ELE6XXE
Lectures

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

Systolic Array Architecture

- A systolic array is a homogeneous network of tightly coupled data processing units (DPUs) called cells or nodes.
- Each node or DPU independently computes a partial result as a function of the data received from its upstream neighbors, stores the result within itself and passes it downstream.

▸ Systolic arrays were invented by H. T. Kung and Charles Leiserson

▸ Array Processors and Mapping

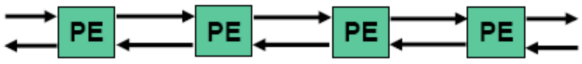
Systolic Array Architecture

- Uniform: Each PE computes the same set of combinatorial functions.
- Regular: All PEs are connected to a small finite number of neighboring PEs via one or more D-elements according to a regular topology. All connections are point-to-point connections.
- Synchronous operation: All PEs operate in lock step (fire concurrently) ; data is pumped through the system, much like the hart pumps blood through the body (hence the name systolic).

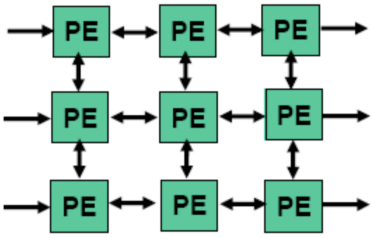
To obtain better systems small relaxations to the systolic model are allowed:

- Not all PEs are identical,small deviations are allowed especially for PEs at the border of the system.
- (A limited form) of broadcasting is allowed.
- Connections need not be to nearest neighbors, but locality needs to be maintained.

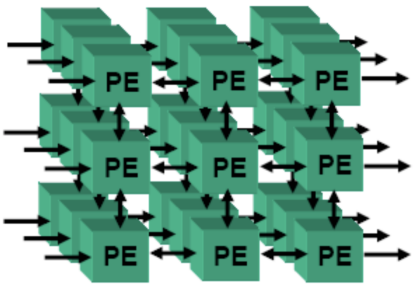
Systolic Array Architecture



1D array



2D array



3D array

Why Systolic Array

- A new class of pipelined array architectures
- Benefits – Simple and regular design (cost-effective) – Concurrency and communication, – Modular and expandable,
- Drawbacks – Not all algorithms can be implemented using a systolic architecture, – Cost in hardware and area, – Cost in latency.

Systolic Array Architecture

- Systolic architecture are designed by using linear mapping techniques on regular Dependence Graph (DG)
- Regular Dependence Graph: the presence of an edge in a certain direction at any node in the DG represents presence of an edge in the same direction at all nodes in the DG
- DG corresponds to space representation $\rightarrow$ no time instance is assigned to any computation
- Systolic architectures have a space-time representation where each node is mapped to a certain processing element (PE) and is scheduled at a particular time instance.
- Systolic design methodology maps an N-dimensional DG to a lower dimensional systolic architecture

Regular Iterative Algorithm

Regular Iterative Algorithms contain all algorithms executed by systolic arrays. ▸ Regular Iterative Algorithm, Homework 1 and 2 in page 261

An RIA is defined by the triple I, X, F where 1) An index space $I = (i, n)^T$, 2) A finite set of variables $X = (x, y, w)$. 3) functional relationship among the indexed variable.

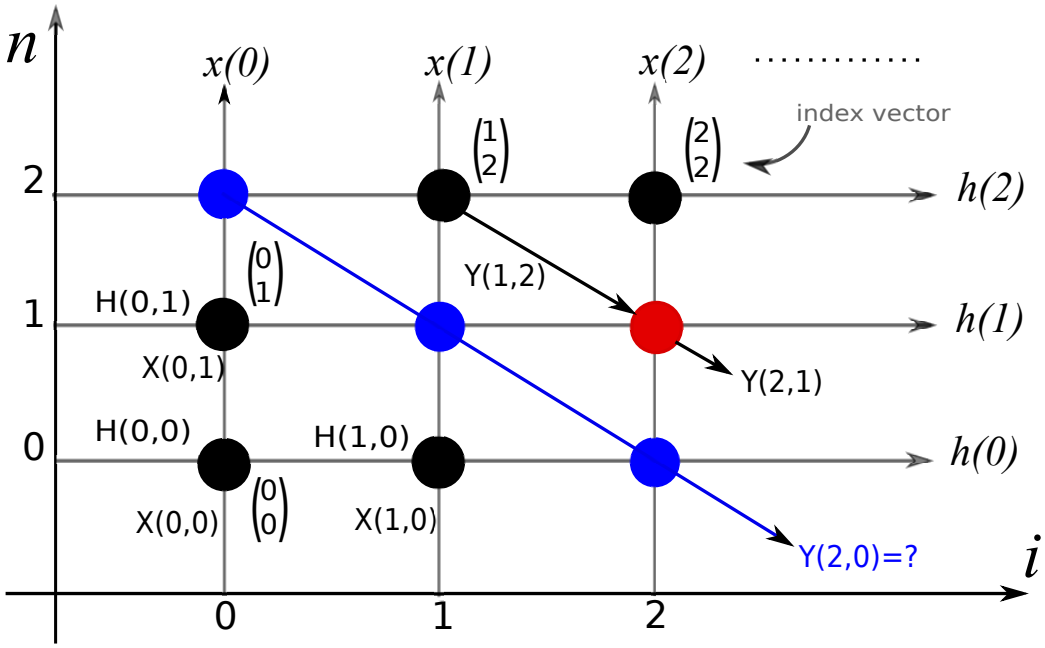
Canonical forms: 1. Standard input(not), 2. Standard output:

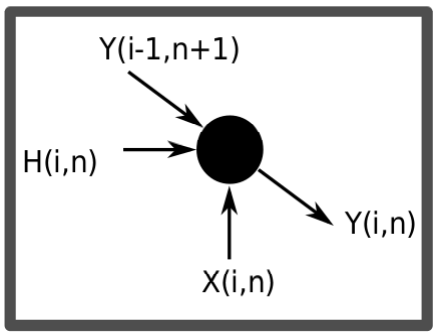
Y(i,n)=H(i,n)X(i,n)+Y(i-1,n+1)
H(i,n)=h(n)
X(i,n)=x(i)
Y(i,n)=h(n)x(i)+Y(i-1,n+1)
Y(2,0)=h(0)x(2)+Y(1,1)
Y(1,1)=

Index displacement vectors:

