ELE6XXE
Lectures

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

Representations of DSP Algorithms

- Read : K. K. Parhi, VLSI Digital Signal Processing Systems Design and Imp., pp. 31-40
- Read: Giovanni De Micheli - Synthesis and Optimization of Digital Circuits, pp. 185-229.

DSP Alg. are described by nonterminating programs which execute the same code repetitevely.

```
while(1) {
y(n)=a x(n)+ b x(n-1)+ c x(n-2);
n++; }
```

- Iteration: execution of all comp.s in the alg. once.
- Iteration period (iteration rate): the time required for execution of one iteration of the alg.

Iteration: 3 MUL and 2 ADD generates 1 output.

Representations of DSP Algorithms

- For architectural design the math. formulations of DSP alg. need to be converted to behavioral description lang. or graphical representations.
- Graphical rep. are efficient for investigating and analyzing data flow properties of DSP alg. and for exploiting the parallelism.

DSP Alg. $\rightarrow$ graphical representation $\rightarrow$ structural implementation

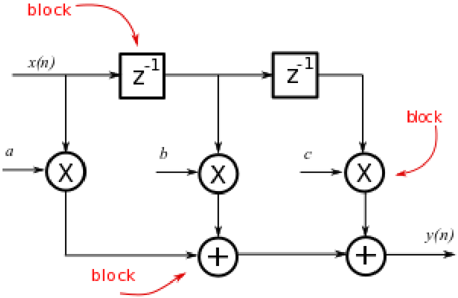
Graphical representations of iterations:

- Block diagram (BD)
- Signal-flow graph (SFG)
- Data-flow graph (DFG)
- Dependence graph (DG)

Block diagram (BD)

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

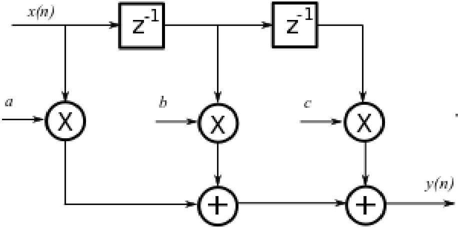
DSP Alg. $\rightarrow$ graphical representation { functional blocks with directed edges}

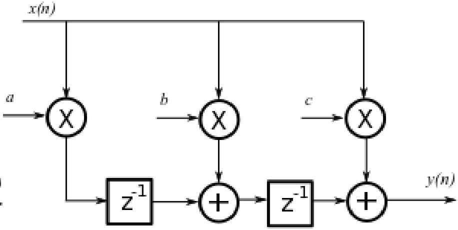


Block diagram (BD)

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

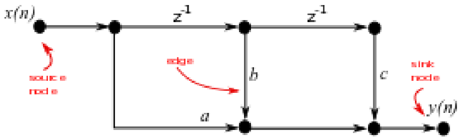
Various block diagram can be derived for the same system with different arrangements.





$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

DSP Alg. $\rightarrow$ graphical representation $\{$ connection of nodes and directed edges$\}$
nodes: represent computational tasks
edges (j,k) (directed): a linear transformation from the signal at node j to the signal at node k

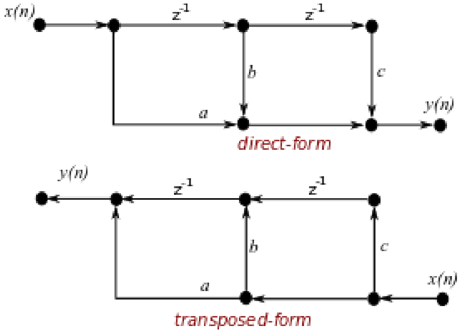


See: Signal Processing lecture...

Signal-flow graph (SFG)

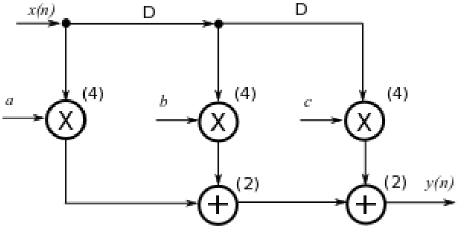
$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

Transposition of "linear" SFG



Data-flow graph (DFG)

node: represent computation (or function or sub-task)
edge:represent communication between nodes.
(xx): represent execution time of a node (unit of time (u.t.)).



DFG is closer to actual HW architecture and DFG describes the data flow among subtasks in signal proc. alg.

