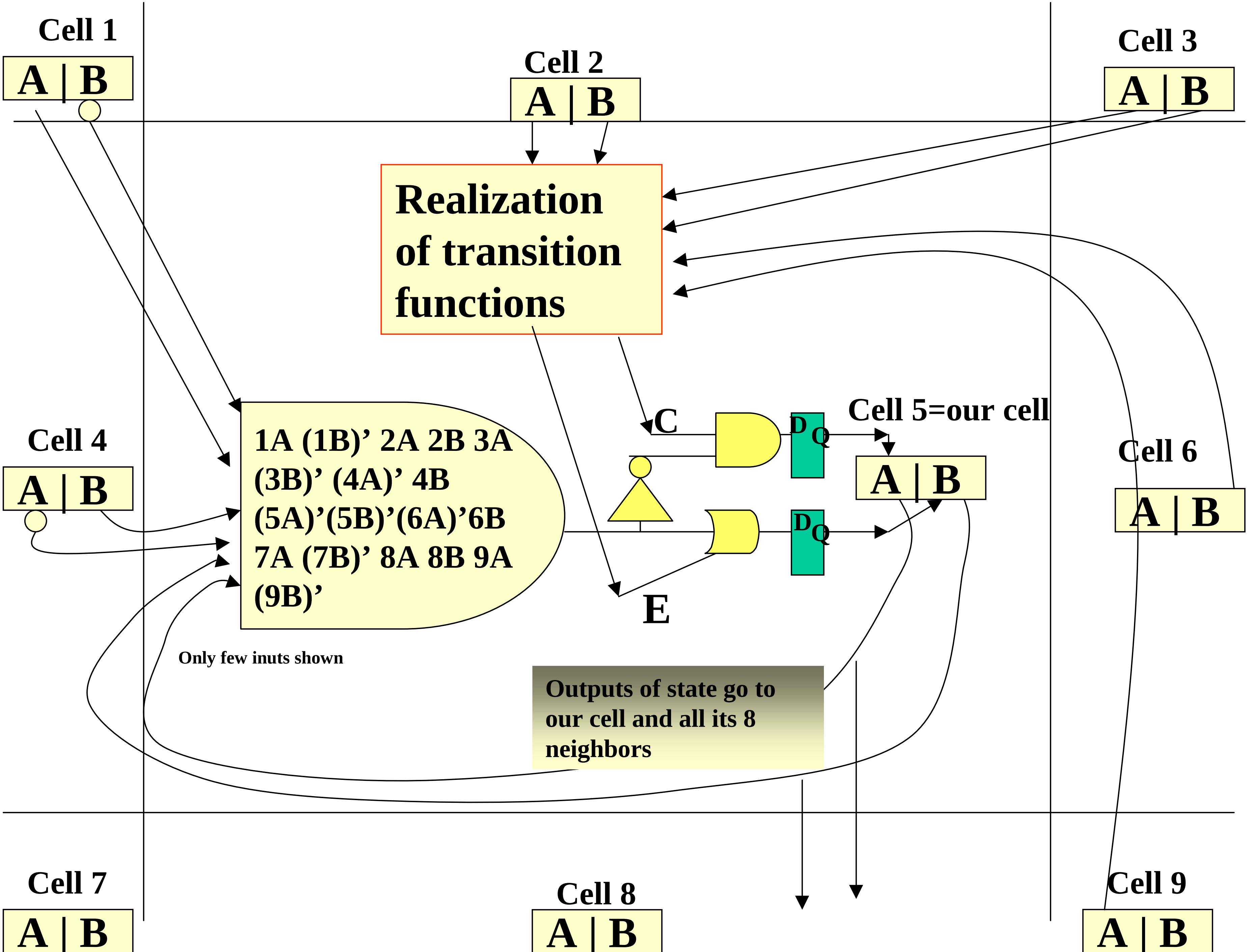
For the above minterm the excitation of FF A of our processor must be zero and FF B = 1. This condition is satisfied by building this circuit:



C and E can be now arbitrary signals to excite FFs A and B. It is up to you to select these signal values and reasoning backwards finding the colors, their encodings and neighbor cell colors.

Execution of RULE 1 is enforced regardless other signals when signal RULE1=1

Rule1	C	E	A B	COLOR
0	0	0	00	red
1	0	1	01	blue (all transitions to Rule)
0	1	0	10	green
0	1	1	11	yellow



Realization of transition functions

1A (1B)’ 2A 2B 3A (3B)’ (4A)’ 4B (5A)’(5B)’(6A)’6B 7A (7B)’ 8A 8B 9A (9B)’

Only few inuts shown

Outputs of state go to our cell and all its 8 neighbors

Problem 16. What are the essential differences between classical and quantum mechanics?

5 points

Solution to Problem 16.

- In classical mechanics states are distinguishable. Any two states are either the same or different. In quantum mechanics there are pairs of states that are mathematically distinct but are not physically distinguishable. They cannot be reliably distinguished by any measurement, no matter how precise. Also, the state depends on observation, which is not true in classical mechanics.
- In classical physics the fundamental properties of an object such as energy, position and velocity are directly accessible to observation. In QM these quantities no longer appear as fundamental, being replaced by the state vector, which cannot be directly observed or measured. It is as though there is a hidden world in quantum mechanics, which we can only indirectly and imperfectly access. Moreover , merely observing a classical system does not change the state of the system.

Problem 17. What is the superposition principle in QM?

5 points

Solution to Problem 17.

- If $|x>$ and $|y>$ are two states of a quantum system then the superposition $a|x> +b|y>$ where a and b are complex numbers is also allowed as state of a quantum system. However, it must be satisfied that $|a|^2+|b|^2=1$

Problem 18. What is nondeterministic about QM?

5 points

Solution to Problem 18.

When the unitary quantum gate, like Hadamard is not a permutative gate, and it is observed, it creates values one or zero with certain probabilities. This property is of course true for complete quantum circuits. Thus, in some situations, the quantum circuit gives probabilistic responses. This non-determinism is the very principle of Quantum Computer work. Quantum Circuit can simulate non-deterministic Finite State Machine, Non-deterministic Turing Machine, probabilistic machine, relational system or any concept that has non-deterministic behavior.

Problem 19. What was significant about the Stern-Gerlach experiment?

10 points

Solution to Problem 19.

This was an experiment conceived by Stern in 1921 and performed with Gerlach in 1922 which gave first experimental evidence that qubits exist in Nature. See Figure 1 below for explanation. In original experiment the hot atoms of silver were beamed from an oven through a deflecting magnetic field and the position of each atom was recorded. For simplification we present the experiment using hydrogen atoms. Such atoms have a proton and an orbiting electron. The orbiting electron is like an electric current so the atom has a magnetic field. Thus the atoms should be deflected in the magnetic field. How much the atom is deflected depends on its dipole moment and on the magnetic field of Stern-Gerlach device which we can control. We can cause the atom to be deflected by an amount that depends on z component of the dipole moment, where z is some fixed external axis. It was observed that we do not get continuous distribution of atoms at different angles but discrete angles. This was explained by quantization of dipole moments. Even more surprising was the number of peaks seen. Since hydrogen has zero magnetic dipole moment a single beam was expected but two were seen. This phenomenon was explained by introducing the concept of spin of electron. This spin is an extra contribution to dipole moment.

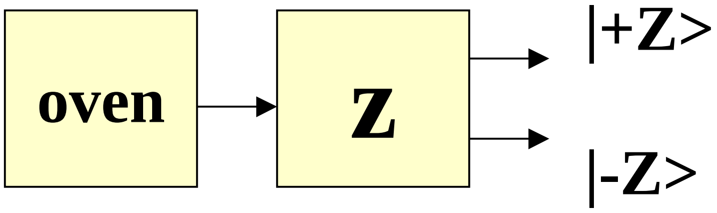
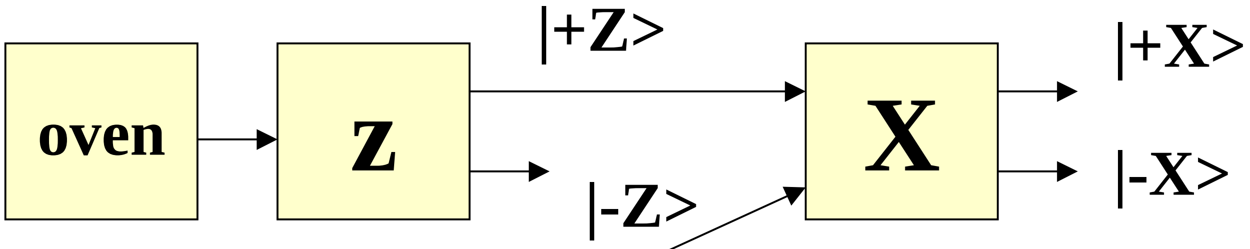


Figure 1: |+Z> and |-Z> denote deflection up and down, respectively

Figure 2: Cascaded Stern-Gerlach measurements



If we cascade two Stern-Gerlach devices together as in Figure 2, to further investigate the nature of qubits, but the second apparatus is tipped sideways so that magnetic field deflects atoms along the x axis. A classical magnetic dipole pointed in z direction has no magnetic moment in z direction so one peak was expected but again two peaks of beam were observed. Therefore it was hypothesized that each atom passing through the second device was in a state

|+Z> |+X> **or** |+Z>|-X>**. Next experiments of this type confirmed the model.**

Problem 20. What is a Hilbert Space?

5 points

Solution to Problem 20.

- Hilbert space is a vector space in which the scalars are complex numbers, with an inner product operation ·

  x · y=(y · x)*

$$x \bullet x \geq 0$$

$$x \bullet x = 0 \quad iff \quad x = 0$$

$$x \bullet y \quad is \quad linear, \textit{ under scalar multiplication and vector addition with both x and y}$$

Problem 21. What are bras and kets?

5 points

Solution to Problem 21.