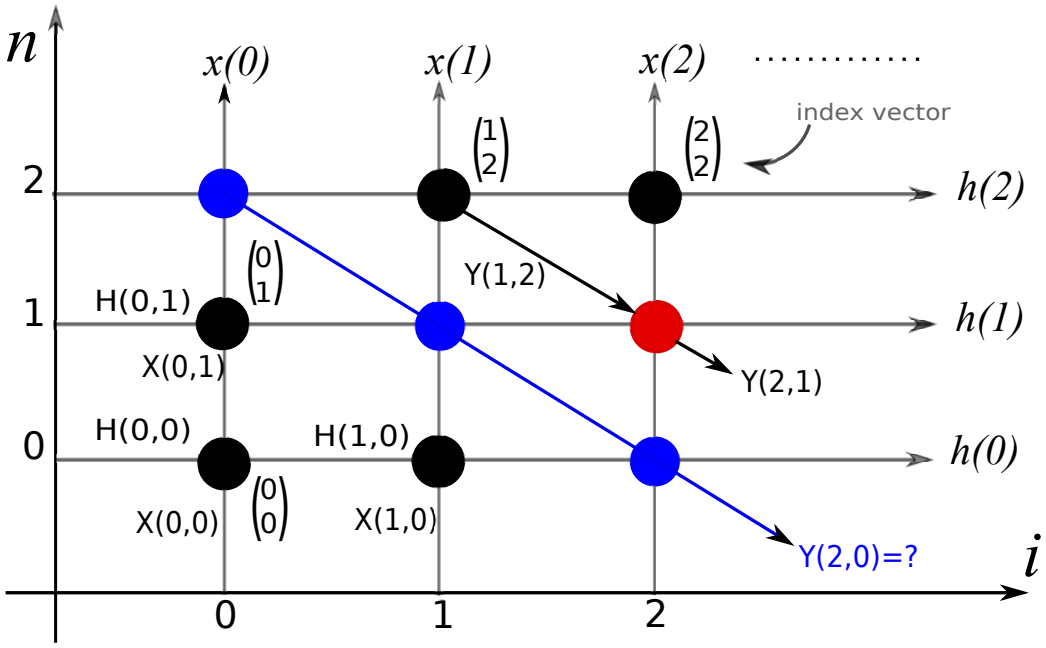
$$X \to X: \ (0\ 1)^T$$
$$H \to H: \ (1\ 0)^T$$
$$X \to Y: \ (0\ 0)^T$$
$$Y \to Y: \ (1\ -1)^T$$

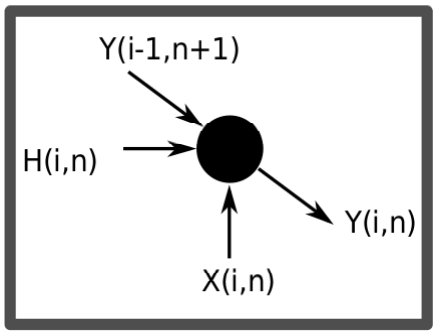
Dependence Graph





Dependence Graph

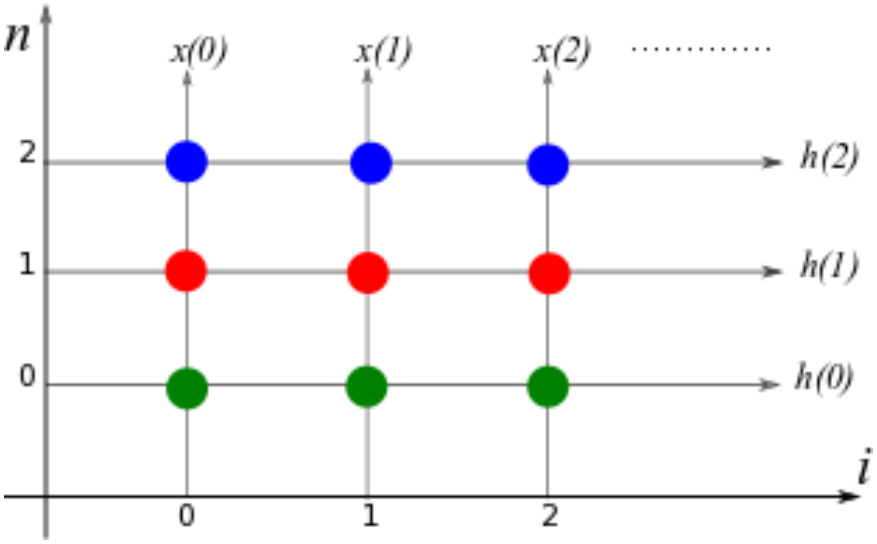




Fundamental edges $\mathbf{e_x} = (0\ 1)^T$, $\mathbf{e_h} = (1\ 0)^T$, $\mathbf{e_y} = (1\ -1)^T$

Dependence Graph $\rightarrow$ Systolic Architecture

$\mathbf{p}$:Processor space vector
Node $\rightarrow \mathbf{p}^T(i,n)^T \rightarrow$ Processing Element (linear mapping)
Örnek :$\mathbf{p}^T = (0\ 1)$

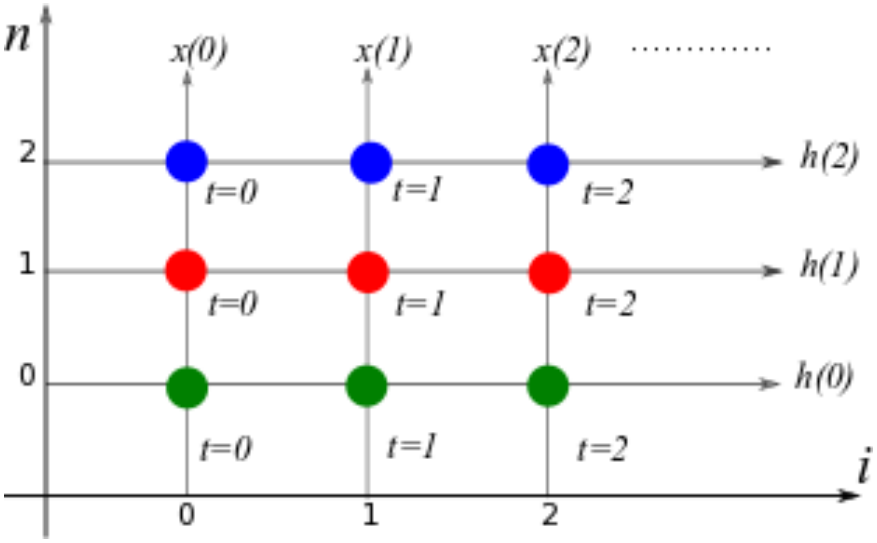


Dependence Graph $\rightarrow$ Systolic Architecture

Each node is mapped to a PE (with $\mathbf{p}^T(i,n)^T$) and is scheduled at a particular time instance with Schedule vector $\mathbf{s}$ such that