DFG are used for high-level synthesis to derive concurrent imp.s of DSP app. onto parallel HW.

Data-flow graph (DFG)

A directed DFG is denoted as $\mathbf{G} =< \mathbf{V}, \mathbf{E}, \mathbf{d}, \mathbf{w} >$ where the notations are as follows

- $\mathbf{V}$: Set of vertices (nodes) of $\mathbf{G}$. The vertices represent operations. The number of nodes in G is $|V|$.
- $\mathbf{E}$: Set of directed edges of $\mathbf{G}$. A directed edge from node $U \in \mathbf{U}$ to node $V \in \mathbf{V}$ is denoted as $U \rightarrow V$. The edges represent communication between the nodes. The number of edges in $\mathbf{G}$ is $|\mathbf{E}|$.
- $\mathbf{d}(\mathbf{U})$: pipeline depth of U (delay of vertice (node) U).
- $\mathbf{w}(e)$: Number of delays on the edge e, also referred to as the weight of the edge.

Data-flow graph (DFG)

Let A be the incidence matrix of the graph **G**,

$$a_{i,j} = \begin{cases} 1 & \text{edge i starts from node j,} \\ -1 & \text{edge i ends from node j,} \\ 0 & \text{otherwise.} \end{cases}$$

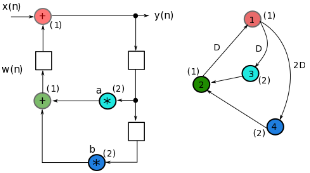
Loop matrix B is defined as

$$b_{i,j} = \begin{cases} 1 & \text{is edge j is in loop i,} \\ 0 & \text{otherwise} \end{cases}$$

The weight vector w is defined as

$$w_i = \text{number of delays on edge i.}$$

Data-flow graph (DFG)

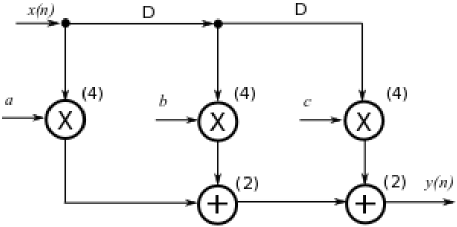


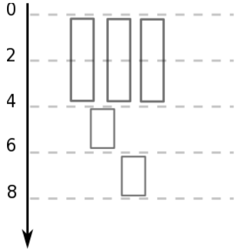
$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

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

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

Data-flow graph (DFG)

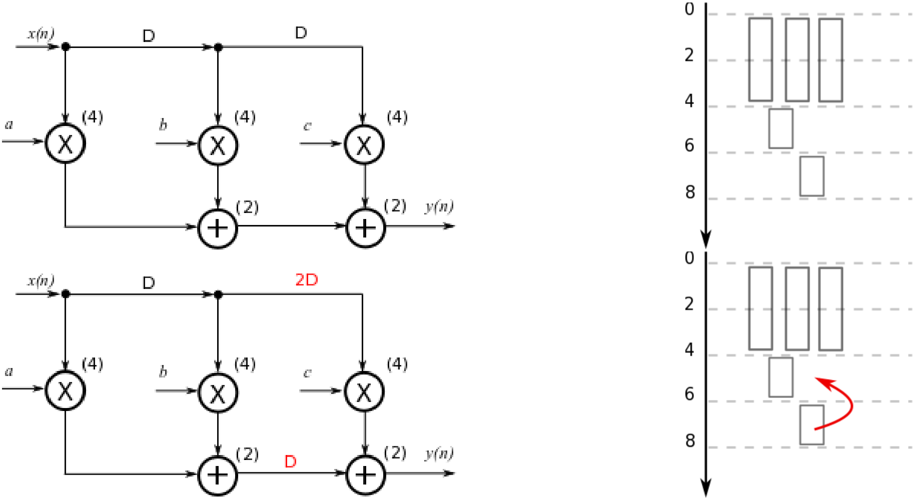




Any node fire whenever all the input data are available ! (thus many nodes can be fired simultaneously.

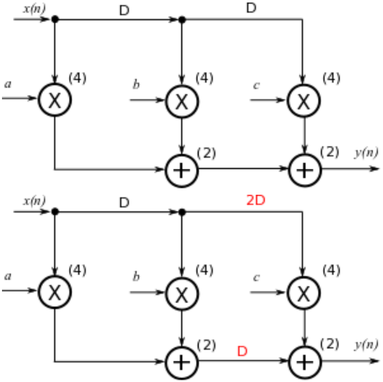
E. Lee, D. Messerschmitt, “Synchronous data flow”, Proceedings of the IEEE, Vol. 75, No.9, September 1987.