- The inner product definition is the same as the matrix product of x as a conjugated row vector, times y as a normal column vector.

$$\langle x | y \rangle = \sum_i x_i^* y_i = [x_1^* x_2^* \ldots] \begin{bmatrix} y_1 \\ y_2 \\ \\ \\ \end{bmatrix} \quad = <s|s>$$

This leads to the definition for state S:

"bra" <s| is the row matrix as above and "ket" |s> means the column matrix as above.

Problem 22. What is an adjoint operator?

5 points

Solution to Problem 22.

Let A be a linear operator on Hilbert space V. There exist exactly one operator A^+ on V such that for all vectors |v>, |w> from V it holds that: (|v>, A|w>) = (A^+|v>, |w>)

This operator is called Adjoint of operator A, it is also called Hermitian conjugate of operator A. You first calculate the transpose matrix of A and next find its conjugate to create a Hermitian of A.

$$\begin{pmatrix} 1+3i & 2i \\ 1+i & 1-4i \end{pmatrix}^{+} = \begin{pmatrix} 1-3i & 1-i \\ -2i & 1+4i \end{pmatrix}$$

Matrix of adjoint of operator A

Matrix of operator A

Problem 23.

- Explain how the DNA computer solves the Hamiltonian Path Problem. Do not go to chemical details -- only the main idea.
- Design a massively parallel cellular, systolic or pipelined processor to solve the Hamiltonian Path problem. Is there any link with the DNA algorithm?

__20 points__

Solution to Problem 23.

- Explain how the DNA computer solves the Hamiltonian Path Problem. Do not go to chemical details -- only the main idea. Lego block corresponds to a DNA strand.

Given is graph G with n nodes. We have to find all paths from starting node A to the terminal node Z of graph G in such a way that every node is encountered and only once.

Observe that the following conditions are imposed on the set of all paths:

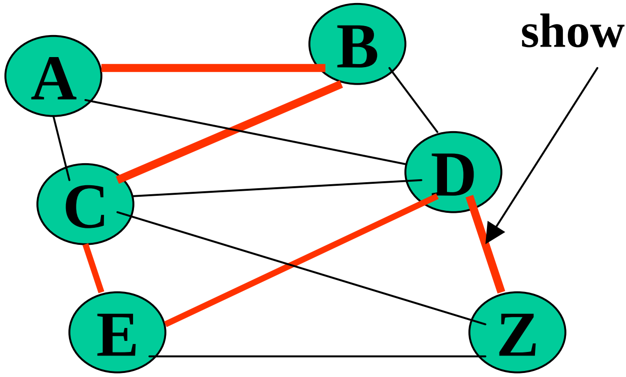
C1. It starts from A.

C2. It terminates with Z.

C3. It goes through n nodes.

C4. Nodes in the path are not repeated.

C5. Every node is in the path.



One solution shown in red

You can think that you are giving orders to unlimited number of totally obedient children who play with Lego blocks. Each sequence of two nodes is a Lego block, like AB, it is known which is the first and the last in the pair.

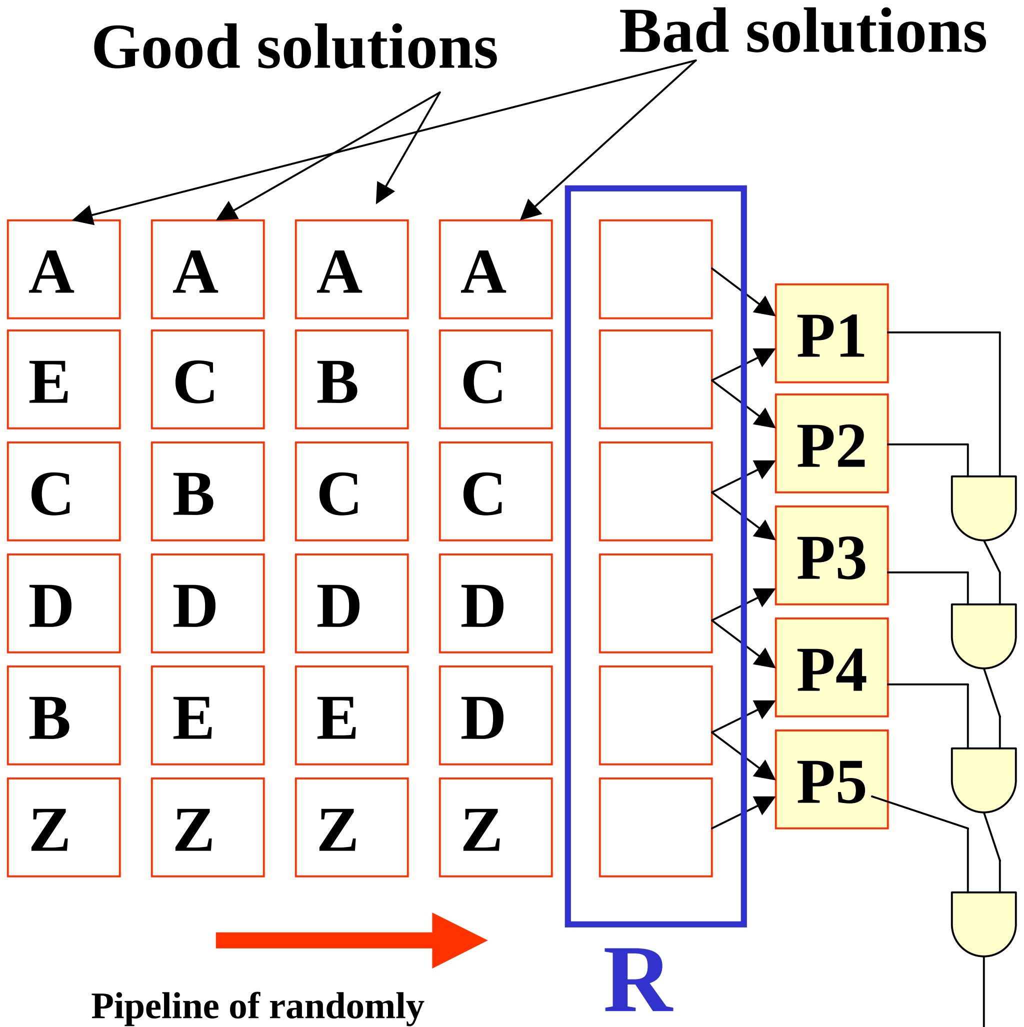
A B | C D | E Z

B C | D E

solution

Solution to Problem 23. (continued)

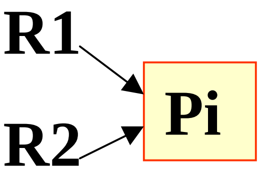
- Design a massively parallel cellular, systolic or pipelined processor to solve the Hamiltonian Path problem. Is there any link with the DNA algorithm?
- The algorithm above is the abstraction of the massively parallel DNA algorithm in which instead of DNA-related chemical operations one gives commands to children playing with Lego blocks.
- Of course, the main idea here is massive parallelism and sequential filtering of candidate solutions, and not use of DNA or children and Lego blocks. Therefore, your creative task is to assume that processor is very inexpensive and to recreate this algorithm.
- The main property of the DNA algorithm is that processors can flow freely, there is not strict connection structure. This is specific only to biomolecular computing.
- We apply a parallel solution in which there is a sequence of registers of length k on which the number of k-1 processors work in parallel. Practically, k<n, but in principle k can be also equal n assuming massive parallelism. We assume here that k=n



A	A	A	A
E	C	B	C
C	B	C	C
D	D	D	D
B	E	E	D
Z	Z	Z	Z

Pipeline of randomly generated sequences of length n that start with A and terminate with Z

If 1 then R is a good solution



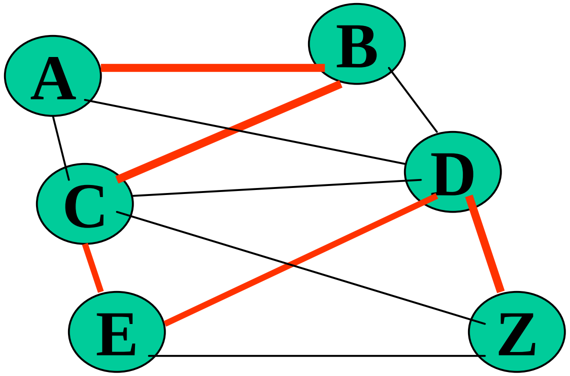
Algorithm of Pi

1. Find set S(R1$_i$) of rules which have the rule predecessor being the contents of R1$_i$. (*these rules are stored only by their second node, because the first is always R1$_i$*).

2. Output := (If R2$_i$ ∈ S(R1$_i$) then 1, else 0)

Rules

AB,AC,AD,BC,BD,CB,CD,CE,CZ,
DB,DC,DZ,DE,EC,ED,EZ



Representation of rules for the algorithm from previous slide:

A - {B,C,D}

B - {C,D}

C - {B,D,E,Z}

D - {B,C,Z,E}

E - {C,D,Z}

These rules have only successor nodes

Problem 24. Explain in brief the Basic Measurement postulate as applied in quantum computers.

10 points

Solution to Problem 24.

Measurement Postulate is -

- The fundamental paradox of Q-Mechanics

- When an object is ‘measured’, the very ‘act of measurement’ causes the object of take on one of the ‘allowed outcomes’

- The registering of only one quantum state even though super positioning causes several states to exist simultaneously

- Mathematically – if the state of a system is ψ, then the probability that a measurement finds the system in the state Φ_j is $|c_j|^2$, where c_j is the weight of that particular ‘jth’ state

- See solutions to problems 9 and 30 for illustration.

Problem 25. De-coherence and what to do about it.

10 points

Solution to Problem 25.

- Decoherence is the effect that makes macroscopic quantum systems *appear* to behave “classically.” Occurs due to inevitable interactions between a given quantum system & an unknown (high-entropy) environment. The external factors of the surroundings like temperature, stray radiations, etc have their influence. Once a quantum system becomes entangled with its surrounding environment (or a macroscopic measuring device) it is no longer isolated and the fragile quantum superpositions are lost. A quantum superposition state gradually “collapses” or “decays” to a classical statistical mixture of the pointer states (measurement eigenstates). This is a real physical process that takes place very quickly when a quantum system interacts with the classical world.

