Fully Dedicated Architectures

Lecture 4

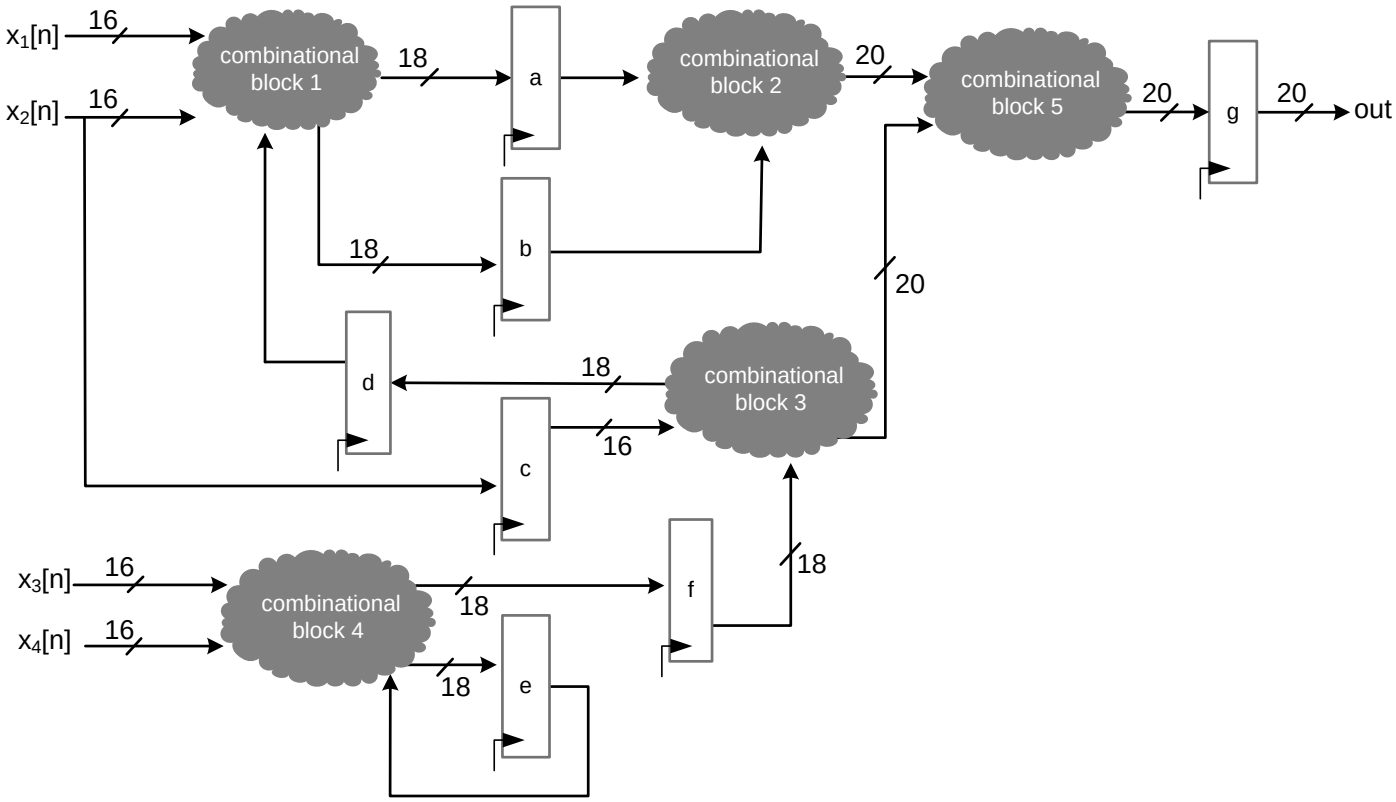
Dr. Shoab A. Khan

Synchronous Digital Logic at RTL

- Digital designs are convenient to handle if they are synchronous
- The ‘synchronous’ signifies that all changes in the digital logic are controlled by one or more circuit clocks
- The ‘digital’ means that the design deals only with digital signals.
- Synchronous digital logic is designed at the register transfer level (RTL) of abstraction

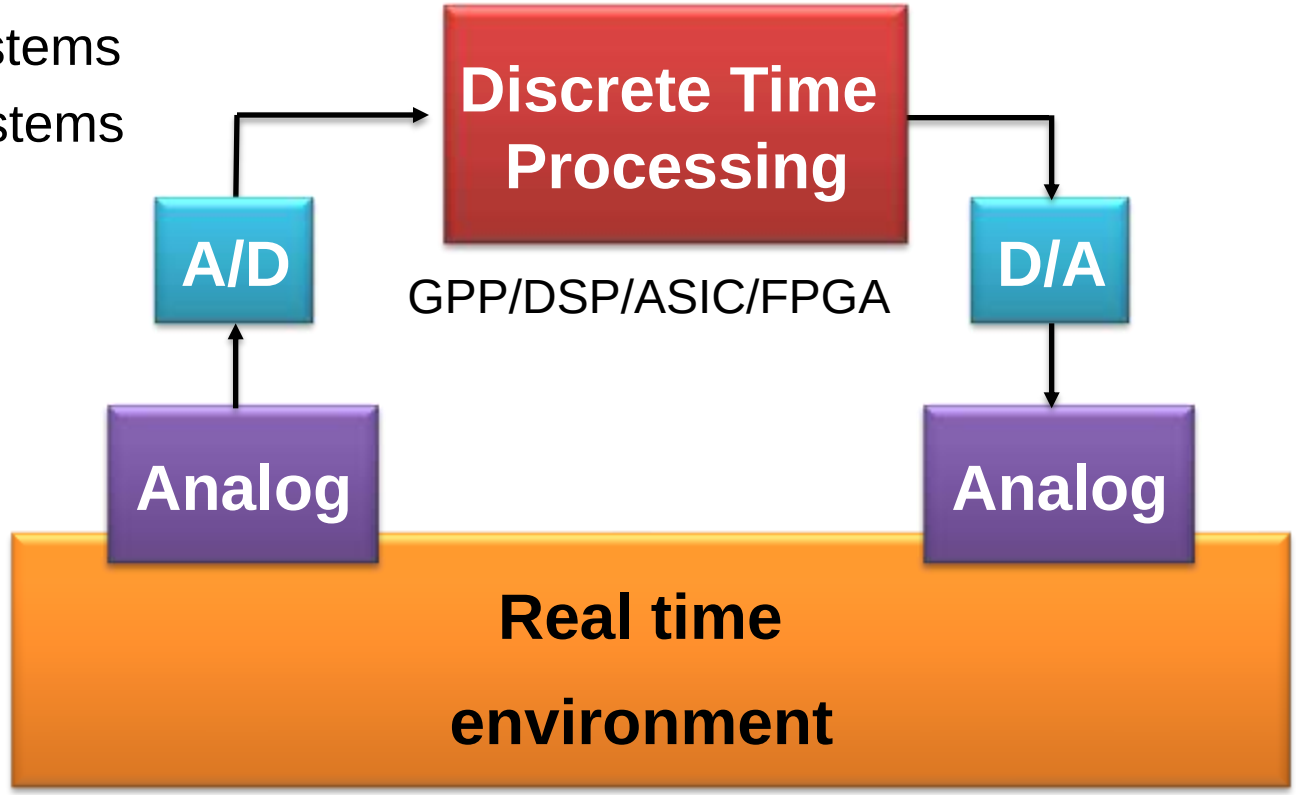
Synchronous Digital Logic at RTL

- At RTL a synchronous digital design consists of combinational logic blocks and synchronously clocked registers



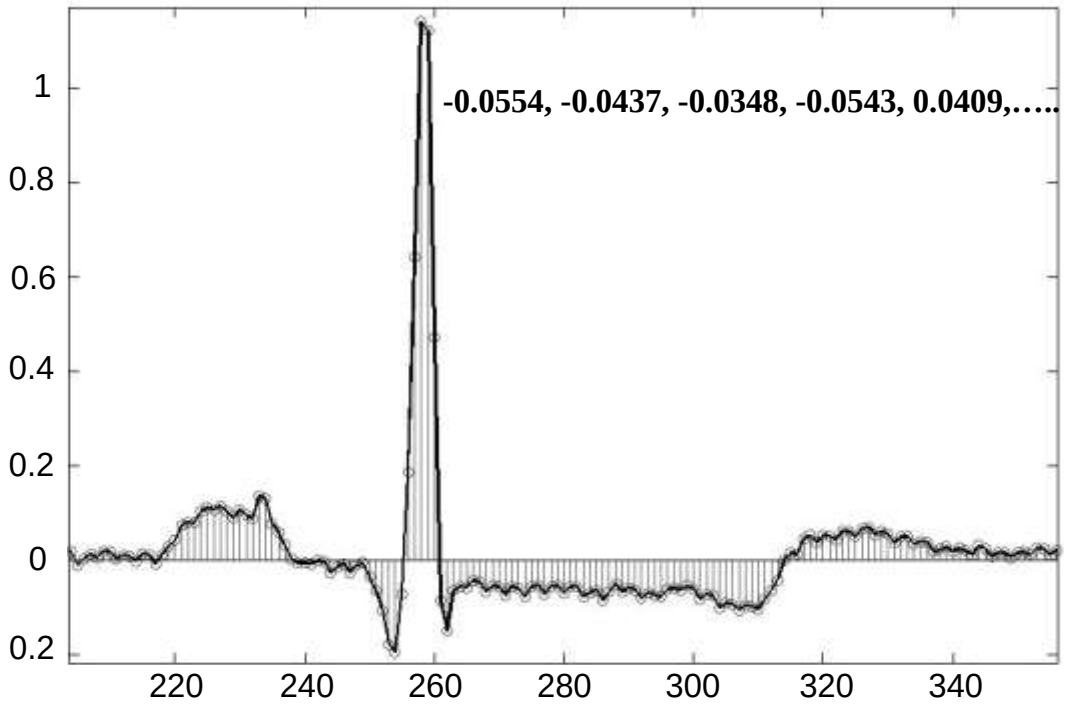
Hard Real-Time Systems

- Hard real-time system constraints the design to process the input samples at the input sampling rate and produces output samples at the output sampling rate
- Examples:
  - Communication Systems
  - Multimedia Systems
  - Biomedical Systems



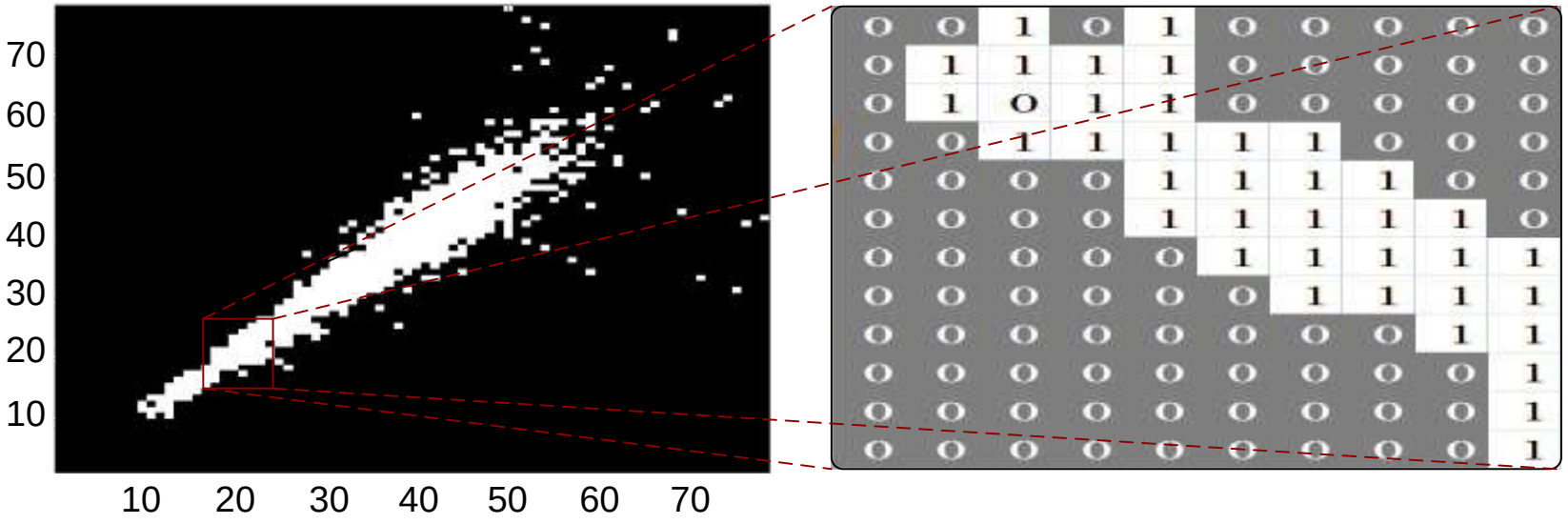
System for discrete 1-D Signal

- For 1-D real-time signals only the amplitude varies in time
  - Example: Voice, ECG, signals from temp and pressure sensors