Data-flow graph (DFG)

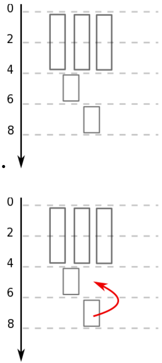


Data-flow graph (DFG)

- Iteration: execution of all comp.s in the alg. once.
- Iteration period (iteration rate): the time required for execution of one iteration of the alg.

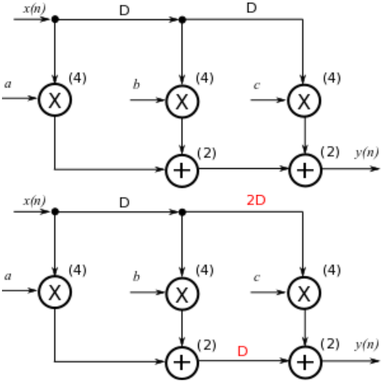


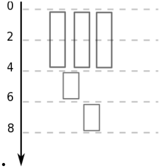
$T_{it} = 8u.t.$



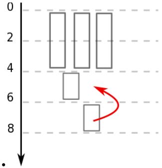
Data-flow graph (DFG)

- Iteration: execution of all comp.s in the alg. once.
- Iteration period (iteration rate): the time required for execution of one iteration of the alg.





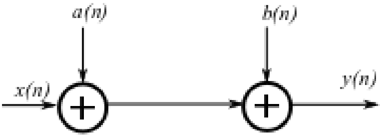
$T_{it} = 8u.t.$



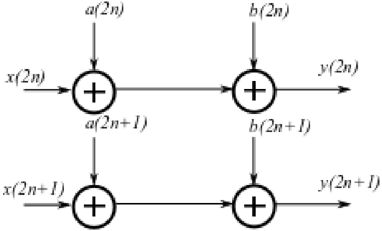
$T_{it} = 6u.t.$

Data-flow graph (DFG)

- Sampling rate (throughput):number of sample processed per second.
- Sampling period :time required to process of one sample.

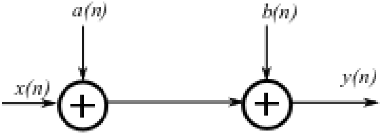


$T_s = 1u.t.$

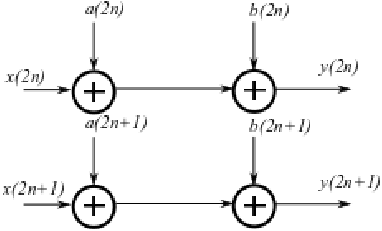


Data-flow graph (DFG)

- Sampling rate (throughput):number of sample processed per second.
- Sampling period :time required to process of one sample.

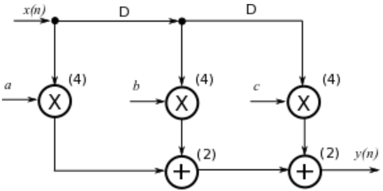


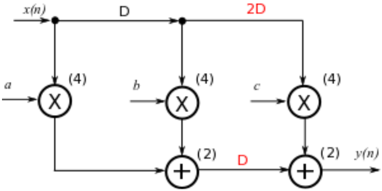
$T_s = 1u.t.$



$$T_s = \frac{1}{2}u.t.$$

- Critical path: the longest path between any two storage elements.

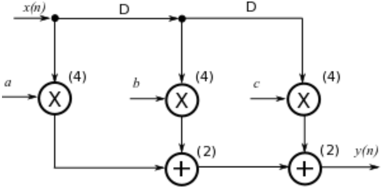


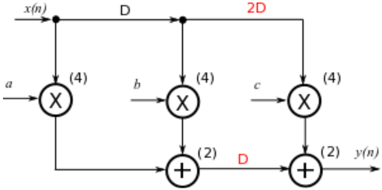


The critical path computation time determines the minimum feasible clock period of a DSP system!.