- **Decoherence is** a possible practical threat to a viable Quantum Computer. The researchers must go to tremendous efforts to prevent the computer from interacting with its environment and decohering from a pristine state of superposition). Loading the input and reading out the output require interaction and thus are problematic. As it is virtually impossible to isolate these systems, decoherence occurs within a few computational steps for all systems studied so far. QCs are prone to many errors because of this problem.
- Additionally there are of course the practical problems of working at single atom and single photon scales. The best solution, so far, to the decoherence problem is the use of a large number of atoms. Instead of trying to isolate one atom from its environment, use instead trillions of atoms so that the background noise averages out. A radio pulse would disturb some of them but the information content of the rest would survive.

Problem 26. What is Quantum Parallelism?

10 points

Solution to Problem 26.

- All possible states can be expressed in a single expression and this states can be calculated at this time by some operation, for example

$$\varphi = \alpha \,|\, 0\rangle + \beta|1\rangle$$

And we apply operation: $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Which leads to new states: $|0\rangle = \alpha + \beta \quad and \quad |1\rangle = \alpha - \beta$

State |0>, state |1> of solution are calculated at the same time, this is parallel processing on micro level. Two operations on complex numbers are executed in one moment. For n qubits 2^n states exist and for each of them a very complex multi-argument operation on complex numbers is executed. This creates unbelievable high parallelism of operations.Think about n Hadamard gates connected in parallel and calculate how many operations are executed in parallel.

Problem 27. What is Quantum entanglement?

10 points

Solution to Problem 27.

- If a state of a compound system C can be expressed as a tensor product of states of two independent subsystems A and B then A and B are not entangled. Otherwise A and B are entangled which means that their states are not independent , for instance **|00> + (-i) |11>**

Problem 28. Hough Transform.

- 1. Design a Hough Transform Processor for straight lines. It can be pipelined, cellular, systolic or parallel. Think about which directions of image or other parameters correspond to time and space in memory (hardware). Use either trigonometric or linear parameterization of lines. Do not show details - only the basic flow diagram and pipeline or cellular automata rules. Assume sparse black and white image.
- 2. Explain how your ideas from point 1 can be generalized to Hough Transform for circles.

20 points

Solution to Problem 28.

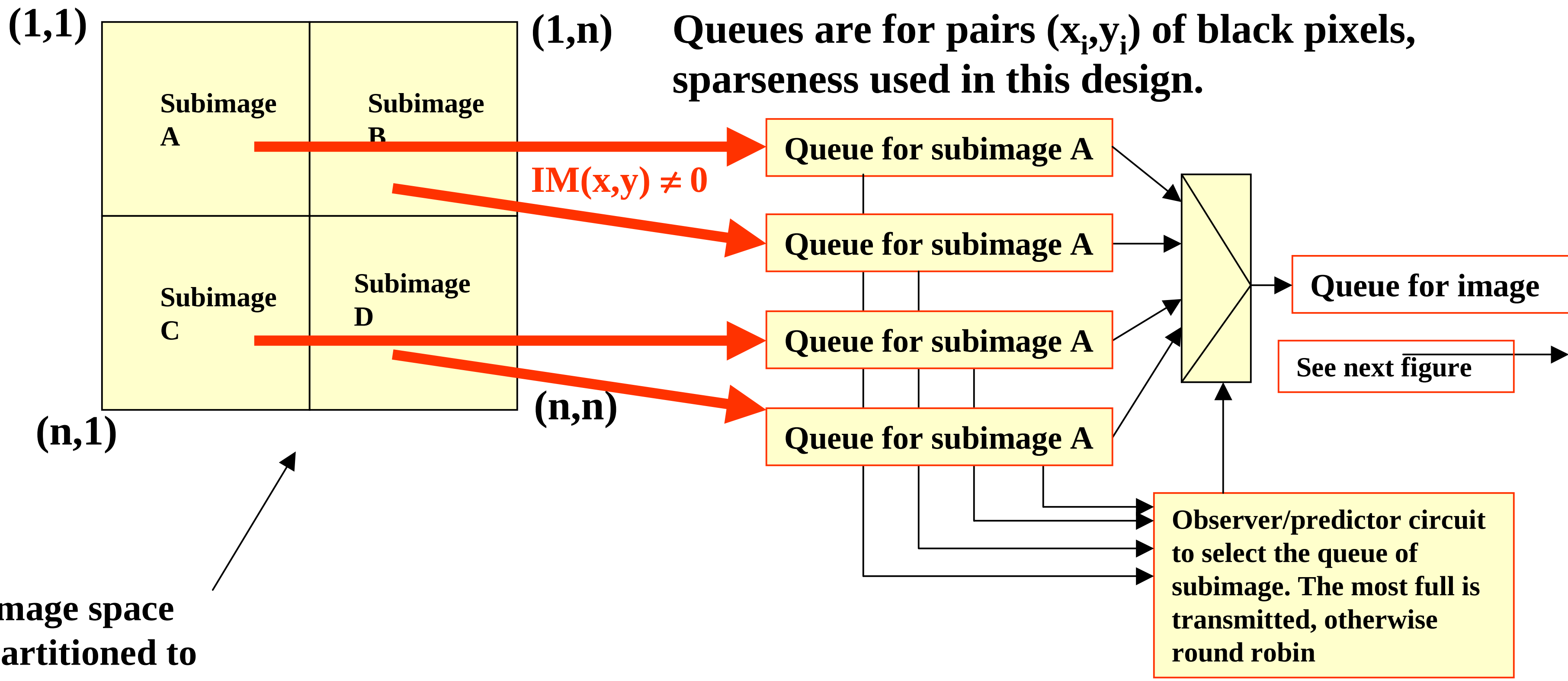
- As presented in class, there are two basic methods to parallelize Hough Transform: in image space and in accumulator (parameter space). We will apply both these methods here for maximal flexibility. The method will be illustrated for linear equations of lines, but analogical method can be applied for trigonometric equations.

```
For x=1 to n do
  For y=1 to n do
    if IM(x,y) $\neq$ 0 then
      For a=0 to m-1 do
        b := y - a*x
        ACC[a,b]=ACC[a,b|+1
```

This is done in image space - Parallelize this.

This is done in parameter space - Parallelize this.

Queues are for pairs (x_i,y_i) of black pixels, sparseness used in this design.



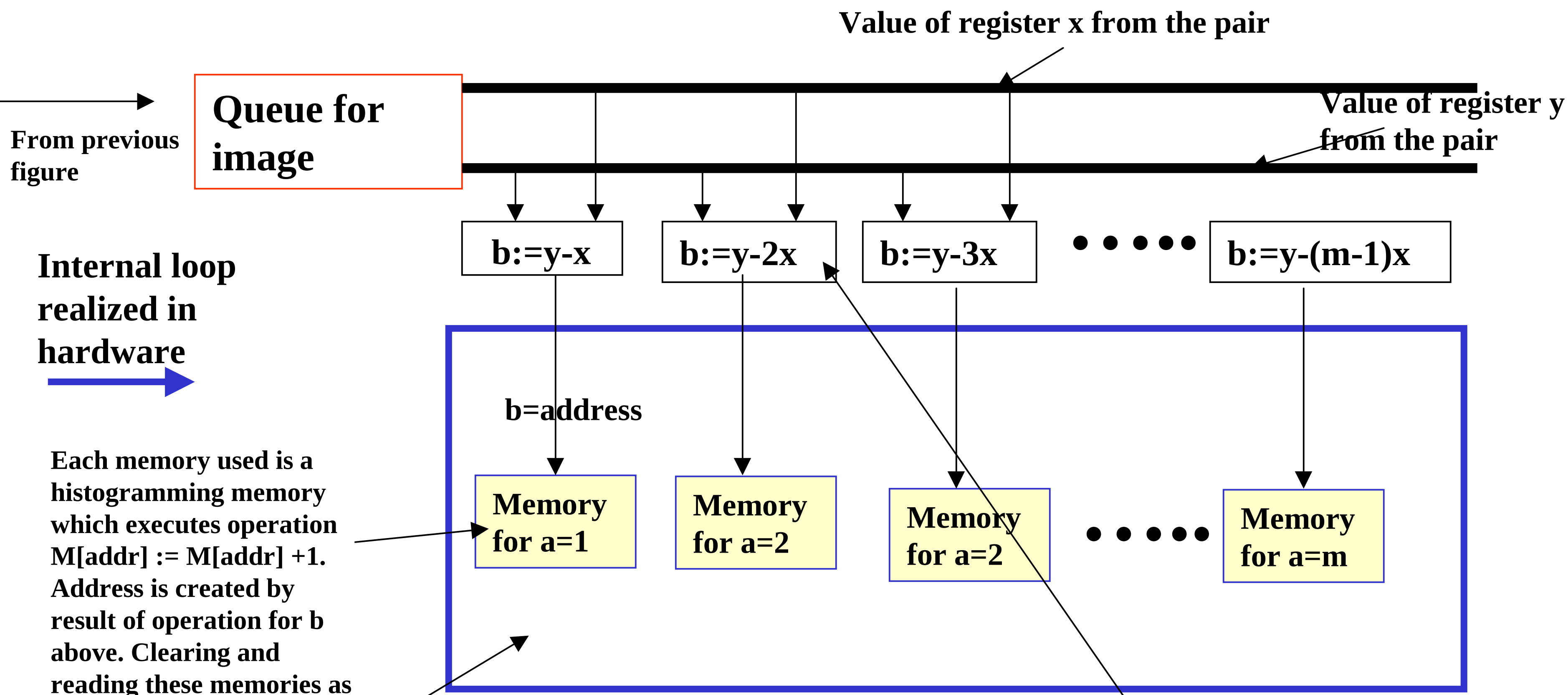
Observer/predictor circuit to select the queue of subimage. The most full is transmitted, otherwise round robin

Image space partitioned to four subspaces for parallelization

```
For x=1 to n do
  For y=1 to n do
     if IM(x,y) ≠ 0 then send (x,y) to queue
        For a=0 to m-1 do
          b := y-ax
          ACC[a,b]=ACC[a,b|+1
```

Repeat this with different starting and ending values, for each of four controllers, keep unique addressing of all array

This figure explains parallelization in the image space



Value of register x from the pair

From previous figure

Queue for image

Value of register y from the pair

b:=y-x

b:=y-2x

b:=y-3x

• • • • •

b:=y-(m-1)x

Internal loop realized in hardware

b=address

Memory for a=1

Memory for a=2

Memory for a=2

• • • • •

Memory for a=m

Each memory used is a histogramming memory which executes operation M[addr] := M[addr] +1. Address is created by result of operation for b above. Clearing and reading these memories as well as details of control are not shown.

```
For x=1 to n do
  For y=1 to n do
    if IM(x,y) ≠ 0 then send (x,y) to queue
        For a=0 to m-1 do
          b := y-ax
          ACC[a,b]=ACC[a,b|+1
```

All these operations done combinationally, including multiplication by constants.