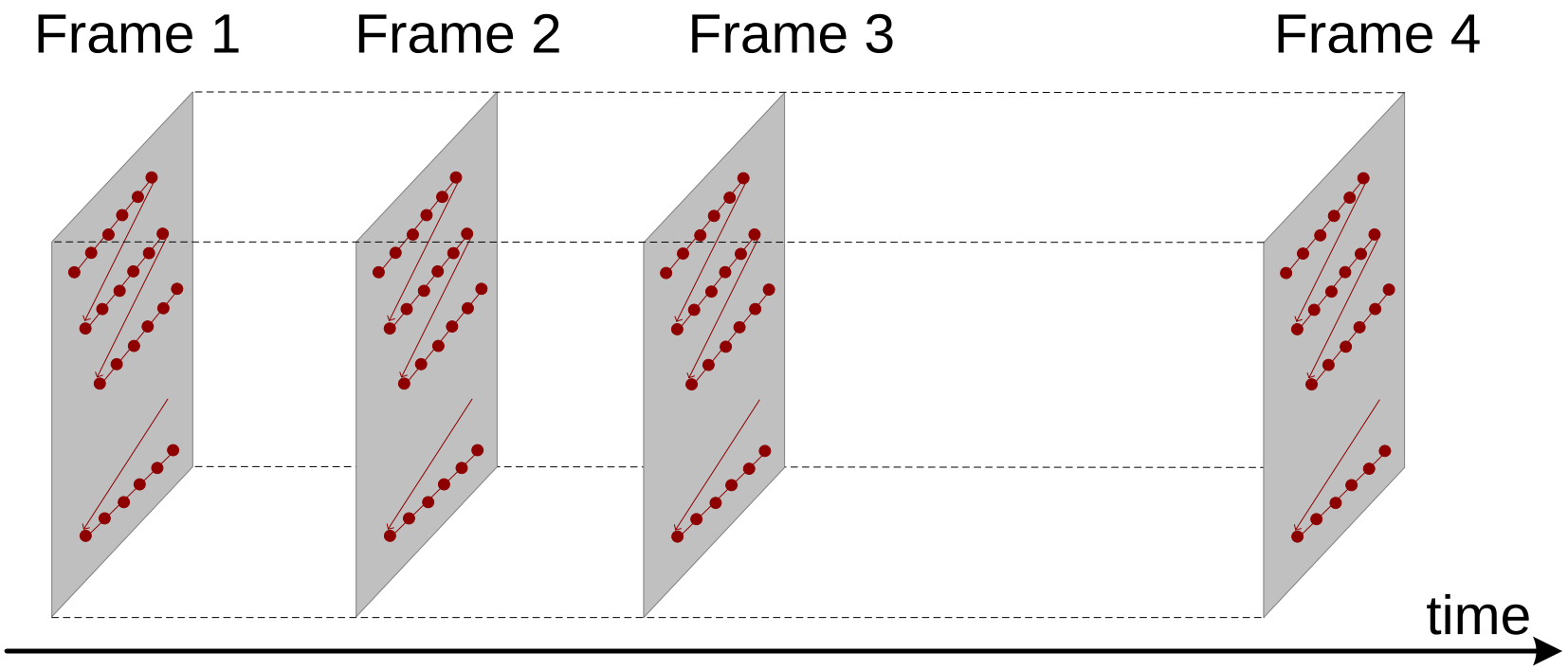
A Discrete 2-D Signal: An Image

- 2-D discrete time signals have a matrix of data values



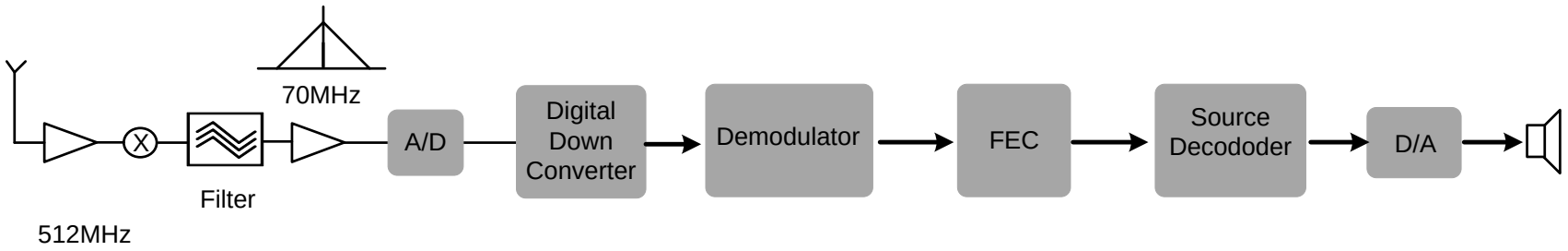
A Discrete time Video

- A video is a 2-D signal that varies in time
  - A real-time video digital signal processor processes a number of image frames per second



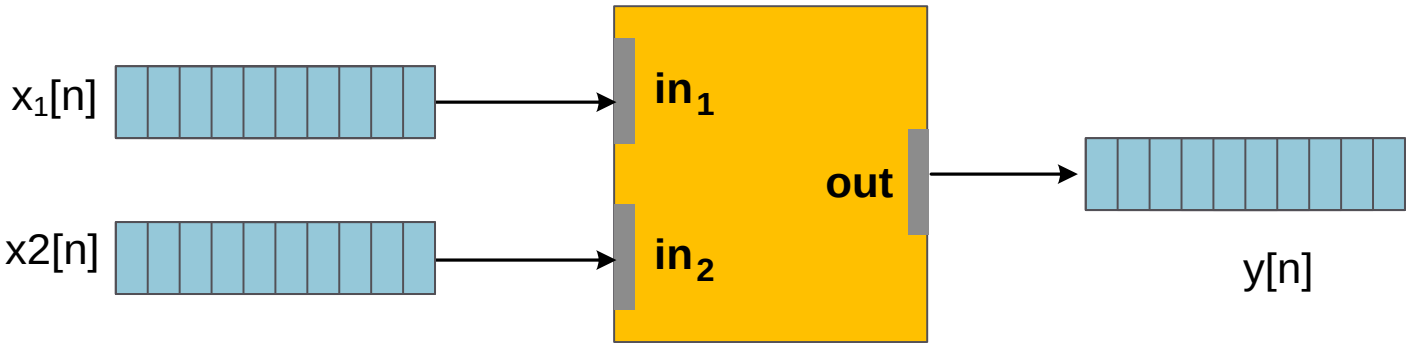
Digital Design of Real-time Systems

- A digital communication receiver is a good example of real-time system
  - Digitizes an analog signal at IF
  - Digitally down converts the signal
  - Performs demodulation and error correction
  - Performs source decoding
  - For high data rates, all these operations are performed in a digital design
  - For digital design, a top level framework is important
    - KPN is a framework that facilities top level design for multi-rate real-time signal processing systems



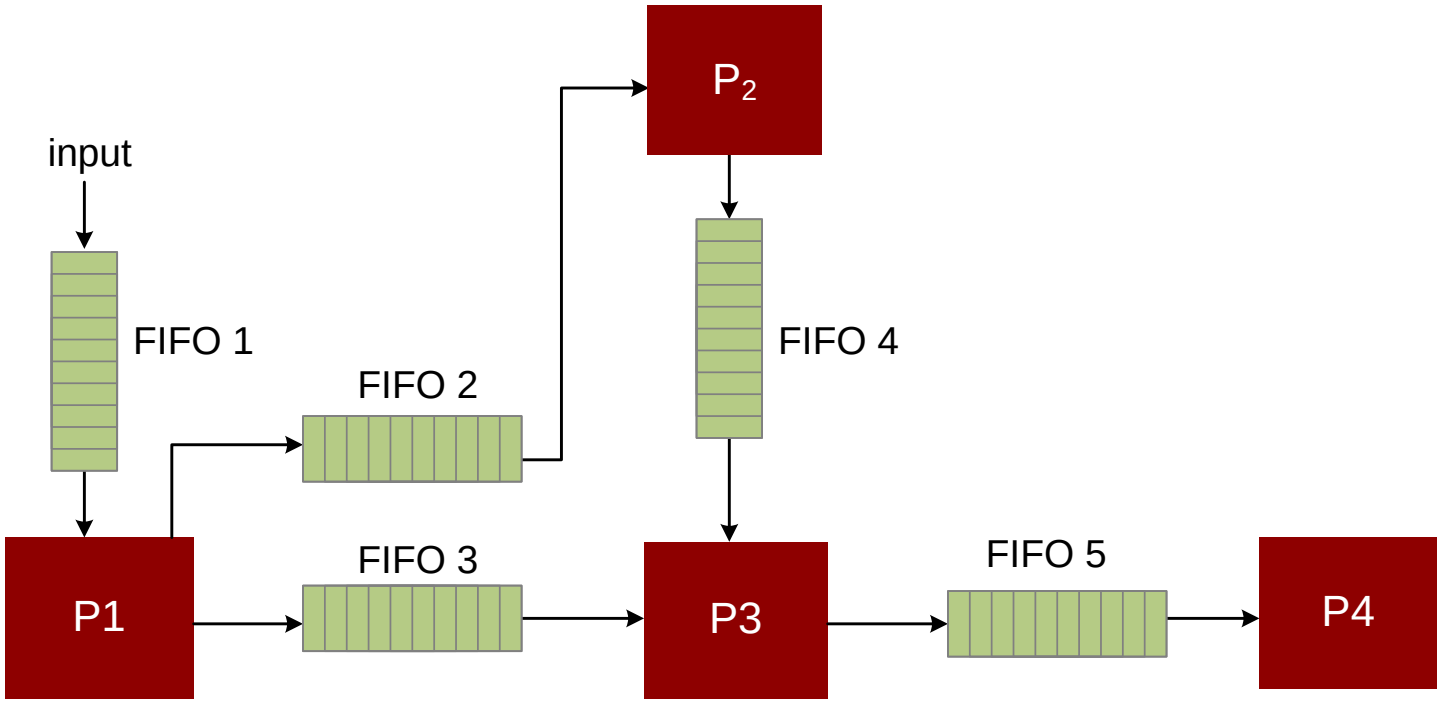
Kahn Process Networks (KPN)

- A system implementing a streaming application is best represented by KPN
- A KPN consists of autonomously running components
  - Takes inputs from FIFOs on input edges
  - Generates output in FIFOs on the output edges
- Nodes execute/fire after their input FIFOs have acquired enough samples/tokens
- Synchronization in KPN is achieved using
  - Blocking read operation
  - Non-blocking writes in FIFO



Example of a KPN

- Four processors P_1,P_2,P_3 and P_4
- A Processor fires if it has enough tokens/data at its input FIFOs
  - Example: P3 fires if there are input samples in FIFO4 and FIFO3