$T_{clk} \geq \max_{p:w(p)=0} \{d(p)\}$ such that p is a path $(V_i \xrightarrow{e_0} V_{i+1}\ V_i \xrightarrow{e_1} V_{i+2} \ldots \xrightarrow{e_L} V_N\ )$, $d(p) = \sum_{k=i}^{N} d(V_k)$ and $w(p) = \sum_{k=0}^{L} w(e_k)$.

- Critical path: the longest path between any two storage elements.





The critical path computation time determines the minimum feasible clock period of a DSP system!.

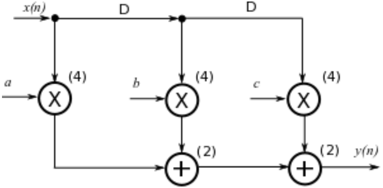
$T_{\text{critical}} = 2T_A + T_M = 6$ u.t.

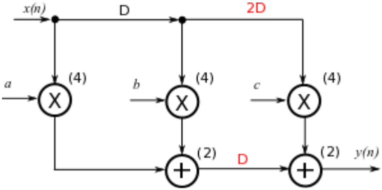
$T_{\text{critical}} = T_A + T_M = 4$ u.t.

$T_{clk} \geq \max_{p:w(p)=0} \{d(p)\}$ such that p is a path $(V_i \xrightarrow{e_0} V_{i+1}\ V_i \xrightarrow{e_1} V_{i+2} \ldots \xrightarrow{e_L} V_N\ )$, $d(p) = \sum_{k=i}^{N} d(V_k)$ and $w(p) = \sum_{k=0}^{L} w(e_k)$.

Data-flow graph (DFG)

- Critical path: the longest path between any two storage elements.





The critical path computation time determines the minimum feasible clock period of a DSP system!.

$T_{\text{critical}} = 2T_A + T_M$ =6 u.t. $T_{\text{critical}} = T_A + T_M$ =4 u.t.

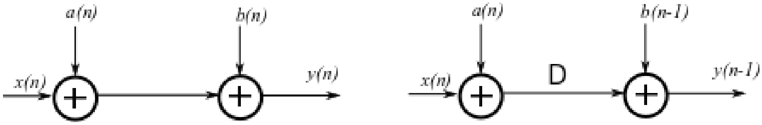
$T_{clk} \geq T_M + 2T_A$ $T_{clk} \geq T_M + T_A$

$T_{clk} \geq \max_{p:w(p)=0} \{d(p)\}$ such that p is a path $(V_i \xrightarrow{e_0} V_{i+1}\ V_i \xrightarrow{e_1} V_{i+2} \ldots$

$\xrightarrow{e_L} V_N$), $d(p) = \sum_{k=i}^{N} d(V_k)$ and $w(p) = \sum_{k=0}^{L} w(e_k)$.

Data-flow graph (DFG)

- Latency: the differnce between the time an output is generated and the time at its corresponding input was received.

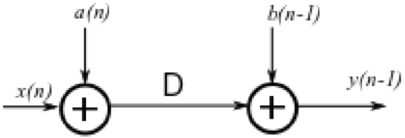


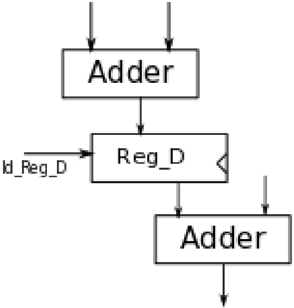
n	x	a	b	y
0	x(0)	a(0)	b(0)	y(0)=x(0)+a(0)+b(0)
1	x(1)	a(1)	b(1)	y(1)=x(1)+a(1)+b(1)

n	x	a	D	b	y
0	x(0)	a(0)	x(-1)+a(-1)	b(-1)	y(-1)=x(-1)+a(-1)+b(-1)
1	x(1)	a(1)	x(0)+a(0)	b(0)	y(0)=x(0)+a(0)+b(0)

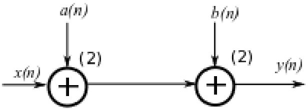
Data-flow graph (DFG)

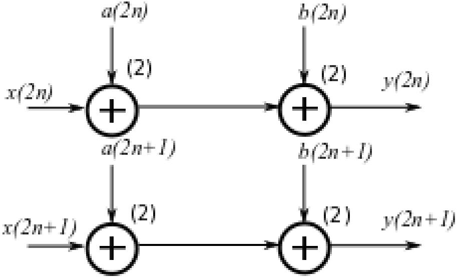
- Latency: the differnce between the time an output is generated and the time at its corresponding input was received.