$$\mathbf{s}^T(i,n)^T$$

Örnek :$\mathbf{s}^T = (1\ 0)$



Definitions

- Processor space vector $\mathbf{p}$
  – Any node with index $I = (i, n)$ would be executed by PE $\mathbf{p}^T I$
- Projection vector (also called iteration vector) $\mathbf{d}$
  – Two nodes ($I(x)$ and $I(y)$) that are displaced by $\mathbf{d} = I(x) - I(y)$ or multiples of $\mathbf{d} = \alpha(I(x) - I(y))$ are executed by the same PE.

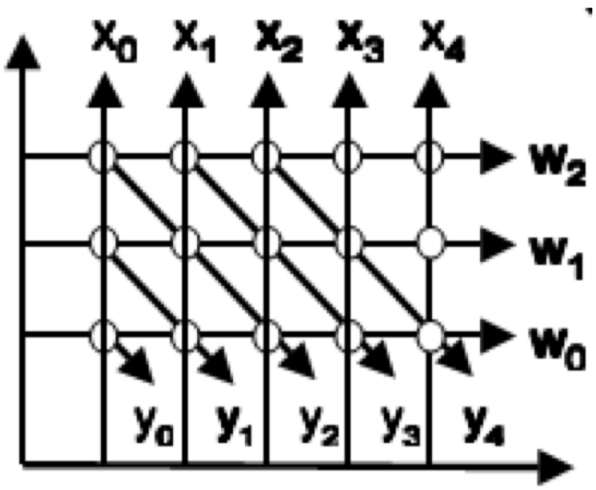
$$\mathbf{p}^T I(x) = \mathbf{p}^T I(y) \rightarrow \mathbf{p}^T I(x) - \mathbf{p}^T I(y) = \mathbf{p}^T (I(x) - I(y)) = \mathbf{p}^T \mathbf{d} = 0$$

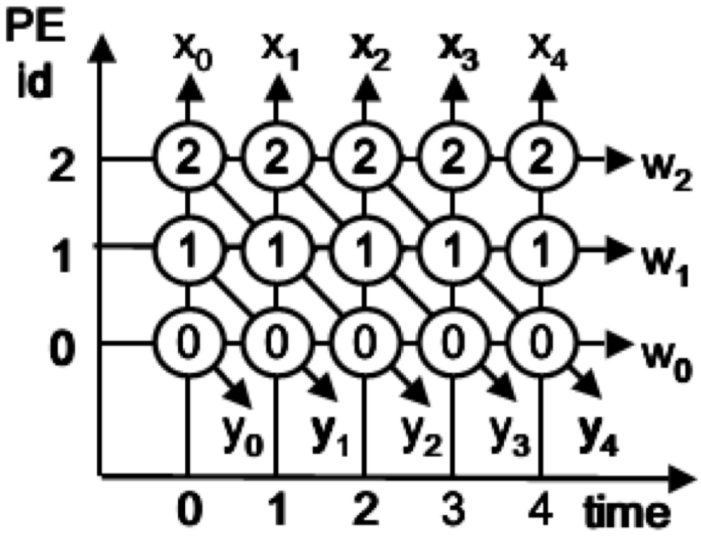
So $\mathbf{d}$ and $\mathbf{p}$ must be orthogonal !

- Scheduling vector $\mathbf{s}$
  – Any node with index I would be executed at time $\mathbf{s}^T I$
  Let $I(x)$ and $I(y)$ are mapped the same PE. So $\mathbf{s}^T I(x) \neq \mathbf{s}^T I(y)$ then $\mathbf{s}^T (I(x) - I(y)) = \mathbf{s}^T \mathbf{d} \neq 0$

Space to Space-Time Representation

- Interpreting one of the spatial dimensions as temporal dimension
- processor axis: $\mathbf{p}^T I$
- scheduling time instance: : $\mathbf{s}^T I$





Edge Mapping

If an edge $\mathbf{e}$ exists in DG, then an edge $\mathbf{p}^T\mathbf{e}$ exists in the systolic array with $\mathbf{s}^T\mathbf{e}$ delays.
Edge Mapping Table:

$\mathbf{e}$	$\mathbf{p}^T\mathbf{e}$	$\mathbf{s}^T\mathbf{e}$
Input: $\mathbf{e_x}$		
Weight:$\mathbf{e_h}$		
Output:$\mathbf{e_y}$		

Fundamental edges $\mathbf{e_x} = (0\ 1)^T$, $\mathbf{e_h} = (1\ 0)^T$, $\mathbf{e_y} = (1\ -1)^T$