KPN for Modeling Streaming Applications

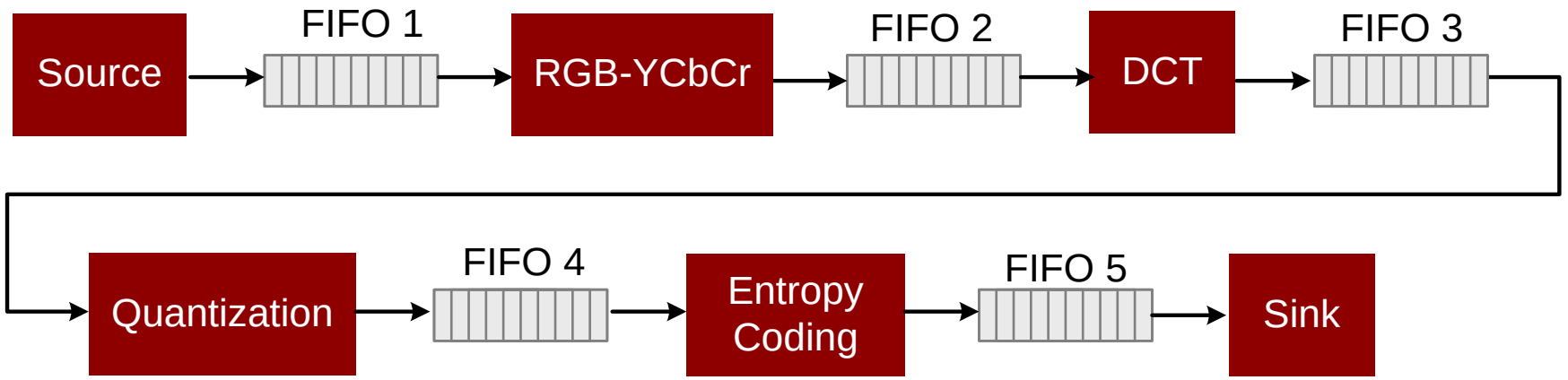
- For mapping applications in HW using KPN, it is recommended that High-level language code should also be broken down into blocks
  - Clearly identify streams of input and output data

Code Example

```
N = 4*16;
K = 4*32;

 % source node
for i=1:N
   for j=1:K
      x(i,j)= src_x();
   end
end

% processing block 1
for i=i:N
   for j=1:K
      y1(i,j)=func1(x(i,j));
   end
end
```

```
 % processing block 2
for i=1:N

 for j=1:K
      y2(i,j)=func2(y1(i,j));
   end
end

% sink block
m=1;
for i=1:N
   for j=1:K
      y(m)=sink (x(i,j), y1(i,j),
y2(i,j));
      m=m+1;
   end
end
```

Building Blocks in JPEG compression

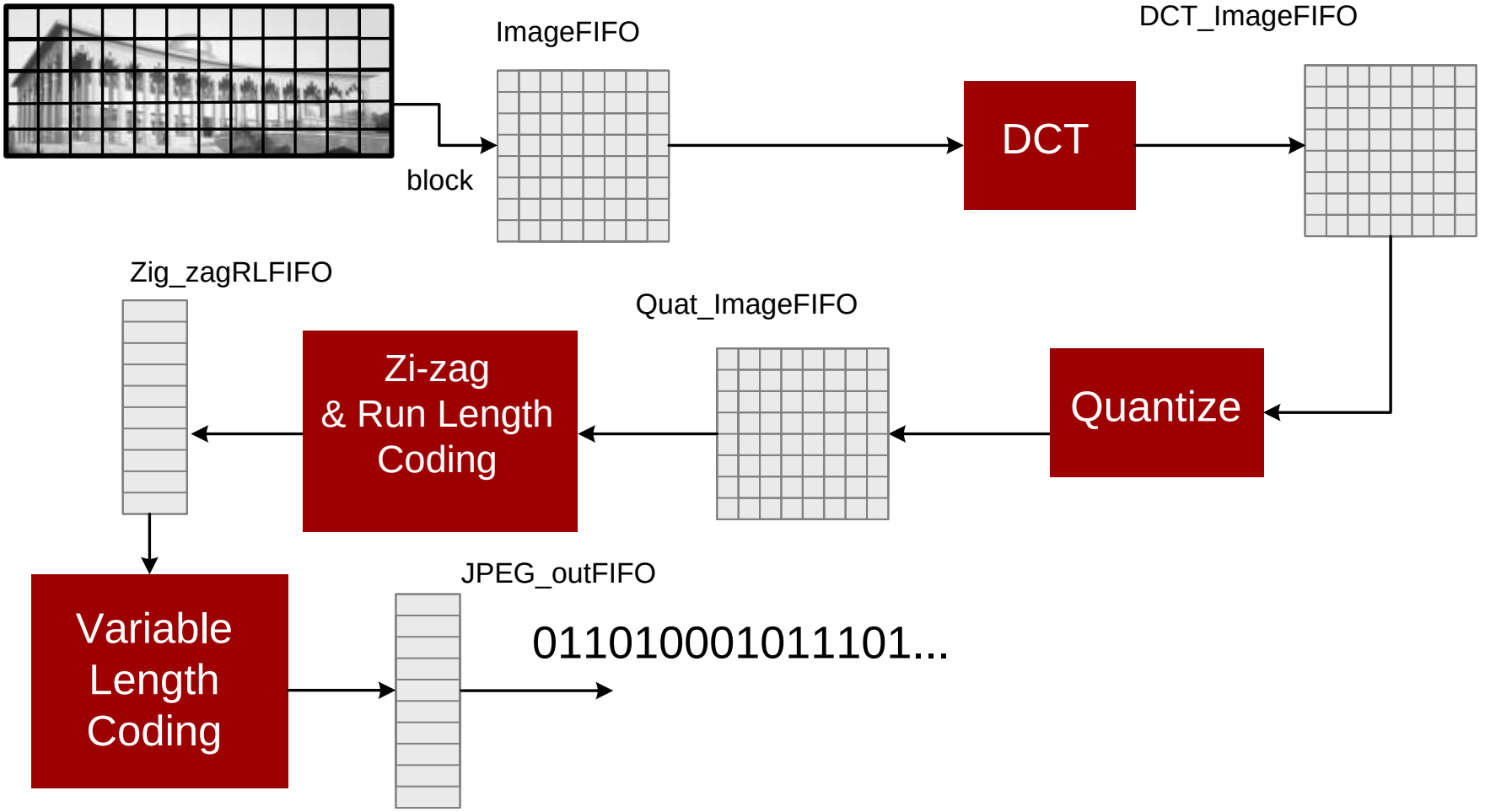


JPEG Code Matlab Code for Mapping as KPN

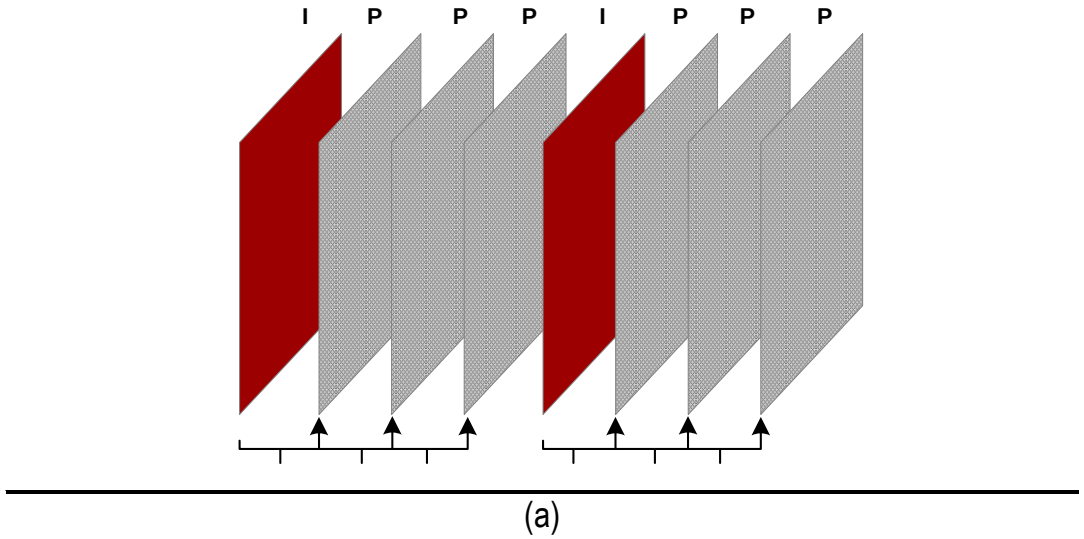
```
Q=[           8  36 36 36 39 45 52 65;...
36 36 36 37 41 47 56 68;...
36 36 38 42 47 54 64 78;...
36 37 42 50 59 69 81 98;...
39 41 47 54 73 89 108 130;...
45 47 54 69 89 115 144 178;...
53 56 64 81 108 144 190 243;...
65 68 78 98 130 178 243 255];
BLK_SZ = 8; % block size


imageRGB = imread('peppers.png'); % default image
figure, imshow(imageRGB);
title('Original Color Image');
ImageFIFO = rgb2gray(RGB);
figure, imshow(ImageFIFO);
title('Original Gray Scale Image');
```

```
fun = @dct2;
DCT_ImageFIFO = blkproc(ImageFIFO,[BLK_SZ BLK_SZ],fun);
figure, imshow(DCT_ImageFIFO);
title('DCT Image');


QuanFIFO = blkproc(DCT_FIFO, [BLK_SZ BLK_SZ], 'round(x./P1)', Q);
figure, imshow(QuanFIFO);


fun = @zigzag_runlength;
zigzagRLFIFO = blkproc(QuanFIFO, [BLK_SZ BLK_SZ], fun);


% variable length coding using Huffman tables
JPEGoutFIFO = VLC_huffman(zigzagRLFIFO, Huffman_dict);
```

JPEG KPN


ImageFIFO
block
DCT
DCT_ImageFIFO
Quantize
Quat_ImageFIFO
Zi-zag
& Run Length
Coding
Zig_zagRLFIFO
Variable
Length
Coding
JPEG_outFIFO
011010001011101...

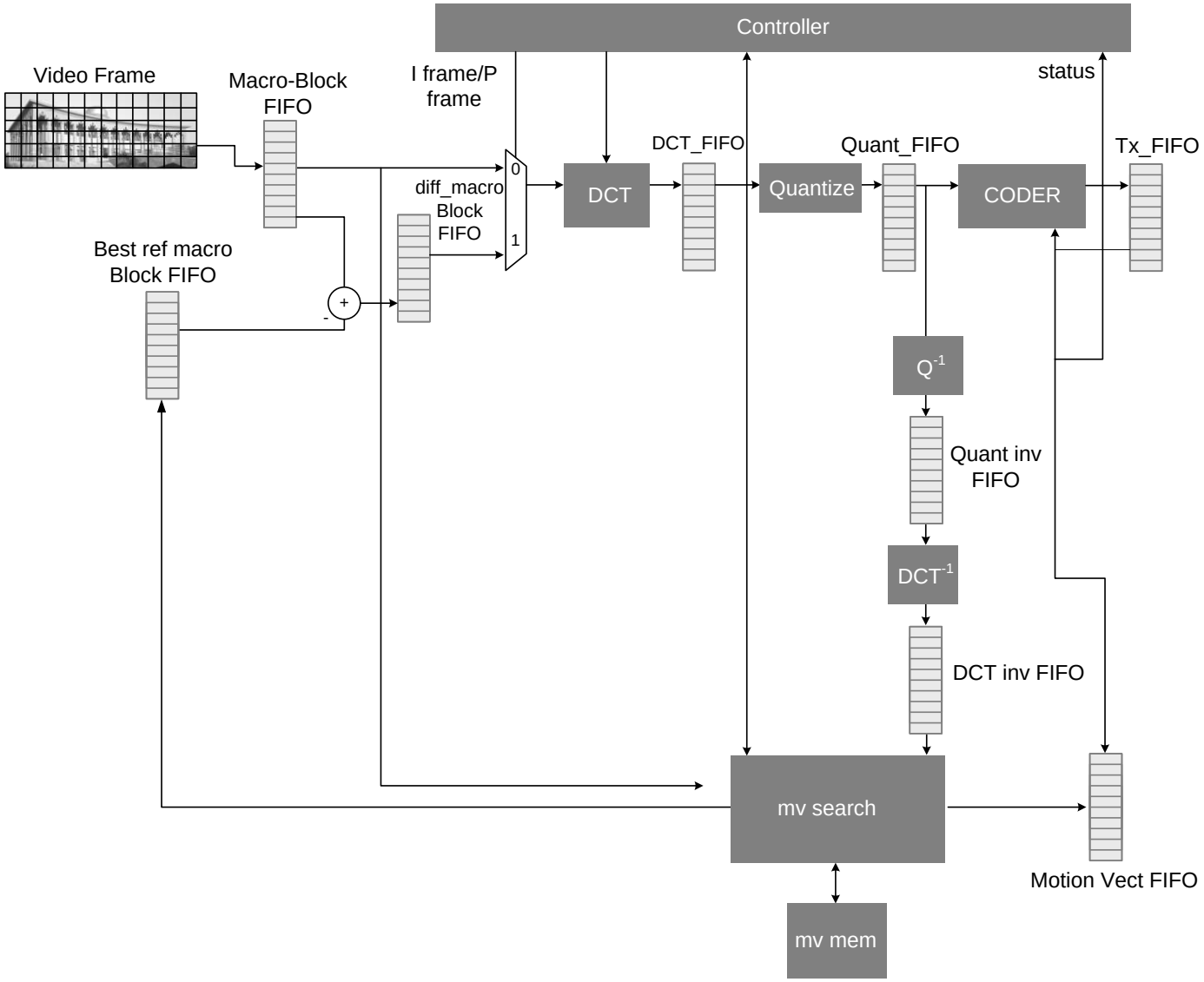
Example: MPEG Encoding



Steps in Video Encoding

- Video is coded as a mixture of
  - Intra-Frames (I-Frames)
  - Inter- or Predicted Frames (P-Frames).
- I-Frame is coded as JPEG frame
- The I-Frame and all other frames are also decoded for coding of P-Frame.
- For P-Frame
  - Frame is divided into sub-blocks
  - Each target block is searched in the previously self-decoded reference frame
  - The difference of target and reference block is coded as a JPEG block
  - The motion vector is also coded and transmitted with the coded block