Partitioned memory of the parameter space

This figure explains parallelization in the parameter space

Additional ideas

- 1. Observe that the algorithm does not work well for lines that are close to vertical. The solution is to rotate image 90 degree and repeat algorithm twice, in time or in space.
- 2. The equation for circles is $(x-a)^2 + (y-b)^2 = r^2$ where a is an x-coordinate of center of the circle, b is the y-coordinate of the center and r is a radius. Since every circle is described by three parameters, a, b and r, we need a three-dimensional parameter space. One dimensional memories from previous slide should be then replaces by two-dimensional memories. Of course, the equations that operate on x and y are also different. There are several variants of the solution, but the principle is the same as shown previously.

Problem 29.

1. Is the state $\frac{|00\rangle + |01\rangle}{\sqrt{2}}$ **is entangled? Provide a proof of your answer.**

2. How many possible entangled 2-qubit states are there?

10 points

Solution to Problem 29

1. $$\frac{|00\rangle + |01\rangle}{\sqrt{2}} = |0\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

So the state can be decomposed into a tensor product of single qubit states. Thus it is not entangled

2. Clearly the state $\frac{|10\rangle + |11\rangle}{\sqrt{2}}$ isn't entangled either. Consider the state $\frac{|10\rangle + |01\rangle}{\sqrt{2}}$

$$\begin{aligned} |x\rangle \otimes |y\rangle &= (x_0|0\rangle + x_1|1\rangle) \otimes (y_0|0\rangle + y_1|1\rangle) \\ &= x_0 y_0 |00\rangle + x_0 y_1 |01\rangle + x_1 y_0 |10\rangle + x_1 y_1 |11\rangle \end{aligned}$$

We must have x_0y_0 and x_1y_1 are both zero. However this implies that at least one of the coefficients of |01> or |10> are zero also. So yes the state is entangled. There are an infinite number of possible entangled 2 qubit states as we can create others by introducing complex coefficients

1/🕑(|00>-i|11>)

Problem 30.

Consider the two-qubit state below:

$$|z\rangle = z_0|00\rangle + z_1|01\rangle + z_2|10\rangle + z_3|11\rangle$$

Suppose the first bit is measured and gives |0>.

Prove a formula that gives the probability that first bit is zero.
Find a formula for the post-measurement state vector

10 points

Solution to Problem 30

Write $$\begin{aligned}|z\rangle &= z_0|00\rangle + z_1|01\rangle + z_2|10\rangle + z_3|11\rangle \\ &= u|0\rangle \otimes \left(\frac{z_0}{u}|0\rangle + \frac{z_1}{u}|1\rangle\right) + v|0\rangle \otimes \left(\frac{z_2}{v}|0\rangle + \frac{z_3}{v}|1\rangle\right)\end{aligned}$$

Where $u = \sqrt{|z_0|^2 + |z_1|^2}, v = \sqrt{|z_2|^2 + |z_3|^2}$

Thus measurement of the first bit will with probability $u = \sqrt{|z_0|^2 + |z_1|^2}$ return zero and leave the state

$$|0\rangle \otimes \left(\frac{z_0}{u}|0\rangle + \frac{z_1}{u}|1\rangle\right) = \frac{z_0|00\rangle + z_1|01\rangle}{\sqrt{|z_0|^2 + |z_1|^2}}$$

Problem 31.

Suppose the measurement operator P=|01><01|+|11><11| acts on a general two qubit state

$$|z\rangle = z_0|00\rangle + z_1|01\rangle + z_2|10\rangle + z_3|11\rangle$$

What is the probability of measuring the second qubit to be 1?

10 points

Solution to Problem 31

The probability u is

$$u = \langle z | P | z \rangle = \langle z | P(z_0 |00\rangle + z_1 |01\rangle + z_2 |10\rangle + z_3 |11\rangle)$$

$$= \langle z | (z_1 |01\rangle + z_3 |11\rangle)$$

$$= (z_0^* \langle 00| + z_1^* \langle 01| + z_2^* \langle 10| + z_3^* \langle 11)(| z_1 |01\rangle + z_3 |11\rangle)$$

$$= z_1^* z_1 + z_3^* z_3$$

The post measurement state is the normalised projection of the two-qubit state

$$\hat{P}(z_0 |00\rangle + z_1 |01\rangle + z_2 |10\rangle + z_3 |11\rangle)$$

$$= \frac{z_1 |01\rangle + z_3 |11\rangle}{\sqrt{|z_1|^2 + |z_3|^2}}$$

Problem 32.

Show that the operator $|x><x|$ is Hermitian for any vector $|x>$†

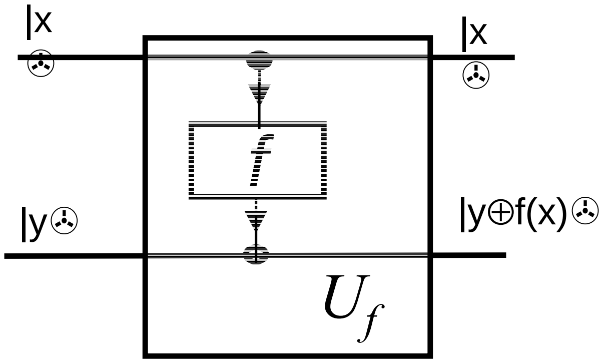
10 points

Solution to Problem 32

$$\left(|x\rangle\langle x|\right)^{\dagger}=\left(\left(|x\rangle\langle x|\right)^{*}\right)^{T}=\left(|x\rangle^{*}\langle x|^{*}\right)^{T}=\left(\langle x|^{*}\right)^{T}\left(|x\rangle^{*}\right)^{T}=|x\rangle\langle x|$$

Problem 33.

- In lecture on Deutsch we used a quantum black box of the form



- Where *f* is a Boolean function of the qubits of |x>. Prove that the transformation U_f is unitary and give an explicit representation of it in matrix form (HINT: think small and work up from there).

10 points

Solution to Problem 33

Assume 2-qubits

$$|0\rangle\otimes|0\rangle=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\rightarrow|0\rangle\otimes|f(0)\rangle=\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}\delta_{f(0),0}\\\delta_{f(0),1}\end{bmatrix}=\begin{bmatrix}\delta_{f(0),0}\\\delta_{f(0),1}\\0\\0\end{bmatrix}$$

$$|0\rangle\otimes|1\rangle=\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\rightarrow|0\rangle\otimes|1\oplus f(0)\rangle=\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}\delta_{f(0),1}\\\delta_{f(0),0}\end{bmatrix}=\begin{bmatrix}\delta_{f(0),1}\\\delta_{f(0),0}\\0\\0\end{bmatrix}$$

$$|1\rangle\otimes|0\rangle=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\rightarrow|1\rangle\otimes|f(1)\rangle=\begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}\delta_{f(1),0}\\\delta_{f(1),1}\end{bmatrix}=\begin{bmatrix}0\\0\\\delta_{f(1),0}\\\delta_{f(1),1}\end{bmatrix}$$

$$|1\rangle\otimes|1\rangle=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\rightarrow|1\rangle\otimes|1\oplus f(1)\rangle=\begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}\delta_{f(1),1}\\\delta_{f(1),0}\end{bmatrix}=\begin{bmatrix}0\\0\\\delta_{f(1),1}\\\delta_{f(1),0}\end{bmatrix}$$

So we get the unitary matrix

$$\begin{bmatrix} \delta_{f(0),0} & \delta_{f(0),1} & 0 & 0 \\ \delta_{f(0),1} & \delta_{f(0),0} & 0 & 0 \\ 0 & 0 & \delta_{f(1),0} & \delta_{f(1),1} \\ 0 & 0 & \delta_{f(1),1} & \delta_{f(1),0} \end{bmatrix}$$

This easily generalises to

$$\begin{bmatrix} \Delta_{f(0)} & 0 & 0 & 0 \\ 0 & \Delta_{f(1)} & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \Delta_{f(n)} \end{bmatrix}, \quad \Delta_{f(n)} = \begin{bmatrix} \delta_{f(n),0} & \delta_{f(n),1} \\ \delta_{f(n),1} & \delta_{f(n),0} \end{bmatrix}$$

for n-qubits

Problem 34.

- Construct a quantum one-bit adder in which inputs x and y are repeated from inputs to outputs.

- Describe how binary addition for an arbitrary number of bits may be carried out with a quantum circuit.