Edge Mapping

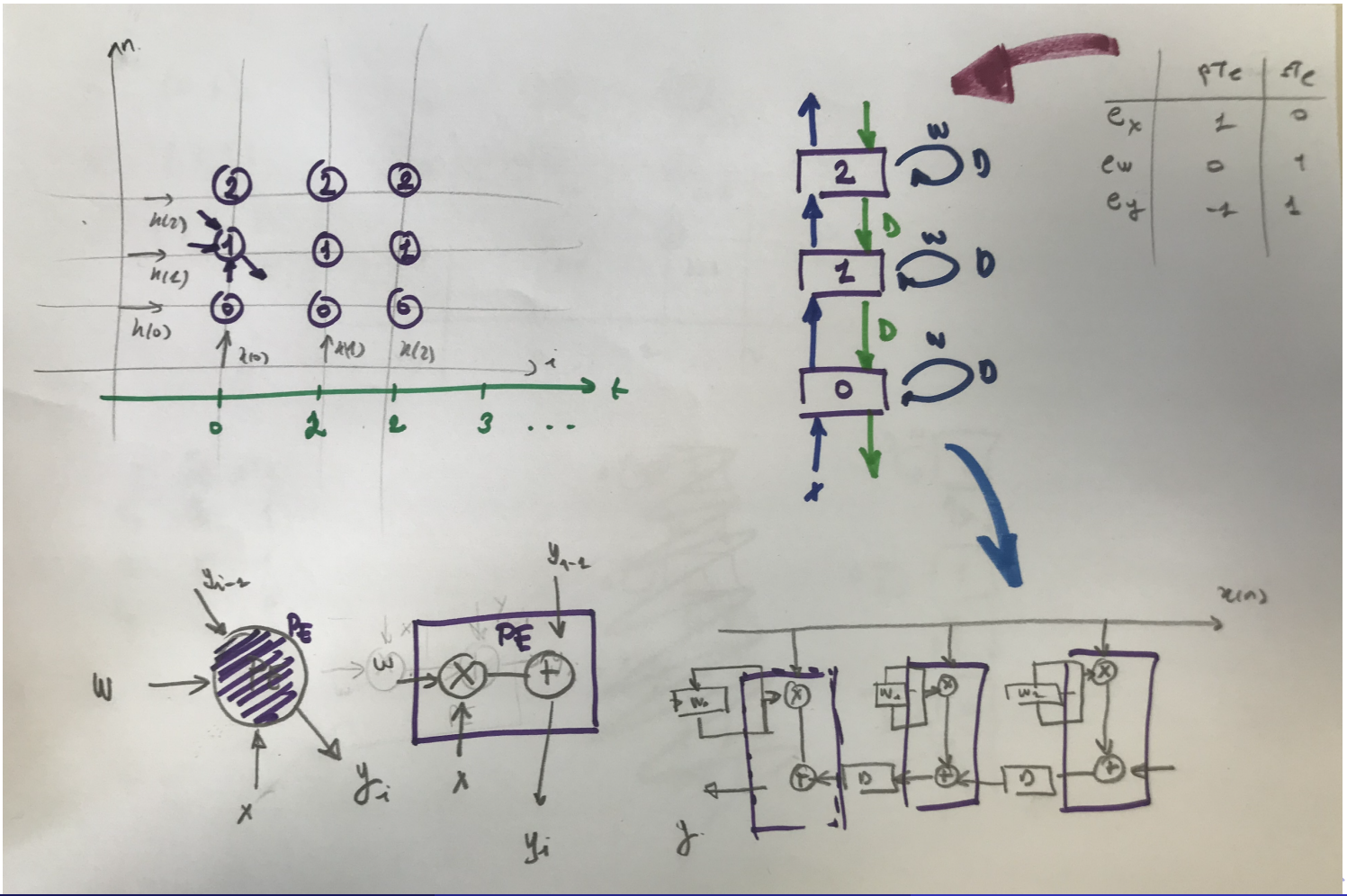
Fundamental edges $\mathbf{e_x} = (0\ 1)^T$, $\mathbf{e_h} = (1\ 0)^T$, $\mathbf{e_y} = (1\ -1)^T$

$\mathbf{B_1}$ design : $\mathbf{d} = \begin{pmatrix}1\\0\end{pmatrix}$, $\mathbf{p} = \begin{pmatrix}0\\1\end{pmatrix}$, $\mathbf{s} = \begin{pmatrix}1\\0\end{pmatrix}$

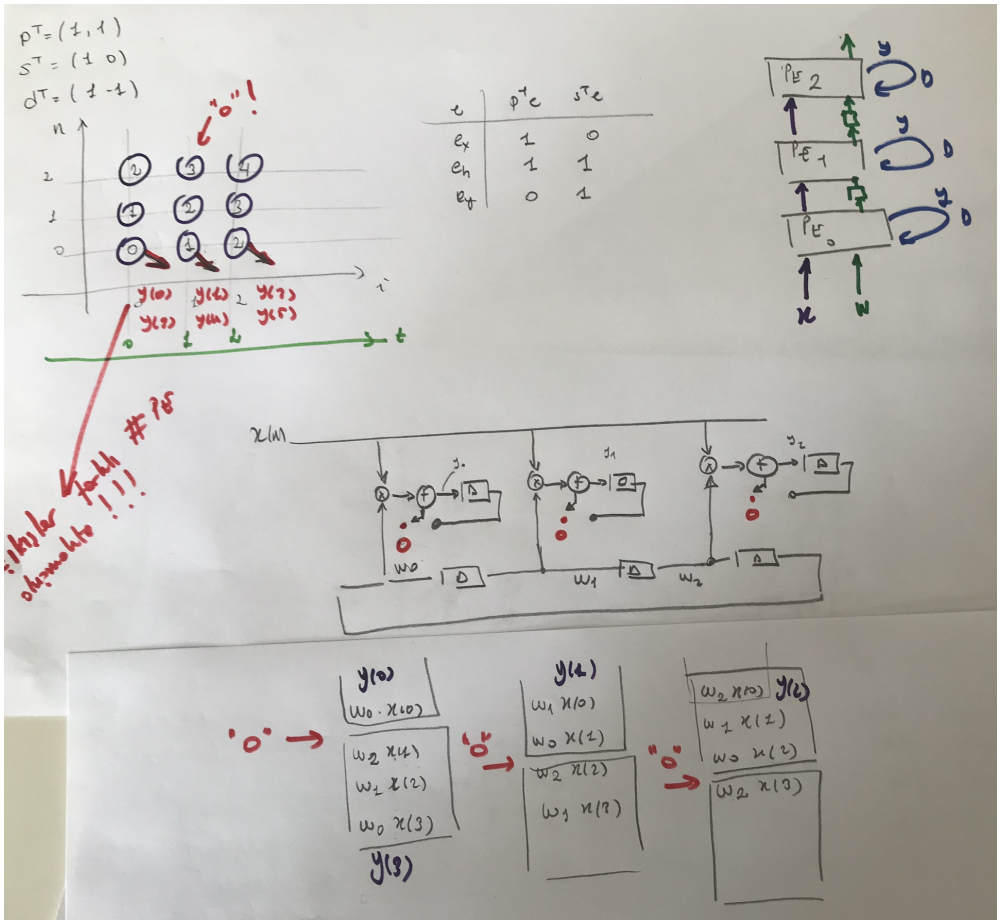
$\mathbf{e}$	$\mathbf{p}^T\mathbf{e}$	$\mathbf{s}^T\mathbf{e}$
$\mathbf{e_x}$	1	0
$\mathbf{e_h}$	0	1
$\mathbf{e_y}$	-1	1

Input is broadcast to PE. Weights appear at PE at the same coordinates. Outputs appear at the PEs at different space and time.