KPN Architecture for MPEG Encoder

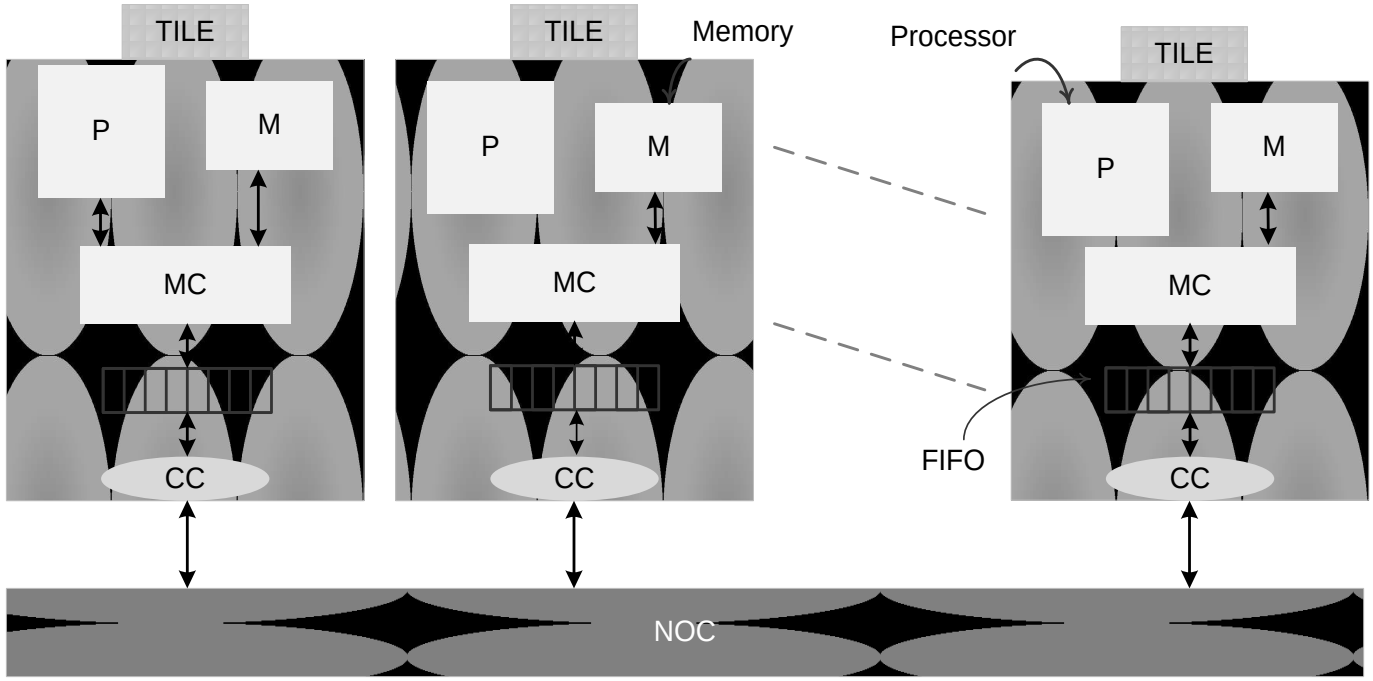


Limitations of KPN

- Reads to follow FIFO operation
  - Read in sequential order from the first value to the last
  - The values once read from the FIFO are deleted
  - All values in the FIFO to be read
- Many Signal Processing Algorithms do not adhere to these strict rules
  - FFT requires bit reverse reading of data
  - Convolution requires multiple read of same value many times
  - Fractional re-sampling may require sparse reading of data
- Similarly the outputs may not be produced in sequential order

MPSoC

- Multi-processor System on Chip (MPSoC) employs some from of KPN
- Local memory in each PE enables the design to counter KPN key limitations
  - The PE (P) first copies the data from FIFO to local memory (M) and then processes it and writes the results in the memory
  - The results are then copied in FIFO for passing to the next PE



Case Study

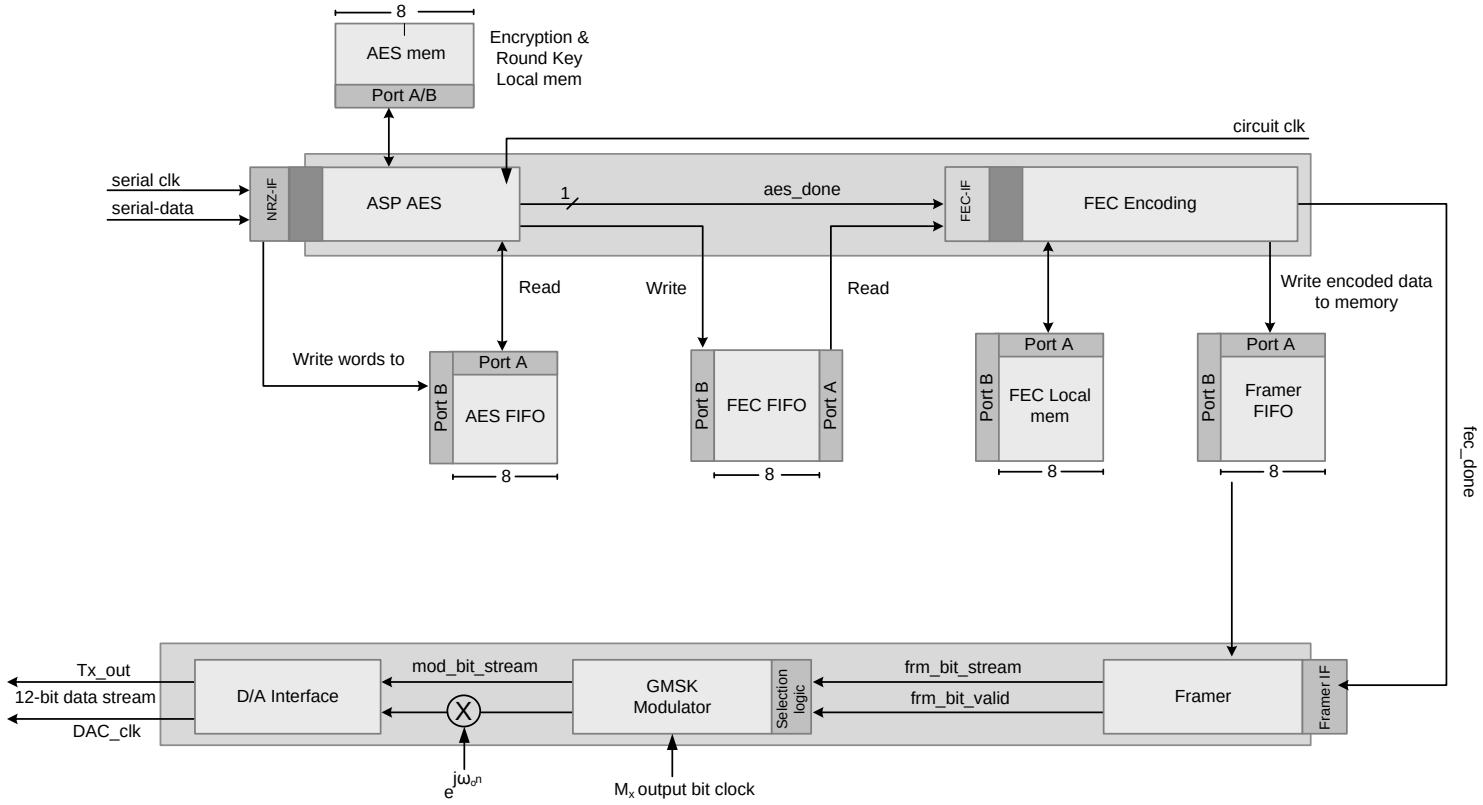
A GMSK Transmitter mapped on KPN architecture

The transmitter performs

- AES Encryption
- Product Turbo Code Encoding for FEC
- Framing for Synchronization
- GMSK Modulation
- Digital Mixing

GMSK mapping on KPN

- Processing Elements (PEs) of AES, FEC, Framer, Modulator & mixer are shown
- FIFOs & PEs implements the transmitter as modified KPN
  - Each PE has its own memory to counter KPN limitations

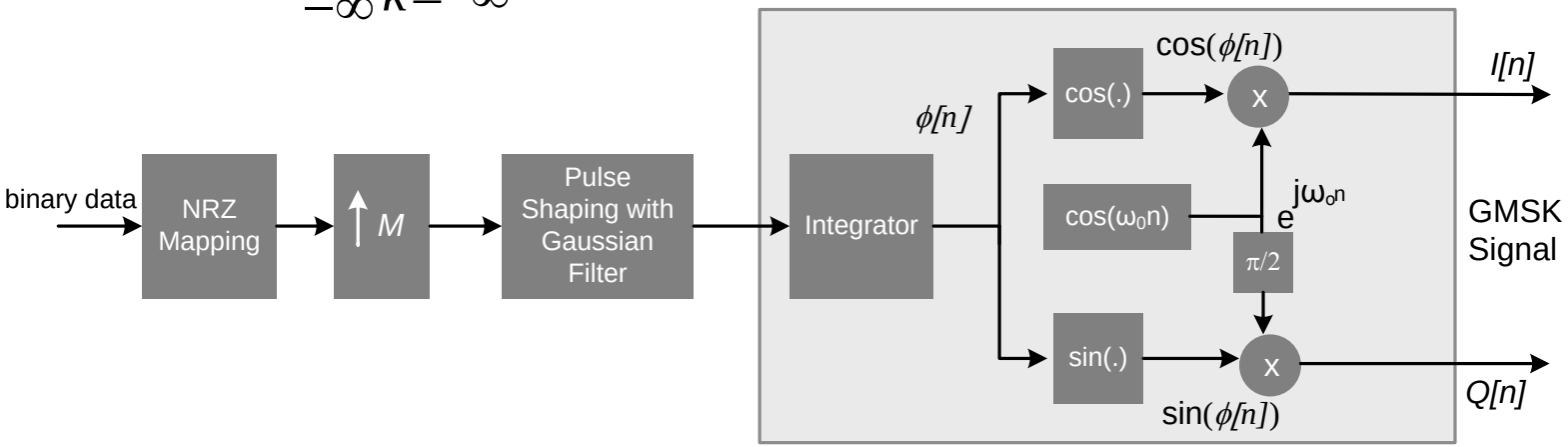


Sample by Sample Processing

- The last PE performs sample by sample GMSK modulation
  - Does not require any FIFO