Two-bit adder

a1 a0

b1 b0

$(a+b)_2 \quad (a+b)_1 \quad (a+b)_0$

---

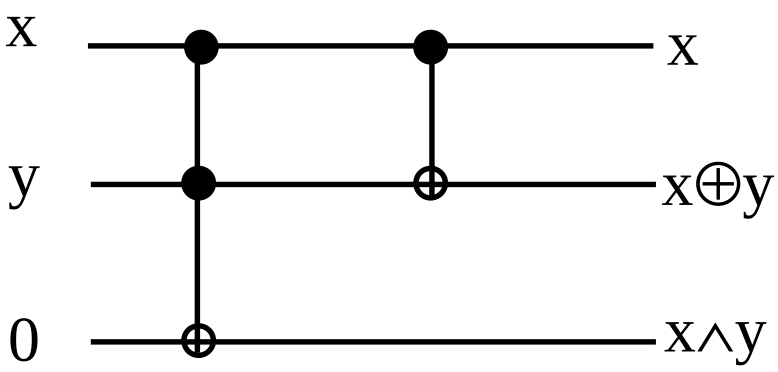
n-bit adder

15 points

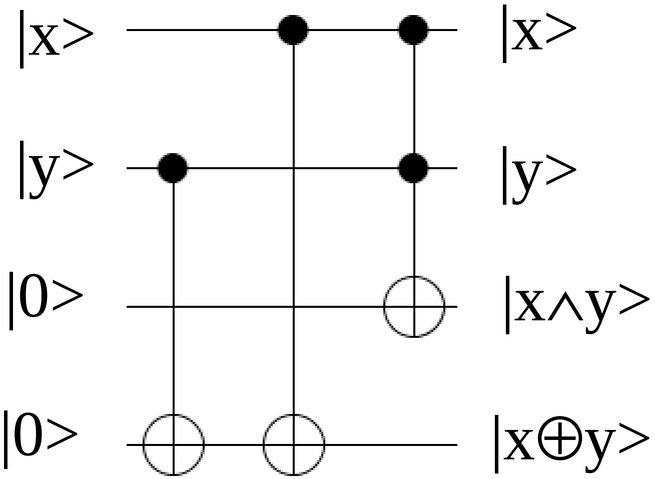
Solution to Problem 34

- Construct a classical reversible one-bit adder and then construct a quantum version

Classical



Quantum

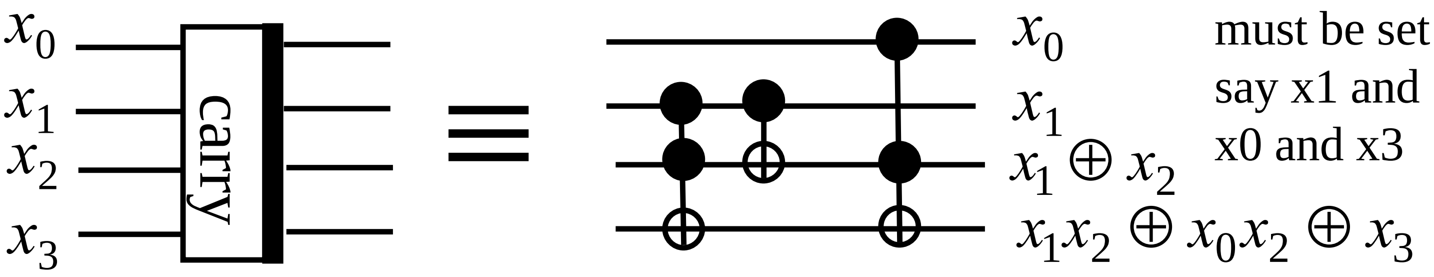


Note: this quantum one-bit adder is more efficient than those shown in next slides and can be used as the basic adder or carry unit

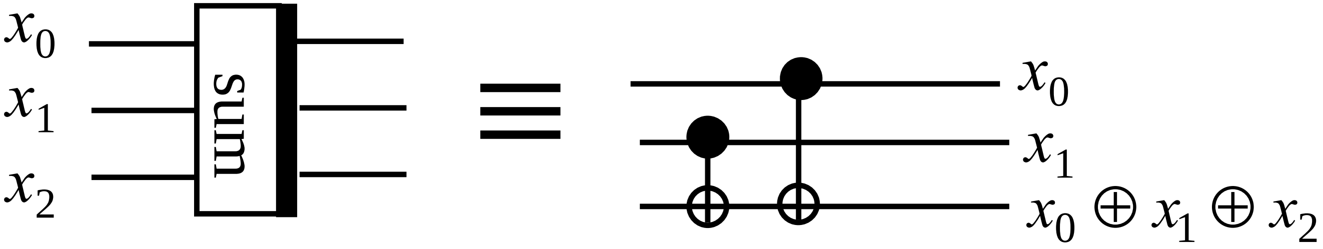
Exercise 3: question and answer

- Describe how binary addition for an arbitrary number of bits may be carried out with a quantum circuit.

(Vedral, Barenco, Ekert 1995 - lanl/quant-ph/9511018



Note: two bits must be set to zero, say x1 and x3 or x0 and x3



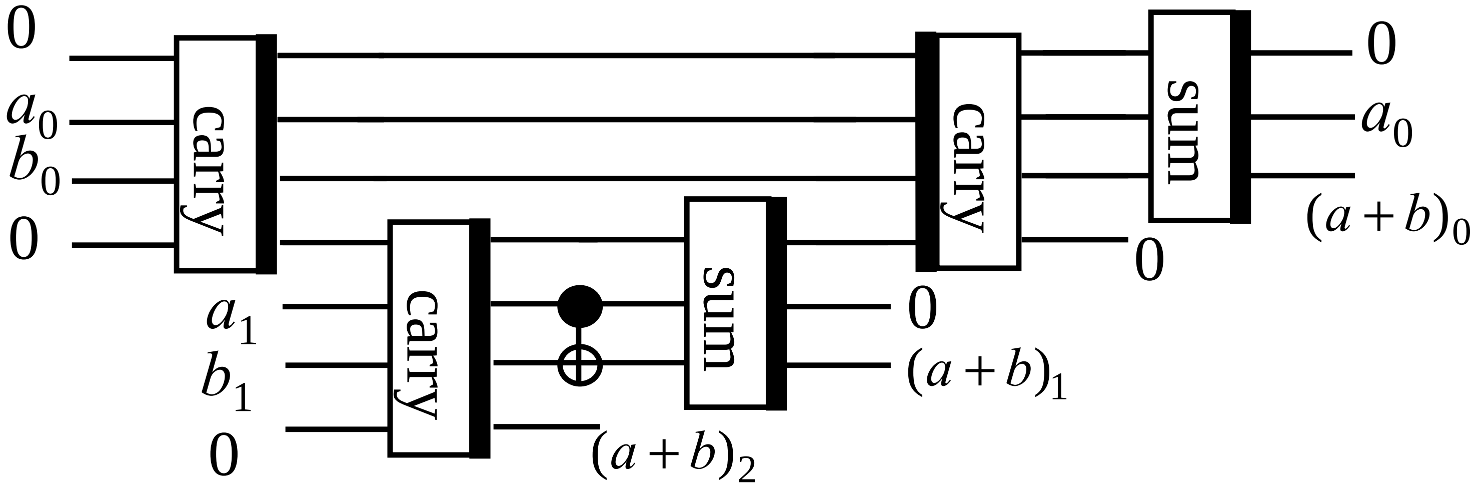
Exercise 3: answer

Two-bit adder

a1 a0

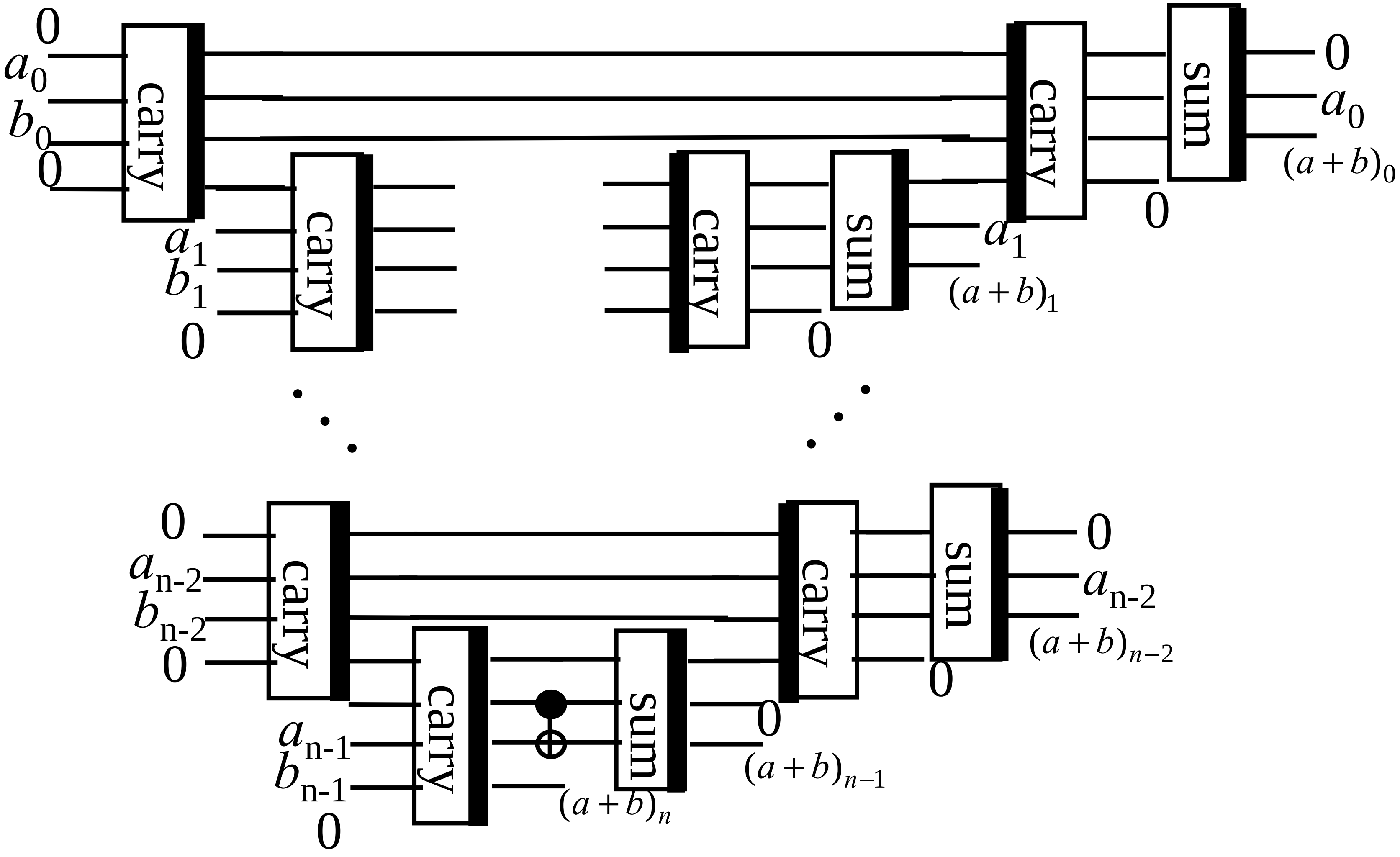
b1 b0

$(a+b)_2$ $(a+b)_1$ $(a+b)_0$



Exercise 3: answer

n-bit adder



Problem 35.