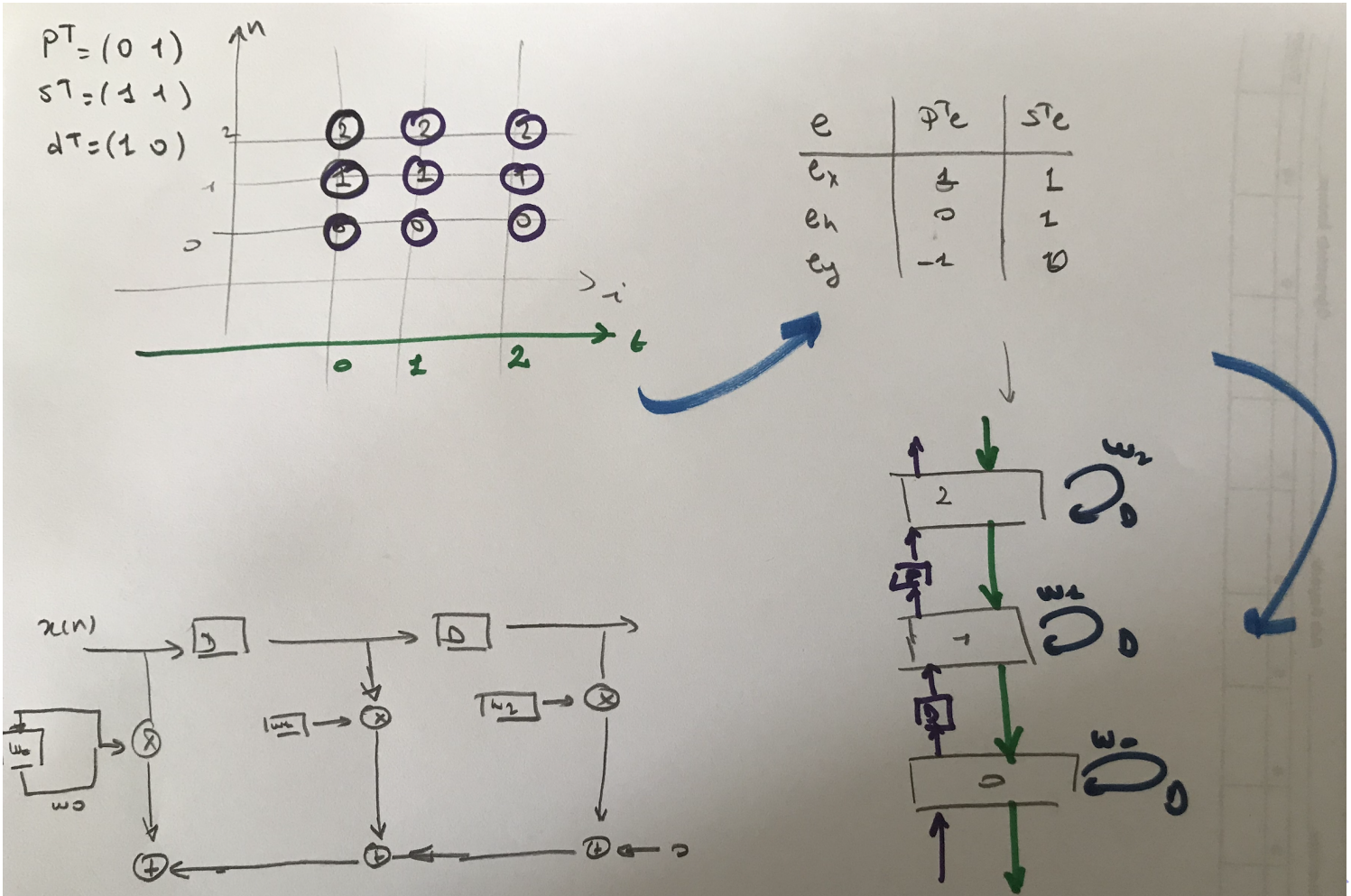
Low Level of Implementation of $\mathbf{B_1}$



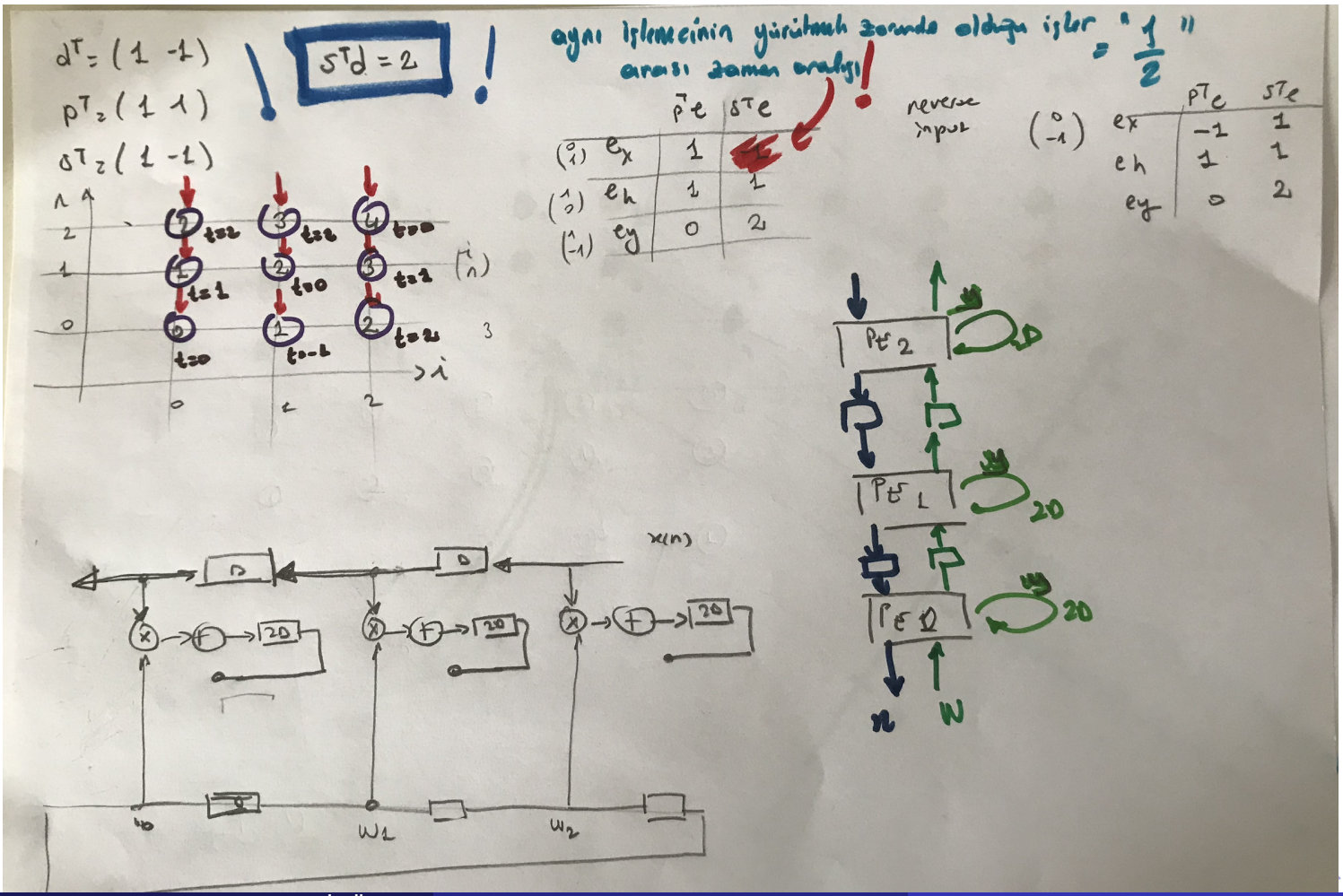
Low Level of Implementation of $\mathbf{B_2}$