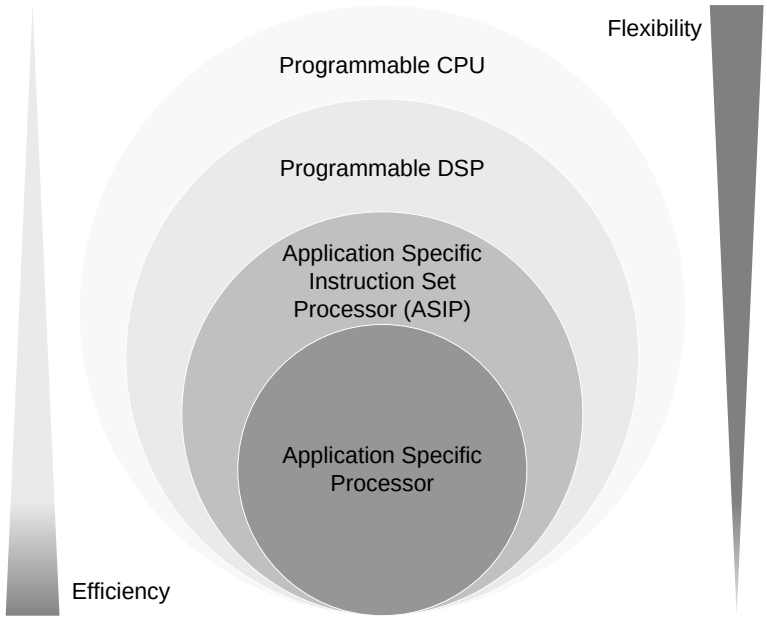
$$x(t) = \sqrt{2P_c}\cos(2\pi f_c t + \varphi_s(t))$$

$$\varphi_s(t) = 2\pi f_d \int_{-\infty}^{t} \sum_{k=-\infty}^{\infty} a_k g(\nu - kT) d\nu$$



Design Options

- A PE in KPN can be implemented using one of the design options:
  - Programmable CPU
  - Programmable DSP
  - Application Specific Instruction Set Processor (ASIP)
  - Application Specific Processor (ASP)
- These options provide tradeoff between flexibility and efficiency
- From digital design our interest is in the last two options
- For HW mapping, most of the DSP algorithms are first represented as some type of graph

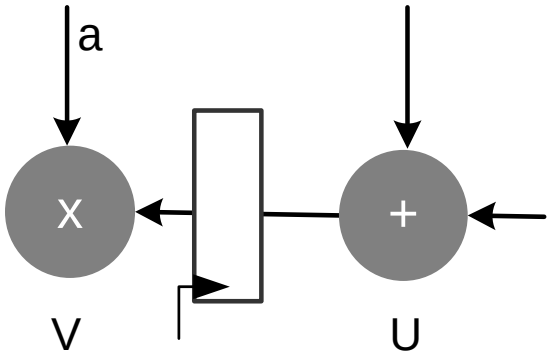


Graphical Representation of DSP Algorithms

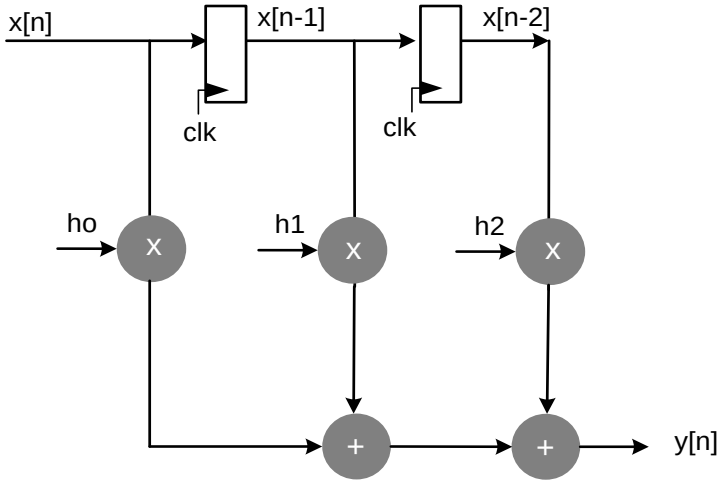
- Most of the DSP algorithms are represented in some form of graphs
- A graphical representation is well suited for subsequent HW mapping
- Graphical methods of representations are
  - Block Diagram
  - Signal Flow Graph
  - Data Flow Graph / Data Dependency Graph

Block Diagram

- A block diagram consists of functional blocks connected with directed edges
  - A connected edge represents flow of data from source block to destination block



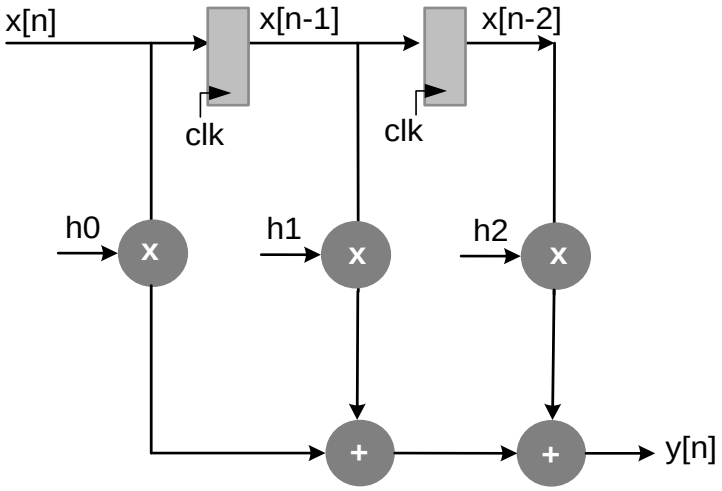
Source and destination nodes with a delay element on the joining edge



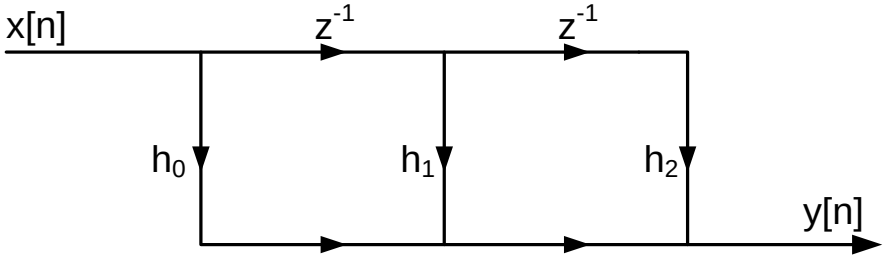
Dataflow graph representation of a 3-coefficient FIR filter

Signal Flow Graph

- A signal flow graph (SFG) is a simplified version of a block diagram.
  - The operation of multiplication with a constant and delays are represented by edges,
  - The nodes represent addition, subtraction and input and output (I/O) operations.



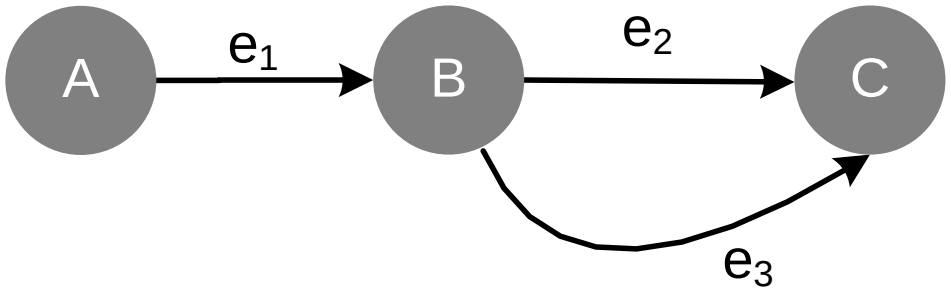
Block Diagram



Signal flow graph representation of a 3-coefficient FIR filter

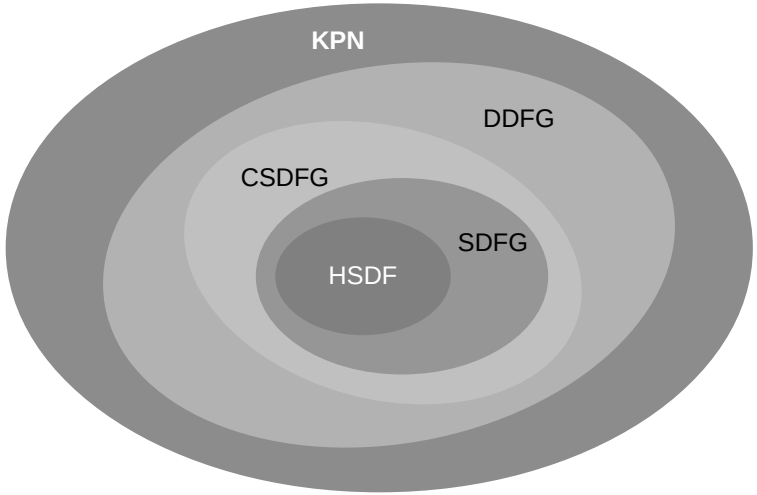
Dataflow Graph or Data Dependency Graph

- A DSP algorithm is described by a directed graph $G = < V, E >$
  - Node $v \in V$ represents a computational unit or, in a hierarchical design, a sub-graph
  - Edge $e \in E$ represents either a FIFO buffer or just precedence of execution



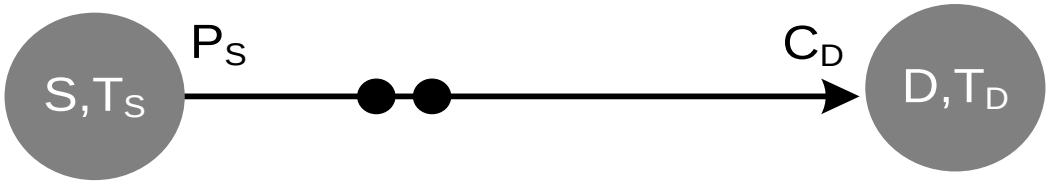
Scope of subclasses of graph representation

- KPN is most generalized
- For more specific representations use
  - Dynamic DFG (DDFG)
  - Cyclo-static DFG (CSDFG),
  - Synchronous DFG (SDFG)
  - Homogenous SDFG (HSDFG)



Synchronous Dataflow Graph

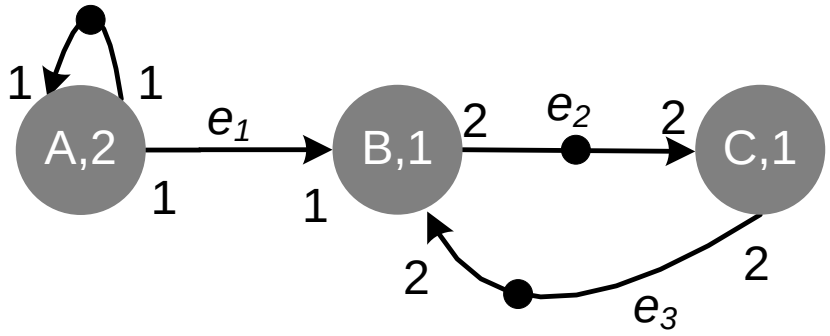
- The number of tokens consumed by a node on each of its edges, and number of tokens produced on its output edges, are known a priori



SDFG with nodes S and D, P_S and C_D are production and consumption rates of tokens of node S and D respectively

DFG and Topology Matrix

- An SDFG can also be represented by a topology matrix
- Column represents a node and row corresponds to an edge