- Decompose the matrix below as a product of two-bit unitary matrices.

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

10 points

Solution to Problem 35

Use the technique mentioned in Nielsen and Chuang, p189-191 This is hard and I do not expect you to be able to do it. $U=U_1U_2U_3U_4U_5U_6$

$$U_1=\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad U_2=\begin{bmatrix} \sqrt{\frac{2}{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & -\sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad U_3=\begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{-\sqrt{3}}{2} \end{bmatrix}$$

$$U_4=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}}{4}(1+i) & \frac{1}{4}(3-i) & 0 \\ 0 & \frac{1}{4}(3+i) & \frac{\sqrt{3}}{4}(-1+i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad U_5=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{\frac{2}{3}} & 0 & -\frac{i}{\sqrt{3}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{i}{\sqrt{3}} & 0 & -\sqrt{\frac{2}{3}} \end{bmatrix} \quad U_6=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 & \frac{-i}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \end{bmatrix}$$

Problem 36.

There are several edge-detection algorithms for black and white and color images used in practice. We presented some of them.

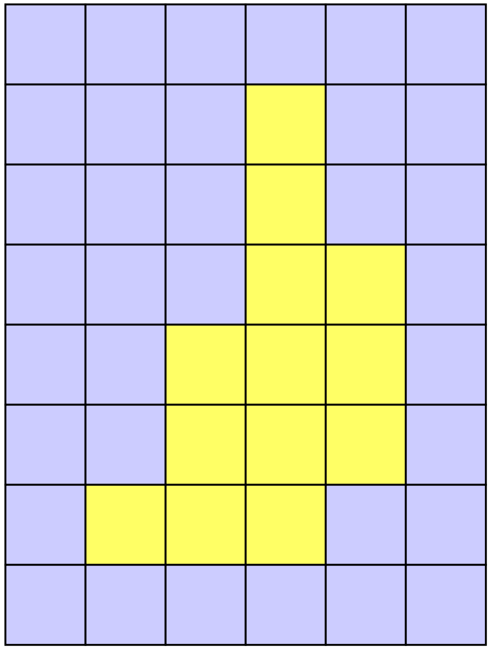
1. Design a cellular automata based processor for edge detection.
2. Use any algorithm for edge detection, black-and-white or grey-level images.

3. Design logically the processor of the single cell and show how it is connected to its neighbors.

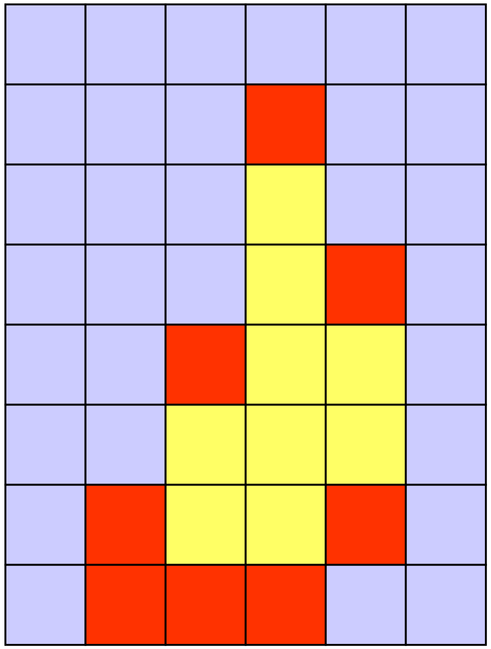
<u>20 points</u>

Solution to Problem 36.

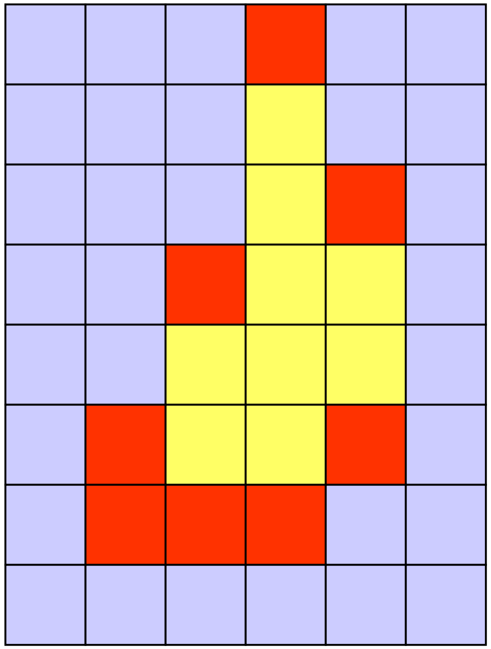
In a cellular automaton of a cell you can use Sober, Prewitt or any other edge-detection rule. Below we will use the simplest rule which is a difference of values horizontally or vertically. For binary images use EXOR, for grey-level use the value of difference exceeding certain threshold.



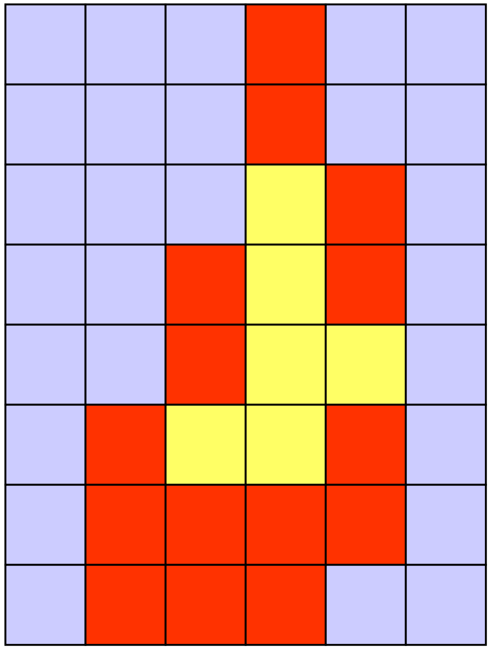
Original image



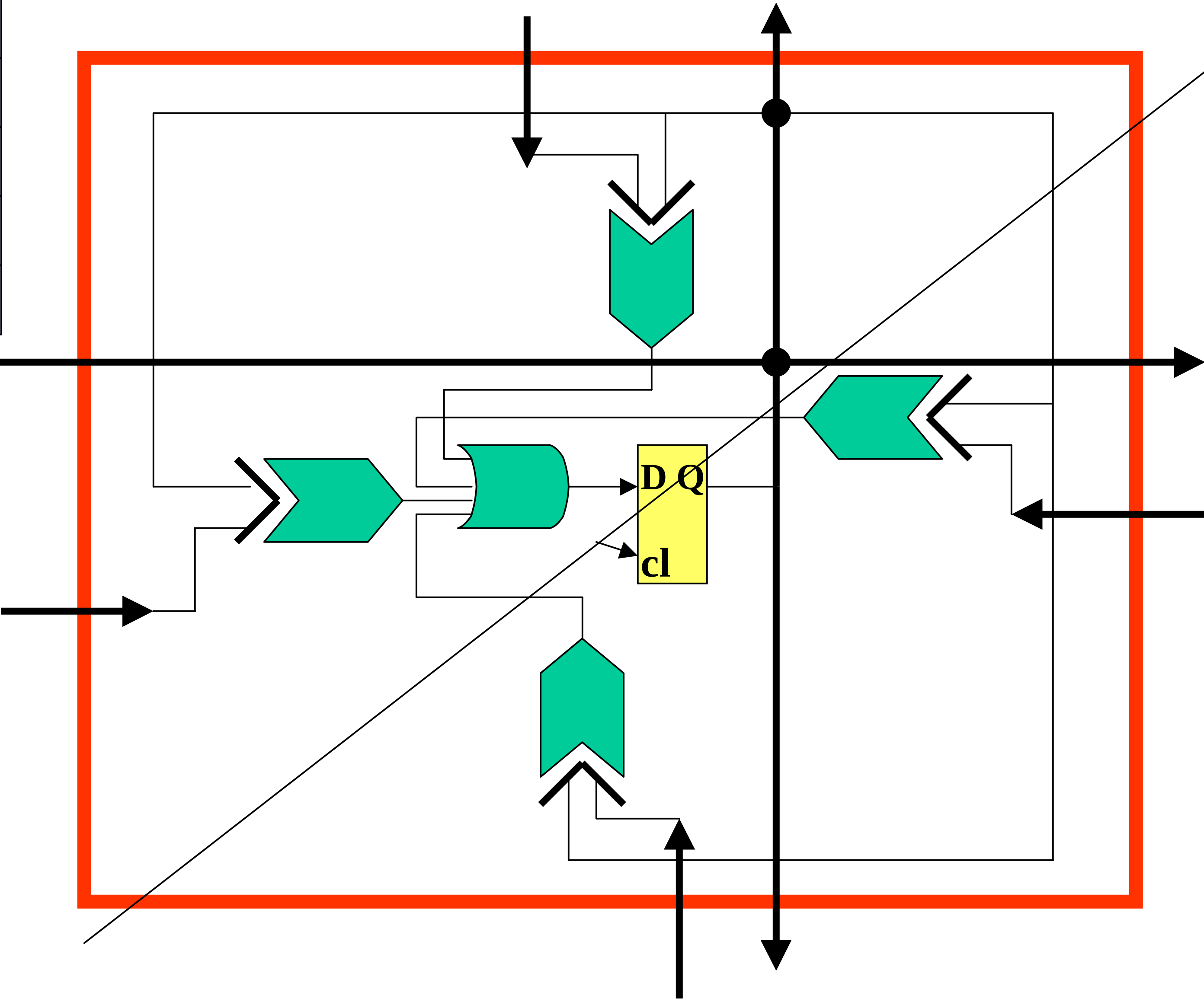
Exors of cell with upper cell



Exors of cell with lower cell



The design of a single cell is. All flip-flops are connected to the same clock.