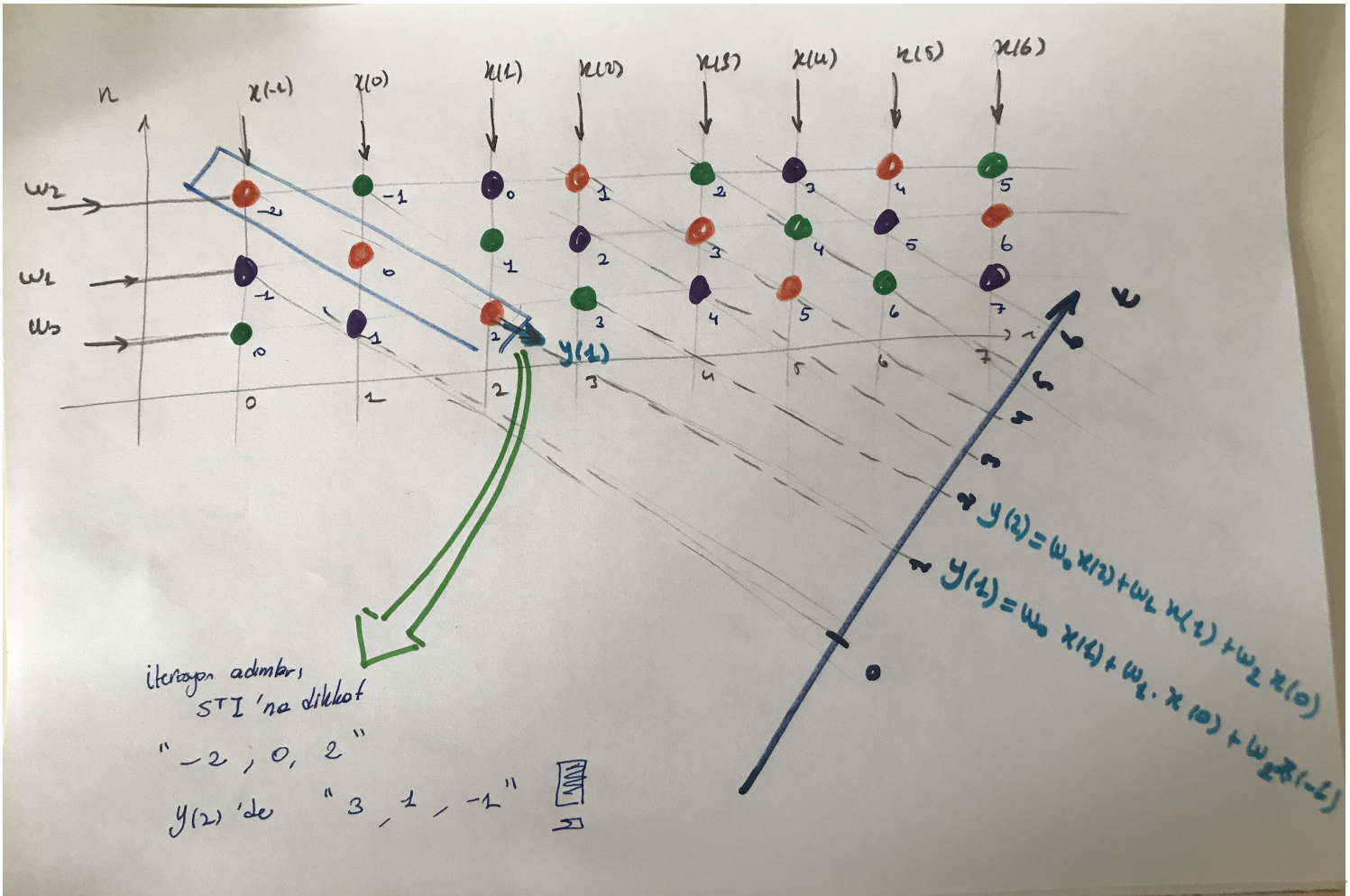
Low Level of Implementation of **F**



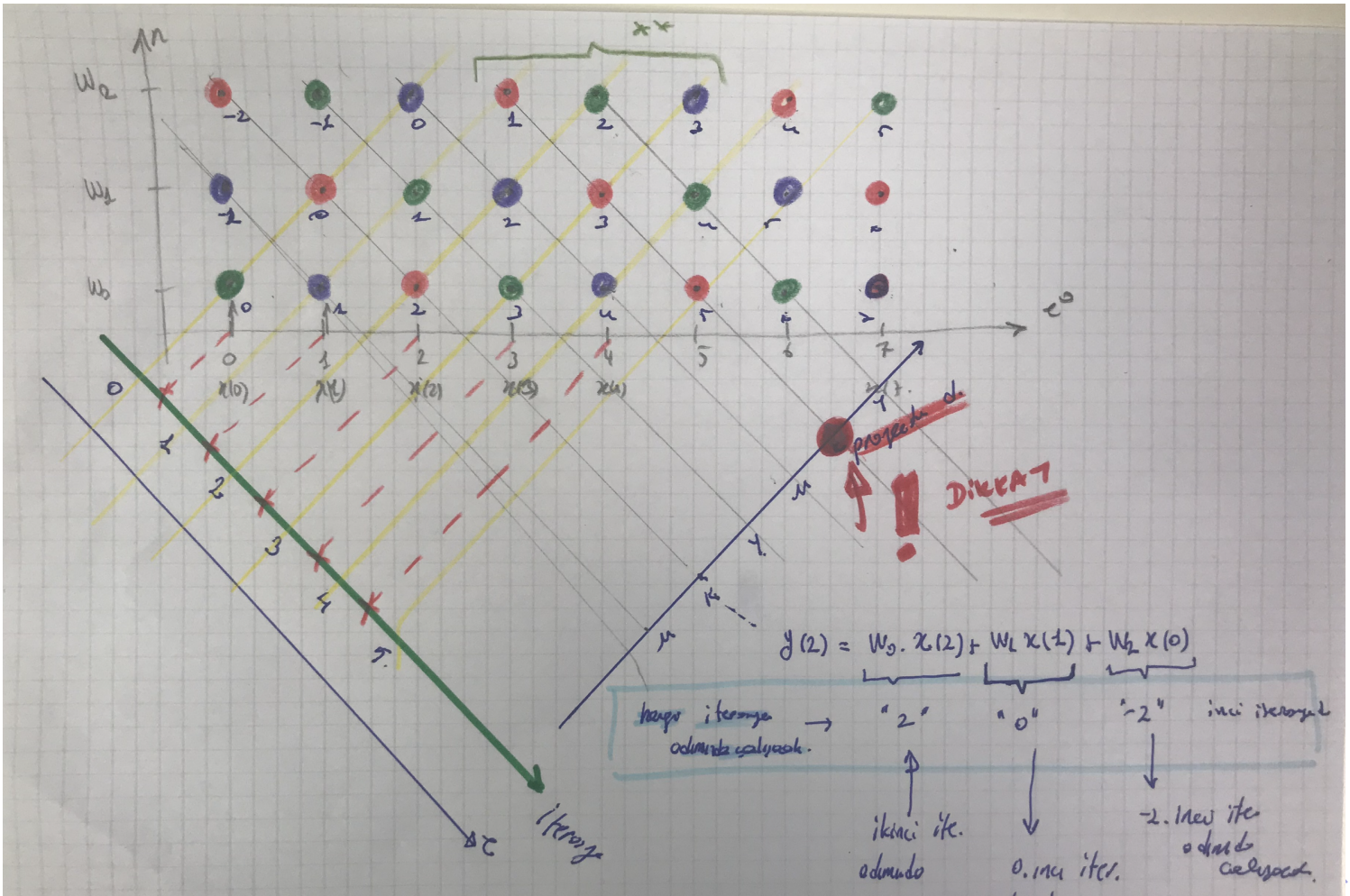