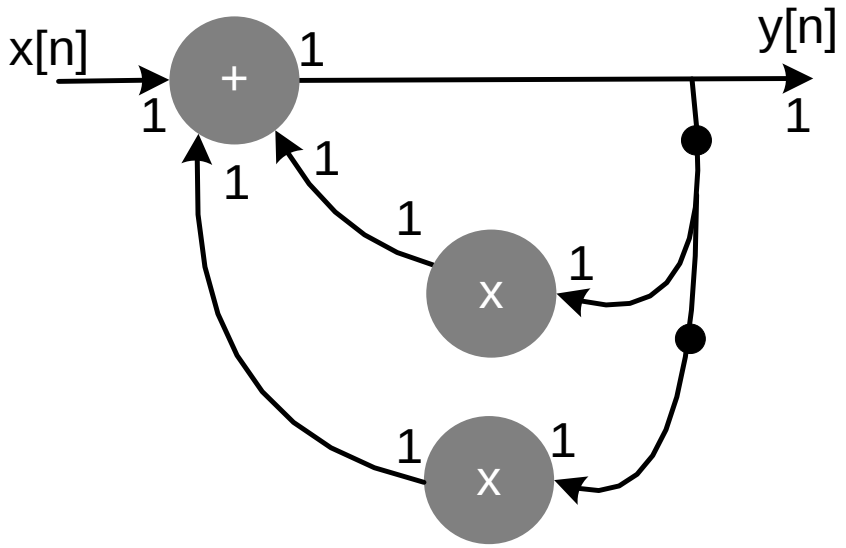
(a)

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

(b)

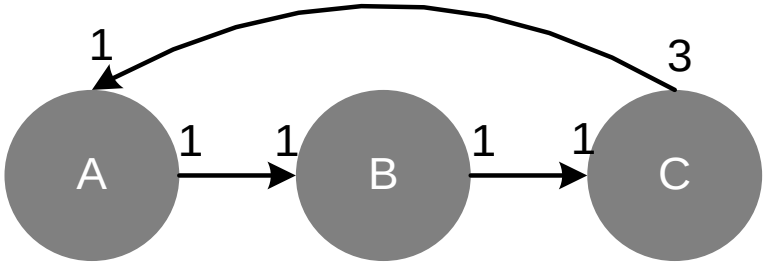
Example: IIR Filter as an SDFG

$$y[n] = x[n] + a_1 y[n-1] + a_2 y[n-2]$$

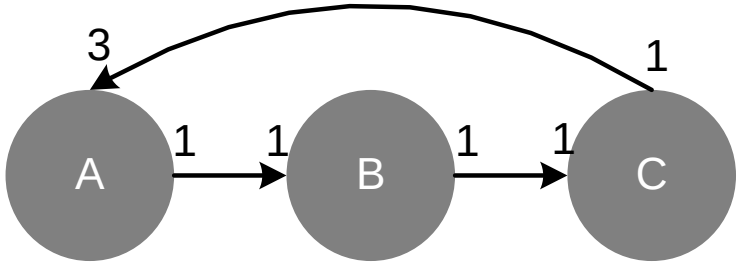


Consistent and Inconsistent SDFG

- An SDFG is consistent if it is correctly constructed.
- An SDFG is inconsistent
  - If some of its nodes starve for data at its input to fire
    - SDFG of Fig (b) is an example
  - Or some of its edges may need unbounded FIFOs for storage
    - SDFG of Fig (a) is an example



(a)



(b)

Balanced Firing Equations

- For designing a scheduler that synchronously makes all the nodes to fire, a repetition vector is computed
- The repetition vector consists of $f_1 \ldots f_N$ values
- The scheduler makes node *i* to fire f_i times for synchronous operation
- Following set of equations are solved

$$f_S p_S = f_D P_D$$

$$f_1 p_1 = f_2 c_2$$

$$f_2 p_2 = f_3 c_3$$

$$\vdots$$

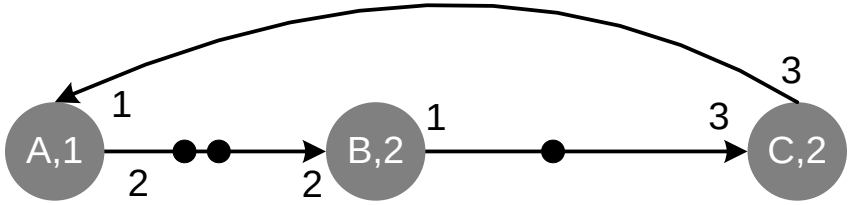
$$f_{N-1} p_{N-1} = f_N c_N$$

HW Mapping and Scheduling

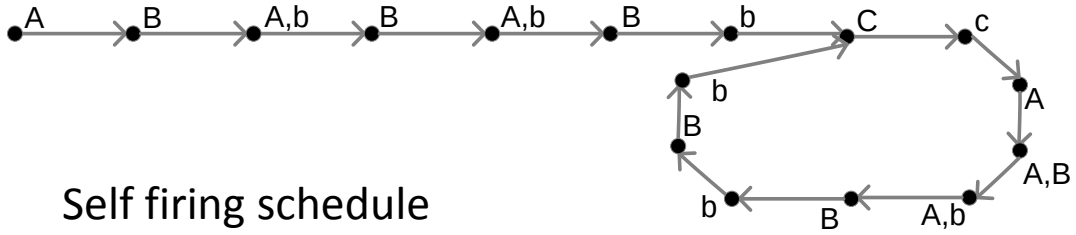
- Repetition Vector based
  - Large buffer sizes
  - Multi firing of each node based on firing vector
- Self Timed
  - Minimum buffer size
  - Transition phase then periodically repeat a pattern of firing

Self-timed Firing

Example of self-timed firing where a node fires as soon as it has enough tokens on its input

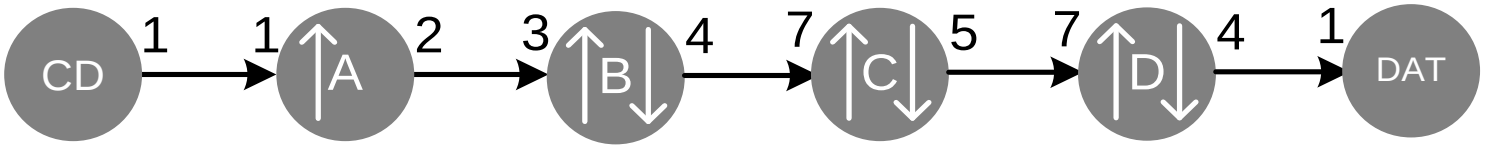


An SDFG



Self firing schedule

Example: CD quality sound to DAT conversion



- CD-quality audio sampled at 44.1 kHz to a digital audio tape (DAT) that records at 48 kHz

$$\frac{f_{DAT}}{f_{CD}} = \frac{480}{441} = \frac{160}{147} = \frac{2}{1} = \frac{4}{3} = \frac{5}{7} = \frac{4}{7}$$

- Firing vector [147 98 56 40] for firing of nodes A, B, C and D, respectively
  - Buffer sizes on the edges 147, 147x2, 98x4, 56x5 and 40x4
- To fire as soon as it gets enough samples at its input port
  - Buffers of sizes 1, 4, 10, 11 and 4

Real-time Processing

- **Dedicated Fully Parallel Implementation**
  - **clock frequency = sample frequency**
  - Each operator in the DFG is mapped on related hardware unit, e.g. addition, multiplication and delay on adder, multiplier and register respectively
- **Most designs: time multiplexing**
  - **clock frequency >= 2 x sample frequency**

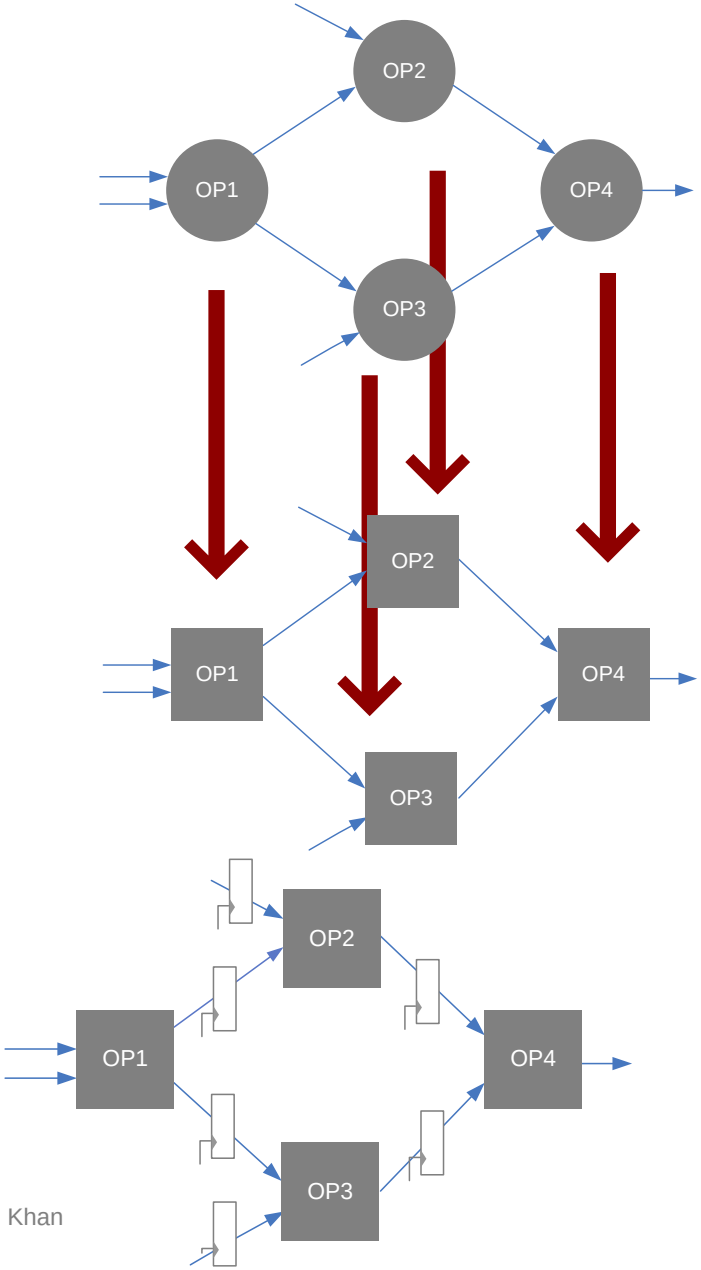
  $$\text{sample frequency} = \frac{\text{clock frequency}}{\text{available number of clock cycles per sample}}$$
- **Pipeline / Parallel Designs**
  - **clock frequency < sample frequency**