Exors of cells above or below with our cell. Similar patterns can be created for left and right cells.

Problem 37.

1. Design any processor for Satisfiability Checking. You can use the *pipelined architecture* from Monday meeting, but you have to design Boolean instructions and random number generator in more detail.
2. How to modify slightly your ideas from point 1 for Petrick Function Minimization?.
3. How to further modify these ideas for simultaneous solving of Satisfiability and Minimizing the number of positive literals in the product – solution, if the product of literals is a solution.

25 points

Solution to Problem 37.

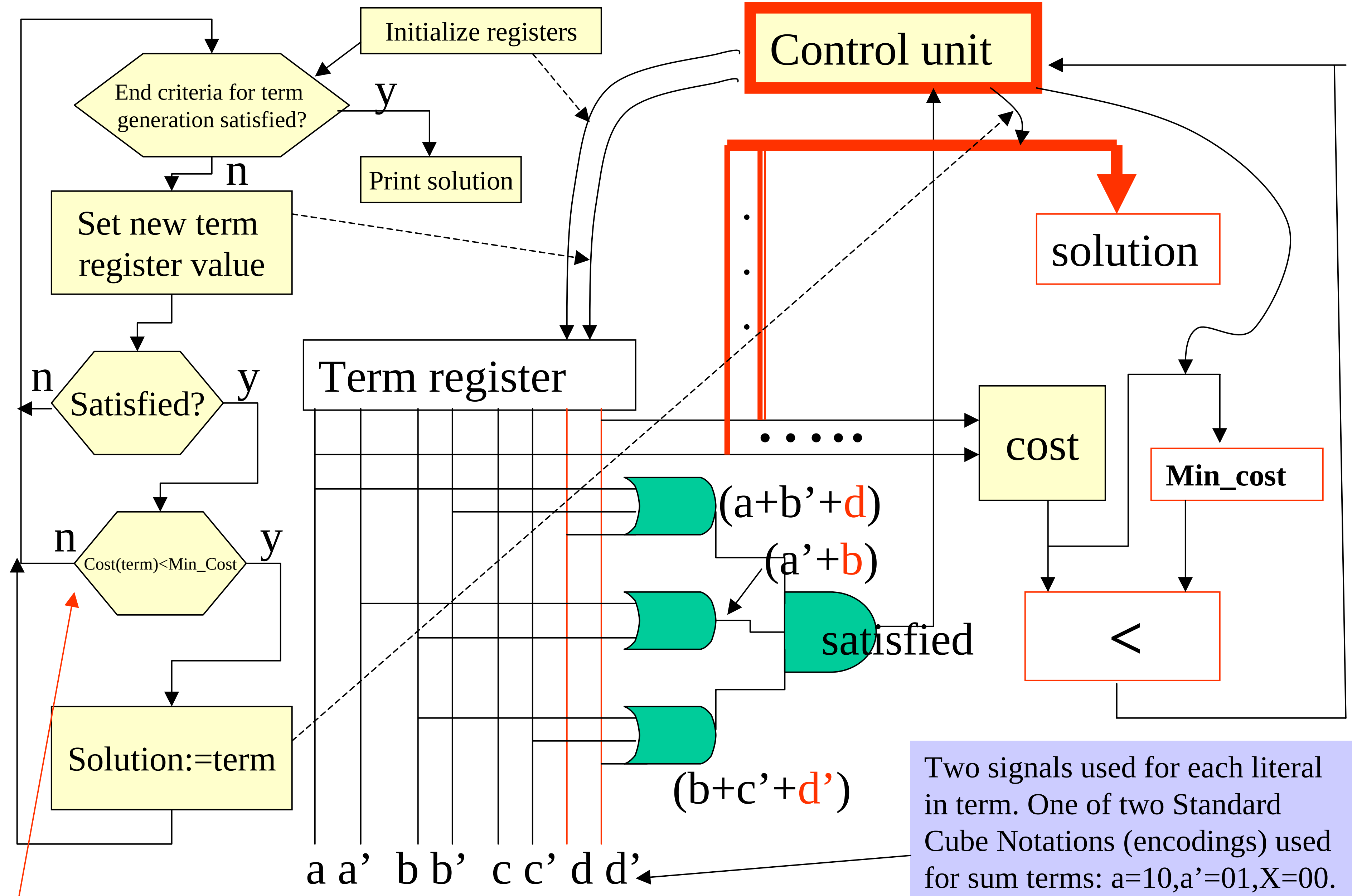
The pipelined processor for Satisfiability Checking was shown in Problem 4.

Here we will show another architecture for simultaneous solving of Satisfiability and Minimizing the number of positive literals in the product – solution, if the product of literals is a solution.

Several complete designs in VHDL of various variants of such machines can be found on my American Webpage.

Petrick function is in a form of Product of Sums of positive variables. Satisfiability is in a form of Product of Sums of literals. Generalized architecture will assume Product of Sums of literals, check for satisfiability and calculate the minimum number of literals (both positive and negative) that satisfy the formula.

The basic idea is to check the formula satisfiability directly in hardware, since this is combinational function. The verification is repeated for new literal candidates and the costs are calculated and compared. There are exhaustive and probabilistic variants of this algorithm.



Two signals used for each literal in term. One of two Standard Cube Notations (encodings) used for sum terms: a=10,a'=01,X=00.

Is the term (new candidate solution)cheaper than stored candidate ***solution***?

You can now design the control unit FSM using standard methods.

There are several variants of the control

- In the probabilistic variant of the algorithm the Term register is a random number generator.
- In the exhaustive (exact solution) variant of the algorithm the Term register is a counter. It can be a counter in binary code modified in such a way that 00=11 on the output. It can be a ternary counter counting up in code 00,01,10,00… for each literal. The best solution is a counter that counts in a code that corresponds to the breadth searching of the space of terms. This counter counts in an increasing number of literal terms. In such case the first solution is optimal. Design of such counter is difficult, but a ROM can be used to store once all the patterns. The contents of such ROM can be calculated by a program.

a	b	c	d	
00	00	00	00	= XXXX
00	00	00	01	= XXXd’
00	00	00	10	= XXXd
00	00	01	00	= XXc’X

Here is an example of literal term generation

The order of “Breadth Searching” is: d’,d,c’,c,b’,b,a’,a,d’c’,d’c,dc’,dc,d’b’…

Problem 38.

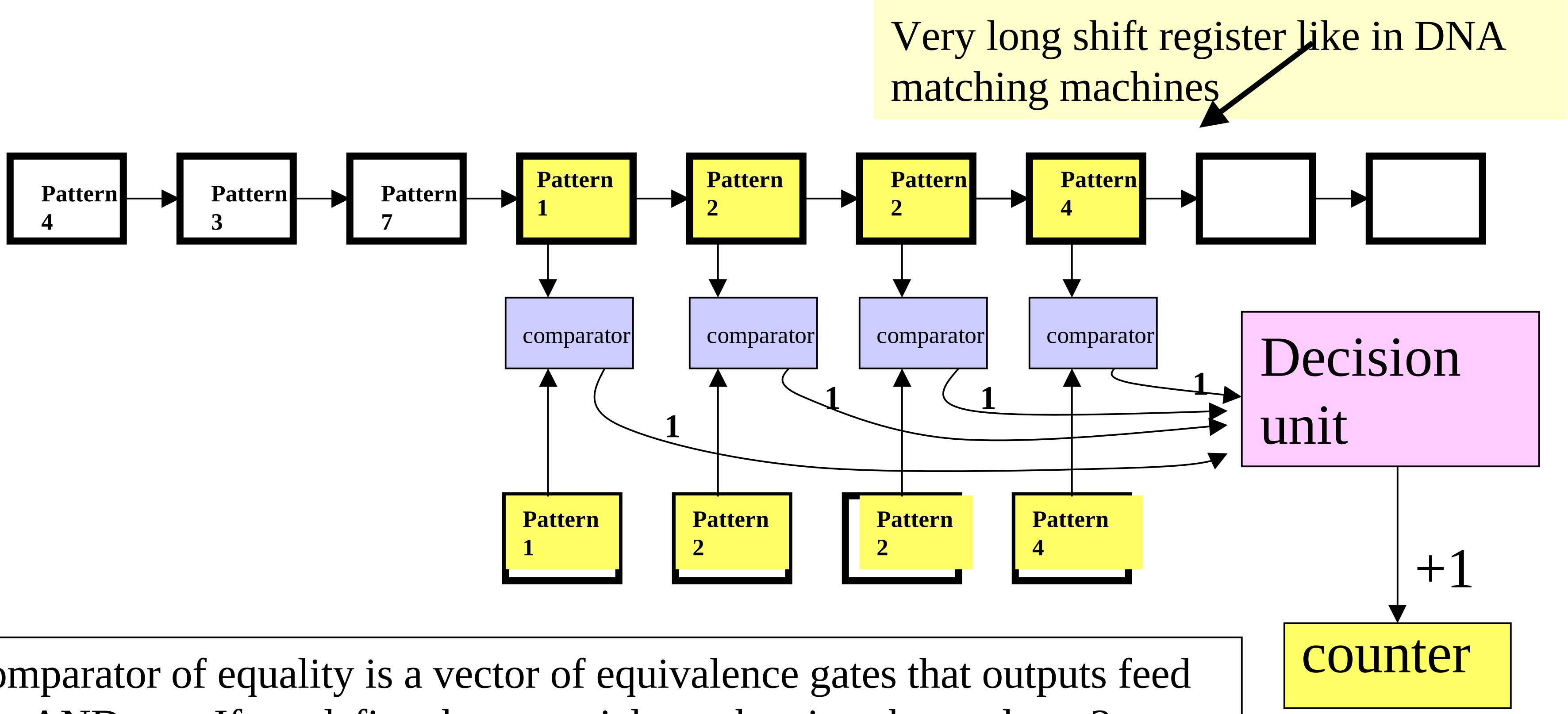
Given is a very long string of characters that is “shifted” from an external memory to a very long shift register.
1. Design a pipelined processor that“observes” the shift register and checks for matching arbitrary number of patterns (short strings of characters) at the same time.
2. Extend it to partial matching and counting of matches.