Low Level of Implementation of $\mathbf{R_1}$



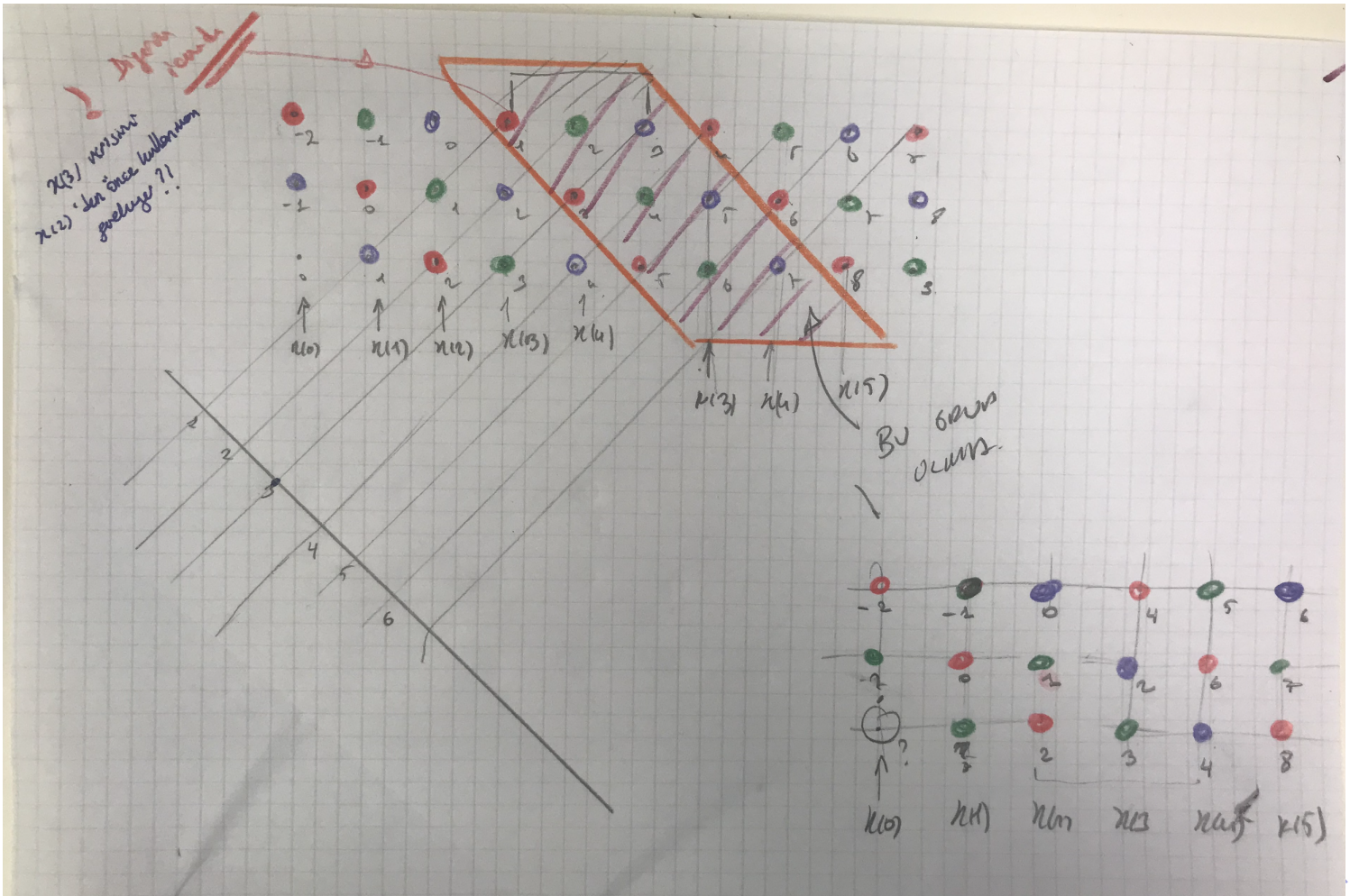
Low Level of Implementation of $\mathbf{R_1}$