Dedicated Fully Parallel Architecture

clock freq = sampling freq

One-to-one mapping

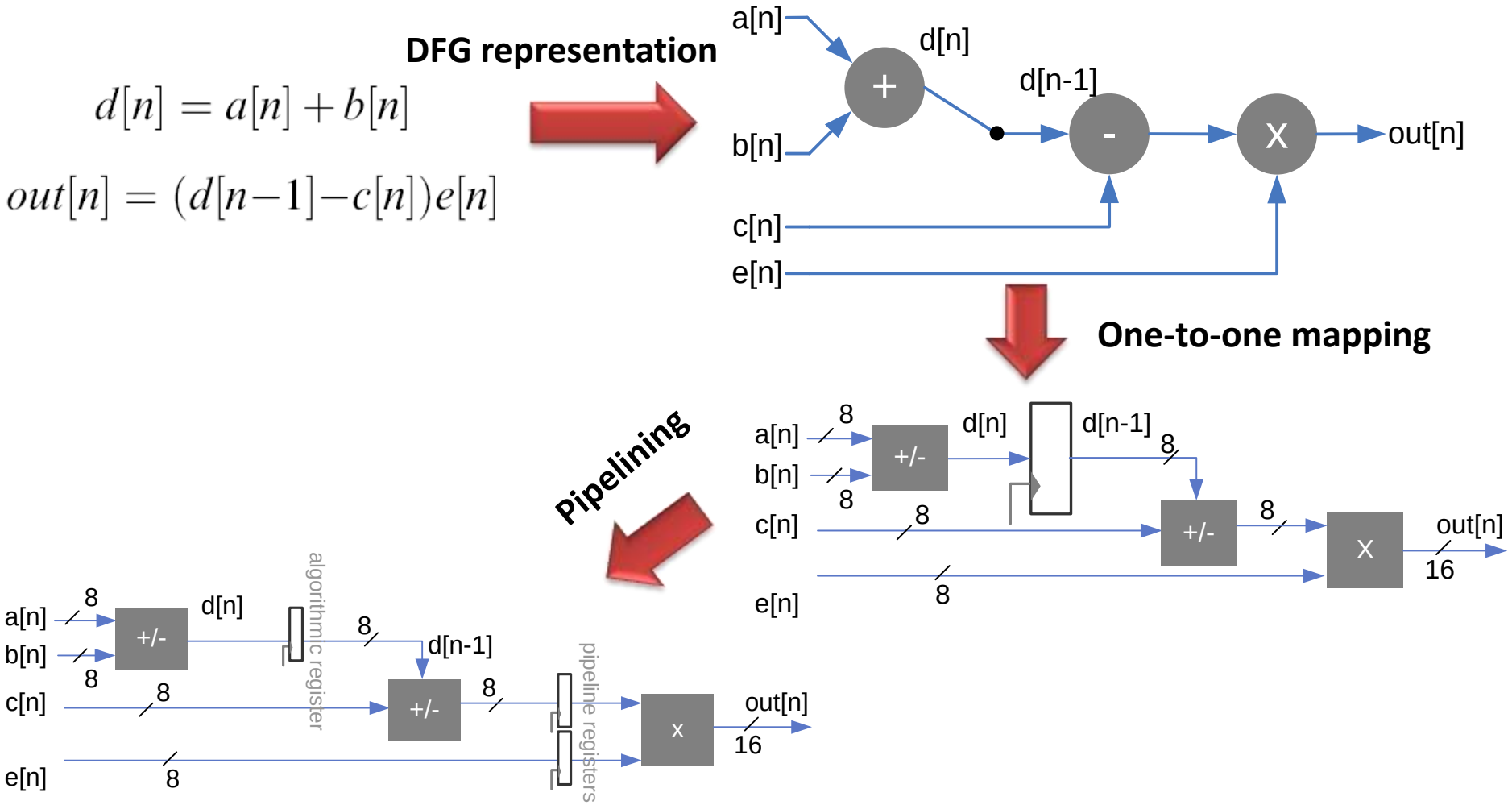
- Step 1: DFG representation
- Step 2: One-to-one mapping of mathematical operators to HW operator
- Step 3: Pipelining and retiming for meeting timing constraint



Example: Dedicated Implementation

$$d[n] = a[n] + b[n]$$

$$out[n] = (d[n-1] - c[n])e[n]$$



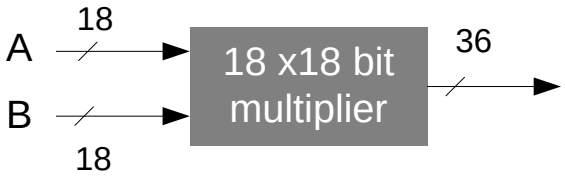
HW Components for Fully Dedicated Architecture

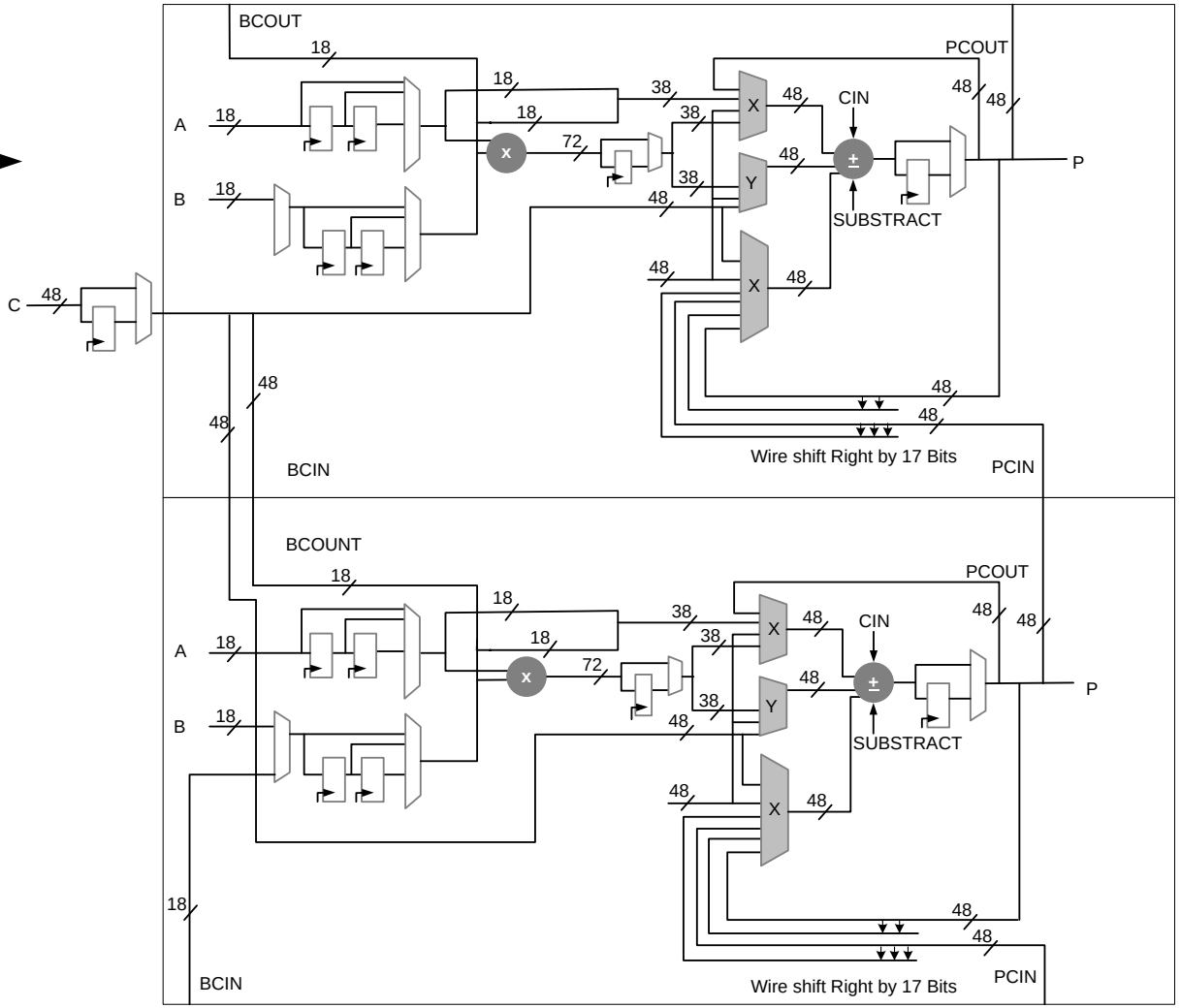
- Adder
- Multiplier
- Barrel Shifter
- Register File

Embedded Functional Units in FPGAs

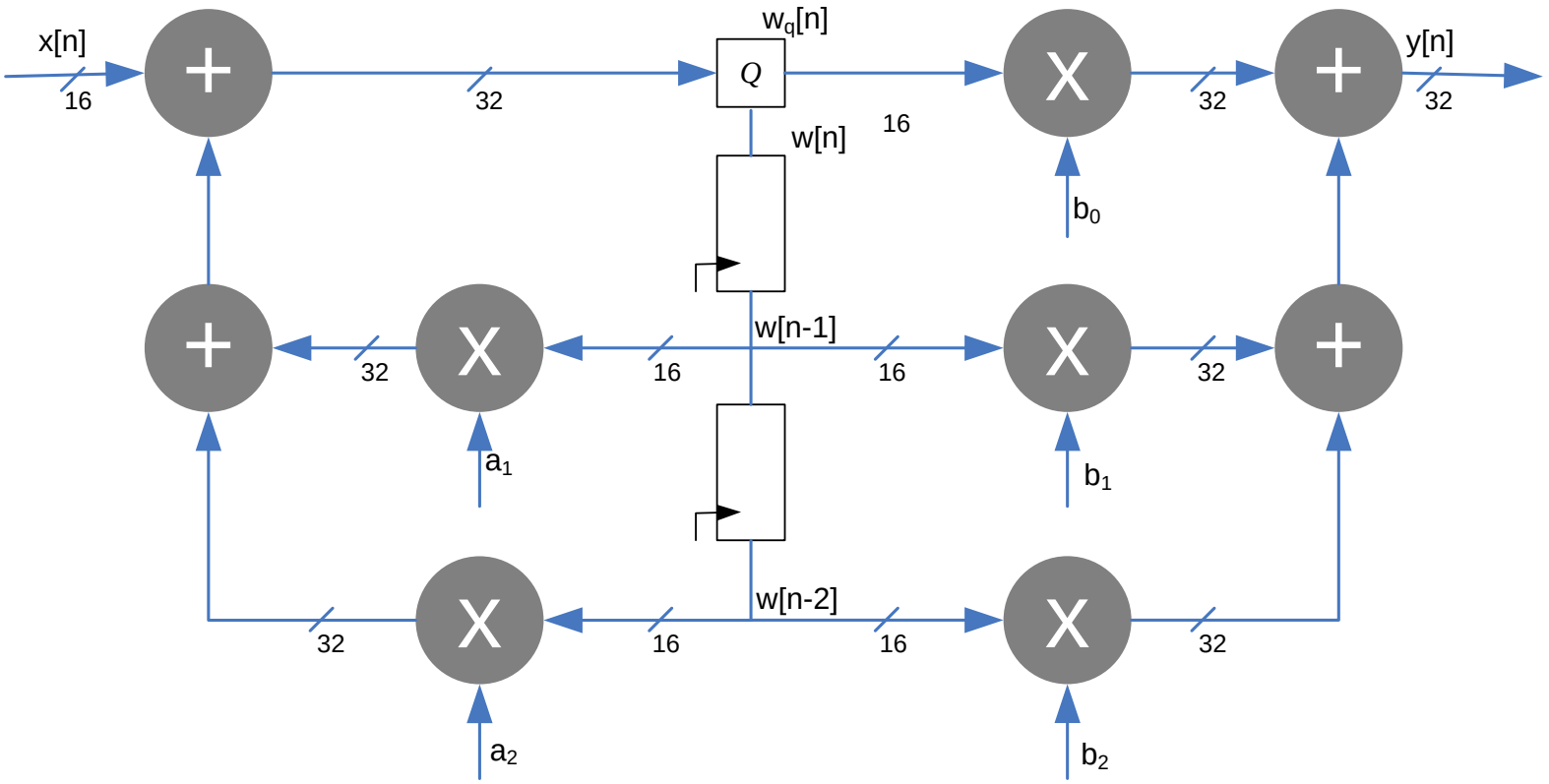
- Xilinx Family
  - DSP48 in Virtex4
    - One 18x18-bit 2's complement multiplier
    - Three 48 bit multiplexer
    - One three input 48-bit adder/subtractor
    - Number of registers
    - 40 modes of operation
- 18x18 multiplier in Spartan3, Spartan 3e and VirtexII and pro

Contd…


A
B
18
18
18 x18 bit multiplier
36


BCOUT
PCOUT
A
B
C
CIN
SUBSTRACT
P
BCIN
PCIN
Wire shift Right by 17 Bits
BCOUNT
PCOUT
A
B
CIN
SUBSTRACT
P
BCIN
PCIN
Wire shift Right by 17 Bits