- This problem was outlined in one of additional meetings.

20 points

Solution to Problem 38.

Assume we match four patterns in parallel. Each of four Comparators can be for equality or for partial match. Decision unit is an AND gate in the simplest case when comparators are equality comparators. It can be a majority or threshold gate, or any (perhaps symmetric) Boolean function, depending on match definition. Shift causes the entire pattern register to be shifted one position to the right. The situation below corresponds to a perfect match of a sequence of four “characters”.



Very long shift register like in DNA matching machines

Comparator of equality is a vector of equivalence gates that outputs feed one AND gate.If we define that a partial match exist when at least 3 patterns are satisfied, then the Decision Unit realizes the majority 3 out of 4 function, which is a symmetric function S^{34}(a,b,c,d) realized using standard methods of this class.

Problem 39.

Given are two integer arrays A and B. Design a processor that calculates a Kronecker Product A * B in hardware – any kind of parallelism is OK, but the processor should have some kind of parallelism.

Assume 2*2 arrays, but architecture should be for arbitrary square arrays. Assume multipliers and adders as black boxes, do not design their internals. Show the block diagram and discuss control.

25 points

$$\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} \otimes \begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix} = \left(\begin{array}{c|c|c|c} a11*b11 & a11*b12 & a12*b11 & a12*b12 \\ \hline a11*b21 & a11*b22 & a12*b21 & a12*b22 \\ \hline a21*b11 & a21*b12 & a22*b11 & a22*b12 \\ \hline a21*b21 & a21*b22 & a22*b21 & a22*b22 \end{array}\right)$$

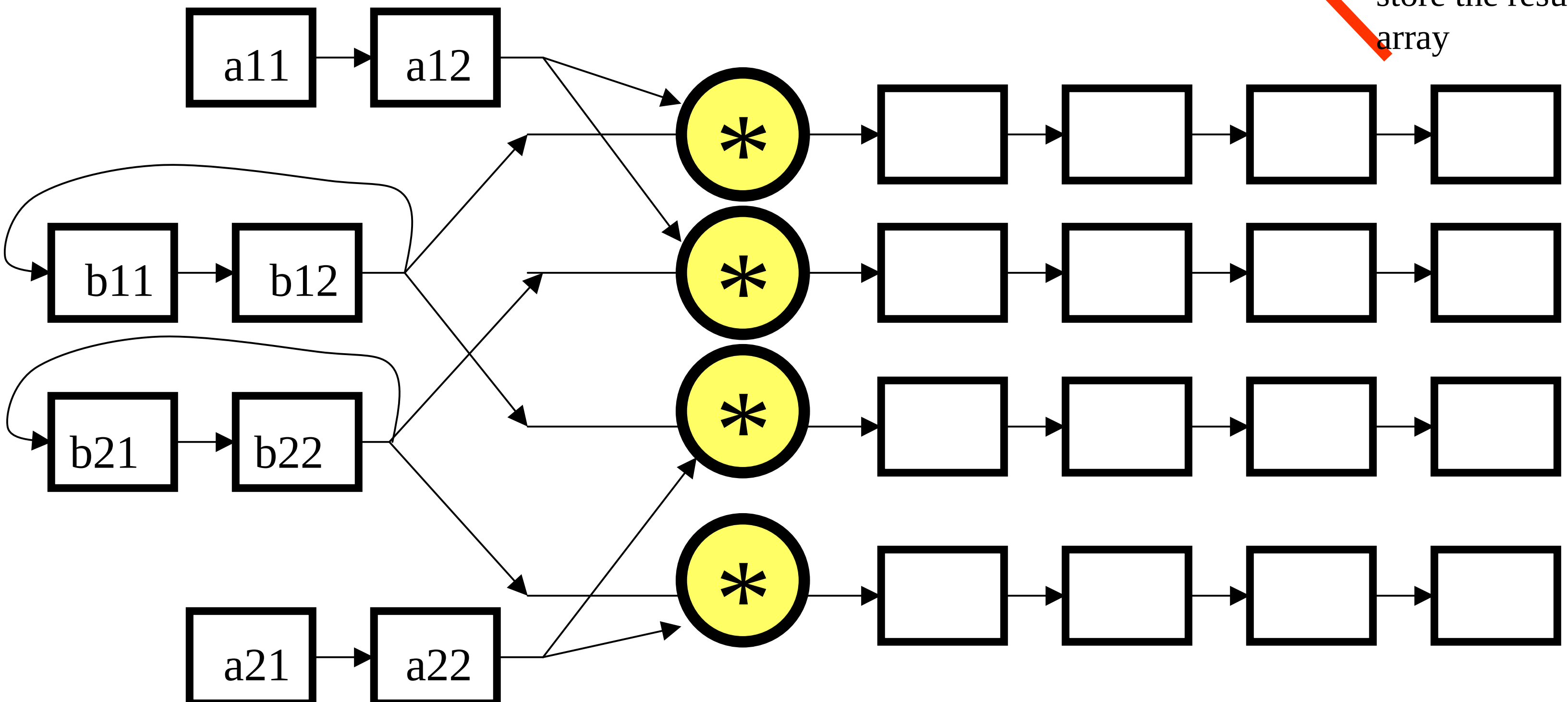
In parallel

From definition of Kronecker product

Solution to Problem 39.

According to the method from the class we have first to decide if we use memories of registers. Assume we use registers. Now we have to decide how many basic operations (in this case only multiplications) done in parallel. Assume that we realize one column in parallel.This leads directly to the data path below.

Shift registers to store the resultant array



The timing of this architecture is the following:

1. Multiply the last (fourth) column. (a12 on top, a22 on bottom, b12 and b22 in bij registers).

2. Shift the results of all multiplication to the right, to the resultant array. Shift cyclically registers bij

3. Multiply the third column(a12 on top, a22 on bottom, b11 and b21 in bij registers).

4. Shift the results to right. Shift cyclically registers bij. Shift registers aij

5. Multiply the second column. (a11 on top, a21 on bottom, b12 and b22 in bij registers).

6. Shift the results to right. Shift cyclically registers bij.

7. Multiply the first column. (a11 on top, a21 on bottom, b11 and b21 in bij registers).

It is now easy to design the controller from this algorithm above, using standard methods. Generalization of the method to any sizes of matrices is also straightforward because of the regularity of the architecture and its similarity to other array-based architectures from the class.

Problem 40.

What is the role of Kronecker (tensor) product in computer architecture, in quantum computing and in logic synthesis? Give examples, try to generalize your answer. Are there similarities? How they can be used?

15 points

Solution to Problem 40.

- Kronecker Product is also called tensor product. It takes two matrices or two vectors and creates a new matrix or a new vector. See next slides for explanation of the operations.

Kronecker product is used in quantum computing predominantly to calculate the resultant matrix of unitary matrices of two quantum gates or blocks that are in parallel. In logic synthesis it is used to calculate the matrix of orthogonal (linearly independent) matrix from matrices of expansion transformations, for instance we illustrated it for Walsh and Reed-Muller transformations. In case of Reed-Muller transformations we found the Kronecker product of the kernel of Davio expansion with itself.

A computer architecture (butterfly or pipeline) are based on the flowgraph of applying the Kronecker transform. This way computer architectures are created for any transform.

Similarities based on Kronecker Product can be used to create new transforms, new canonical logic forms, new architectures. For instance we used them in class when we discussed Walsh, Reed-Muller and Fourier transforms. Whenever you have to deal with a transform, think about its kernel, Kronecker Product and how to create few stages of data-flow for this transform.

The answer is also illustrated by solutions to other problems in this exam

Application of Kronecker product vectors in quantum computing

If we concatenate two qubits

$$\left(\alpha_0|0\rangle + \alpha_1|1\rangle\right)\ \left(\beta_0|0\rangle + \beta_1|1\rangle\right)$$

we have a 2-qubit system with 4 basis states

$$|0\rangle|0\rangle = |00\rangle \qquad |0\rangle|1\rangle = |01\rangle \qquad |1\rangle|0\rangle = |10\rangle \qquad |1\rangle|1\rangle = |11\rangle$$

and we can also describe the state as

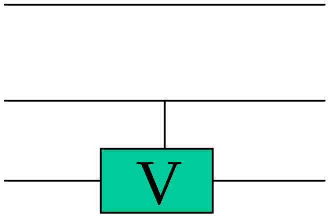
$$\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$

Kronecker Product of state vectors

or by the vector

$$\begin{pmatrix}\alpha_0\beta_0\\ \alpha_0\beta\\ \alpha_1\beta_0\\ \alpha_1\beta_1\end{pmatrix} = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix} \otimes \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}$$

Application of Kronecker Product of matrices to find the Unitary matrix of a parallel connection of blocks



A parallel connection of a wire and a controlled-V gate

$$U_1 = I_1 \otimes C(V)$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & v_{00} & v_{01} \\ 0 & 0 & v_{10} & v_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{00} & v_{01} & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{10} & v_{11} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & v_{00} & v_{01} \\ 0 & 0 & 0 & 0 & 0 & 0 & v_{10} & v_{11} \end{pmatrix}$$

Unitary matrix of a wire

Kronecker since this is a parallel connection

Unitary matrix of a controlled V gate (from definition)