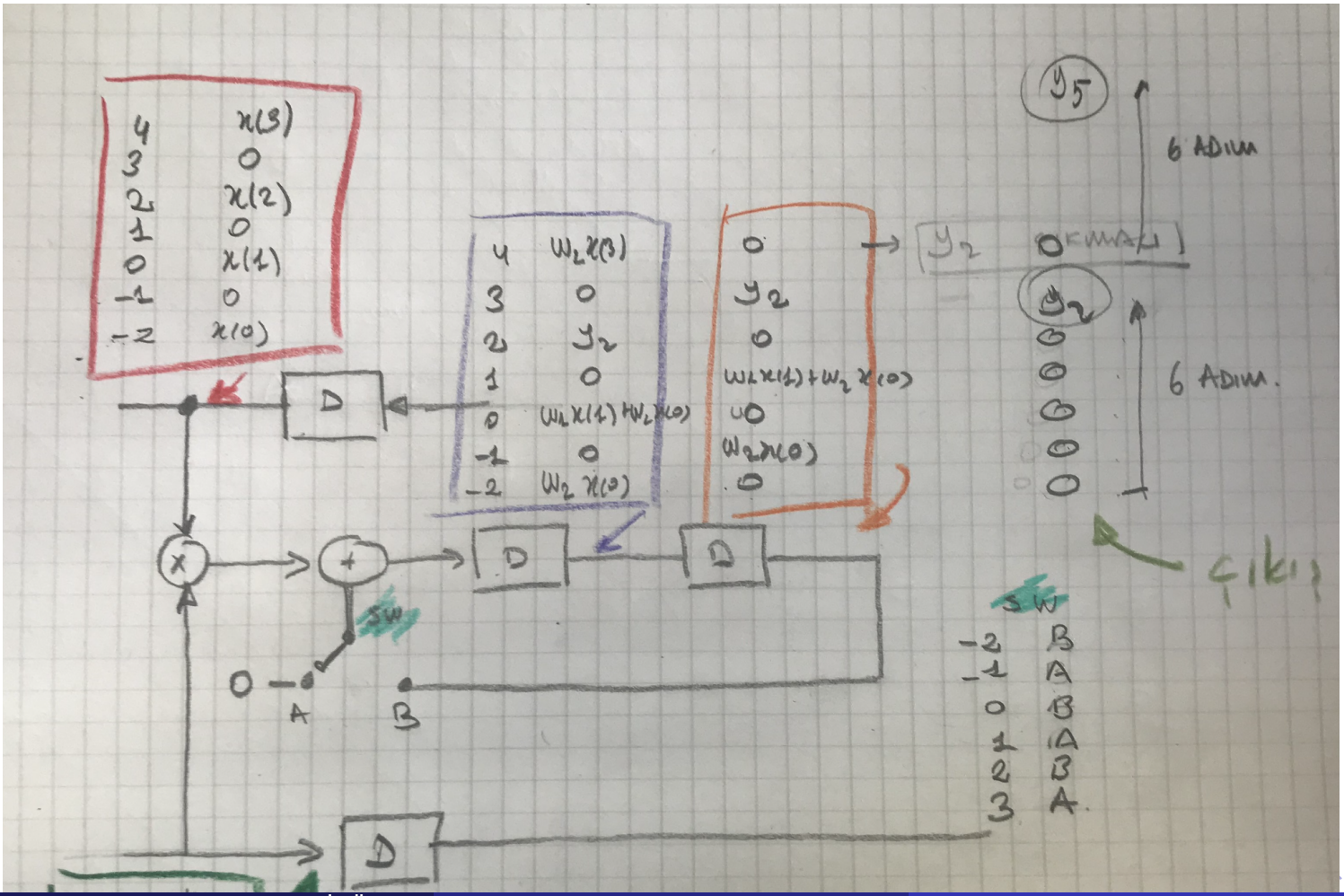
Low Level of Implementation of $\mathbf{R_1}$



Low Level of Implementation of $\mathbf{R_1}$



Low Level of Implementation of $\mathbf{R_1}$



Homework

	s^T	d^T	p^T	$p^T(e_h/e_x/e_y)$	$s^T(e_h/e_x/e_y)$
B1	(1, 0)	(1, 0)	(0, 1)	(0, 1,-1)	(1, 0, 1)
F	(1, 1)	(1, 0)	(0, 1)	(0, 1,-1)	(1, 1, 0)
W1	(2, 1)	(1, 0)	(0, 1)	(0, 1,-1)	(2, 1, 1)
W2	(1, 2)	(1, 0)	(0, 1)	(0, 1,-1)	(1, 2,-1)
DW2	(1,-1)	(1, 0)	(0, 1)	(0, 1,-1)	(1,-1, 2)
B2	(1, 0)	(1,-1)	(1, 1)	(1, 1, 0)	(1, 0, 1)
R1	(1,-1)	(1,-1)	(1, 1)	(1, 1, 0)	(1,-1, 2)
R2	(2, 1)	(1,-1)	(1, 1)	(1, 1, 0)	(2, 1, 1)
DR2	(1, 2)	(1,-1)	(1, 1)	(1, 1, 0)	(1, 2,-1)

reverse direction funda-mental edge

$e_y = -e_y$

$e_x = -e_x$

$e_x = -e_x$

$e_y = -e_y$

Determining **d**,**p**,**s**

- Trial-and-error approach
- Constructive approach : 1. Determine an **s** 2. Determine a **d** such that $\mathbf{s}^T\mathbf{d} = 0$ 3. Determine a **p** such that $\mathbf{p}^T\mathbf{d} = 0$.

Consider the dependence relation $X \rightarrow Y$ so

$$S_y \geq S_x + T_x$$

Using the all fundamental edges, construct the scheduling inequalities and solve them for feasible **s**.
Example 7.4.1