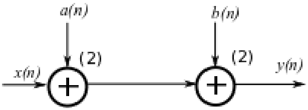
Data-flow graph (DFG)

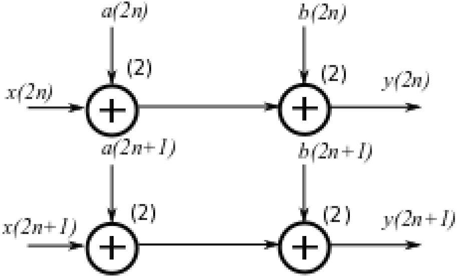




The Sampling rate is not the same as its clock rate (generally)!

Data-flow graph (DFG)



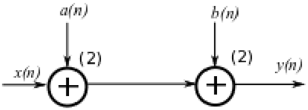


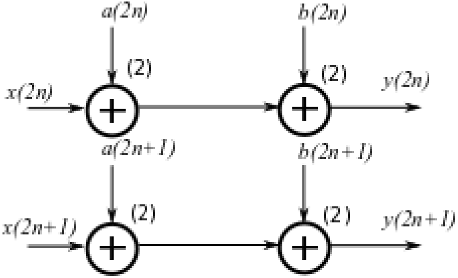
$T_{\text{critical}} = 2T_A$

$T_{\text{critical}} = 2T_A$

The Sampling rate is not the same as its clock rate (generally)!

Data-flow graph (DFG)

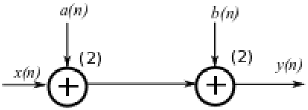


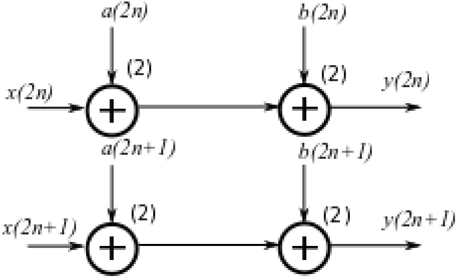


$T_{\text{critical}} = 2T_A$ $T_{\text{critical}} = 2T_A$

$$T_{clk} \geq 2T_A$$

The Sampling rate is not the same as its clock rate (generally)!





$T_{\text{critical}} = 2T_A$

$T_{\text{critical}} = 2T_A$

$$T_{clk} \geq 2T_A$$

$T_s = T_{clk}$

$T_s = \dfrac{T_{clk}}{2}$

The Sampling rate is not the same as its clock rate (generally)!

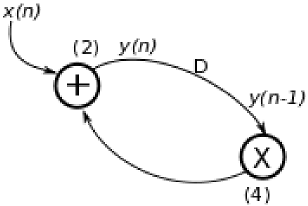
Data-flow graph (DFG)

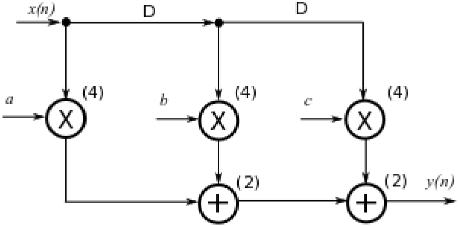
Nonrecursive DFG

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

Recursive DFG

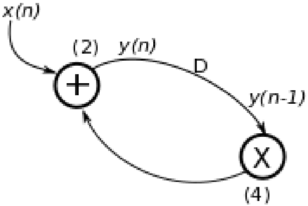
$$y(n) = x(n) + ay(n-1)$$





Data-flow graph (DFG)

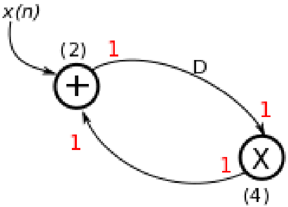
$$y(n) = x(n) + ay(n-1)$$



Any node fire whenever all the input data are available ! (thus many nodes can be fired simultaneously (leading to concurrency).

Each edge describes a precedence constraint between two nodes:
intra-iteration precendence constraint : if the edge has zero delays.
inter-iteration precendence constraint : if the edge has one or more delays.
x : $ay(n-1)$
$+$: $x(n) + ay(n-1)$

Synchronous data flow: the number of tokens produced/consumed is know beforehand (a priori)!



single-rate SDFG and multi-rate SDFG (multi-rate to single-rate see 39.)