Example: One to one mapping using embedded multipliers and DSP48 blocks

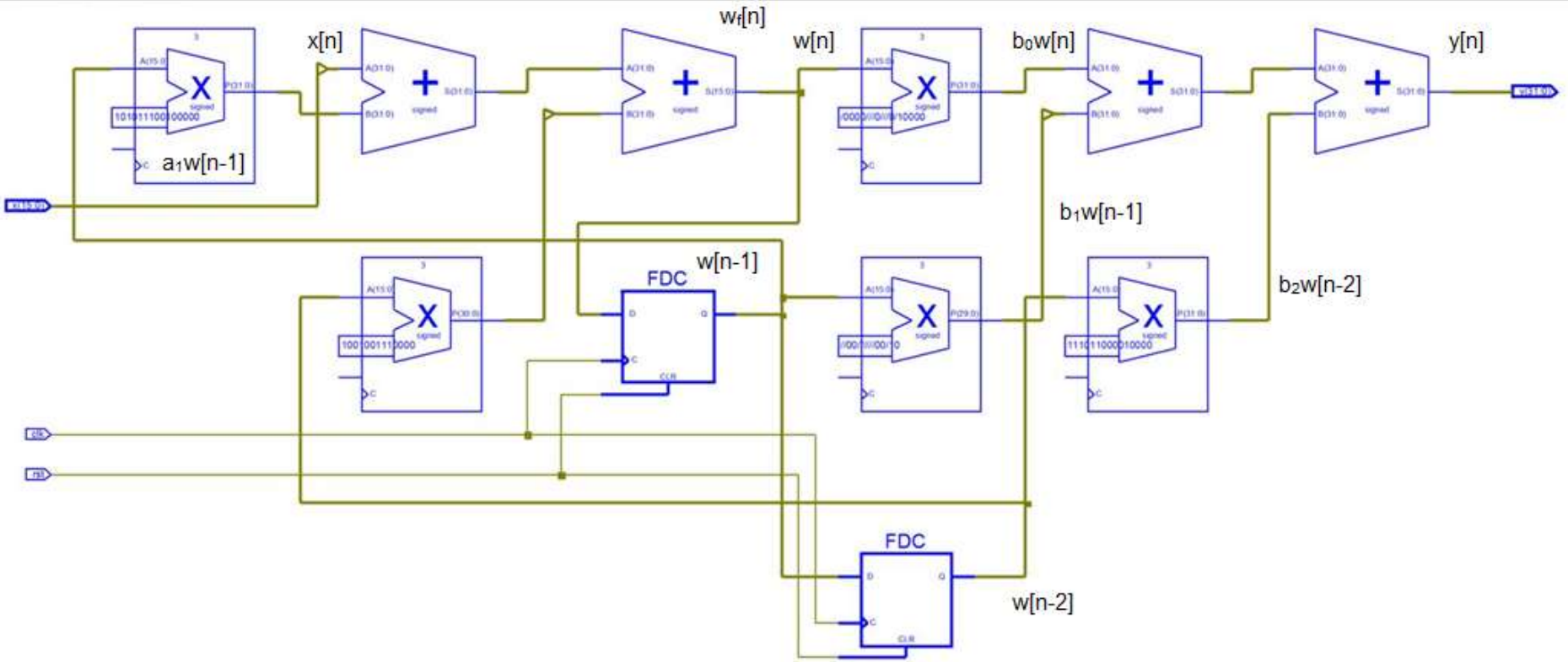


Implementation in RTL Verilog

```
module iir(xn, clk, rst, yn);

// x[n] is in Q1.15 format
input signed [15:0] xn;
input clk, rst;

// y[n] is in Q2.30 format
output signed [31:0] yn;

// Full precision w[n] in Q2.30 format
wire signed [31:0] wfn;

// Quantized w[n] in Q1.15 format
wire signed [15:0] wn;

// w[n-1]and w[n-2] in Q1.15 format
reg signed [15:0] wn_1, wn_2;

// all the coefficients are in Q1.15 format
wire signed [15:0] b0 = 16'ha7b0;
wire signed [15:0] b1 = 16'hf2b2;
wire signed [15:0] b2 = 16'h7610;
wire signed [15:0] a1 = 16'h5720;
wire signed [15:0] a2 = 16'h1270;
```

```
// w[n] in Q2.30 format with one redundant sign bit
assign wfn = wn_1*a1+wn_2*a2;

/* through away redundant sign bit and keeping
  16 MSB and adding x[n] to get w[n] in Q1.15 format */

assign wn = wfn[30:15]+xn;

// computing y[n] in Q2.30 format with one redundant sign bit
assign yn = b0*wn + b1*wn_1 + b2*wn_2;

always @(posedge clk or posedge rst)
begin
        if(rst)
        begin
                wn_1 <= 0;
                wn_2 <=0;
        end
        else
        begin
                wn_1 <= wn;
                wn_2 <= wn_1;
        end
End

endmodule
```

Synthesis on Spartan 3



Synthesis Results

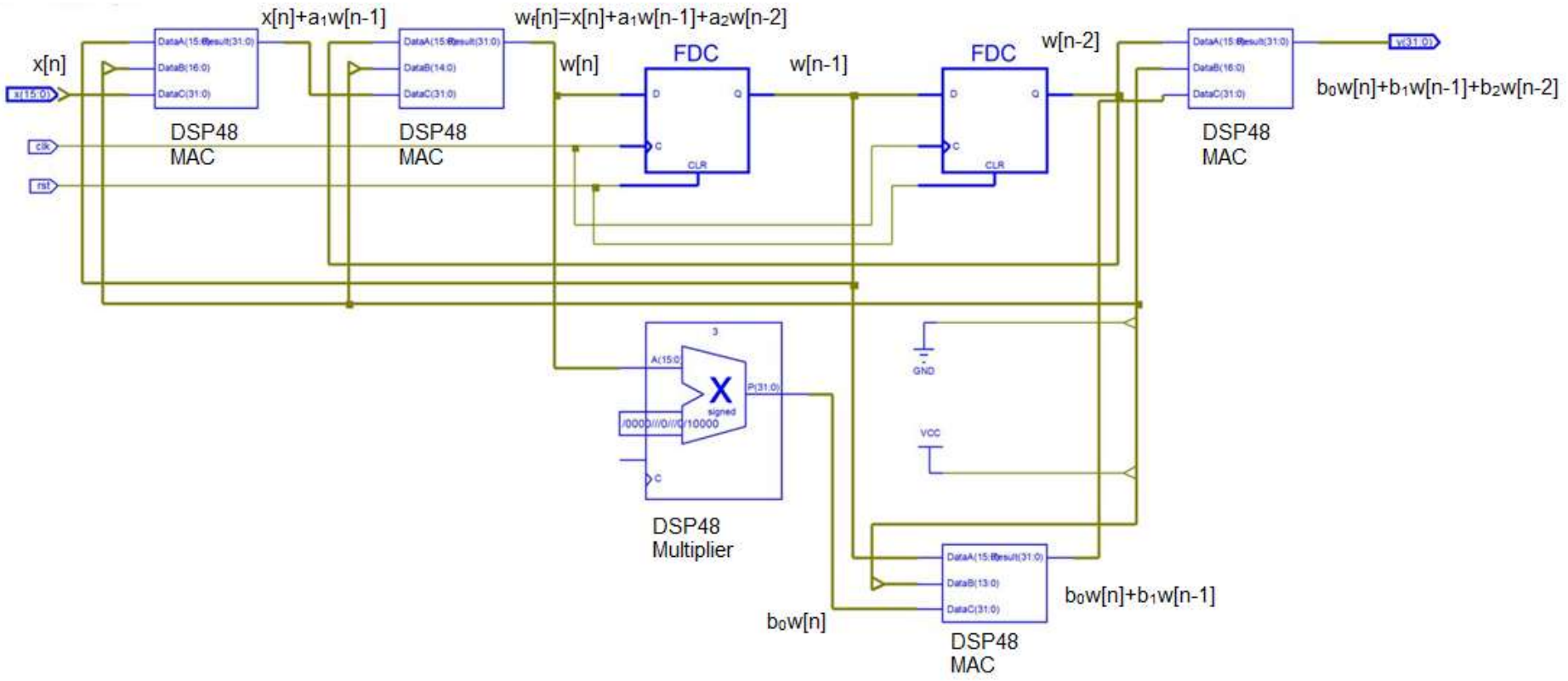
Selected Device : 3s400pq208-5

Minimum period: 10.917ns (Maximum Frequency: 91.597MHz)

Number of Slices:	58 out of 3584	1%
Number of Slice Flip Flops:	32 out of 7168	0%
Number of 4 input LUTs:	109 out of 7168	1%
Number of IOs:	50	
Number of bonded IOBs:	50 out of 141	35%
Number of MULT18X18s:	5 out of 16	31%
Number of GCLKs:	1 out of 8	12%

Synthesis on Virtex-4 Device

- Multiplication and addition operations in the DFG are now mapped on DSP48 blocks

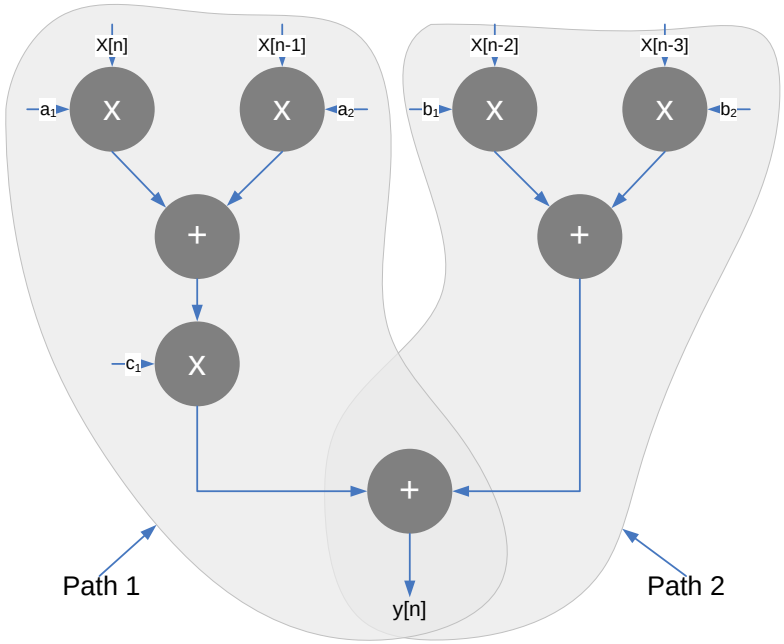


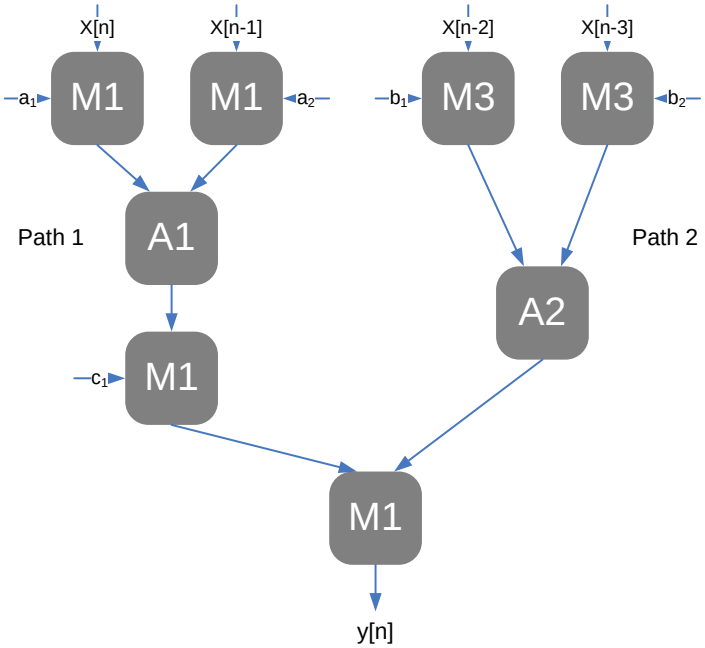
Design Options

- In case multiple design options for HW blocks available
  - Choose fastest blocks for critical path
  - Select area efficient blocks for other parallel paths that meets the timing

Basic Building Blocks	Relative Timing	Relative Area
Adder 1 (A1)	T	2.0A
Adder 2 (A2)	1.3T	1.7A
Adder 3 (A3)	1.8T	1.3A
Multiplier 1 (M1)	1.5T	2.5A
Multiplier 2 (M2)	2.0T	2.0A
Multiplier 3 (M3)	2.4T	1.7A

Design Options

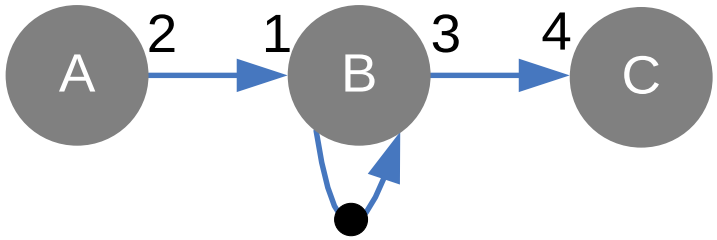




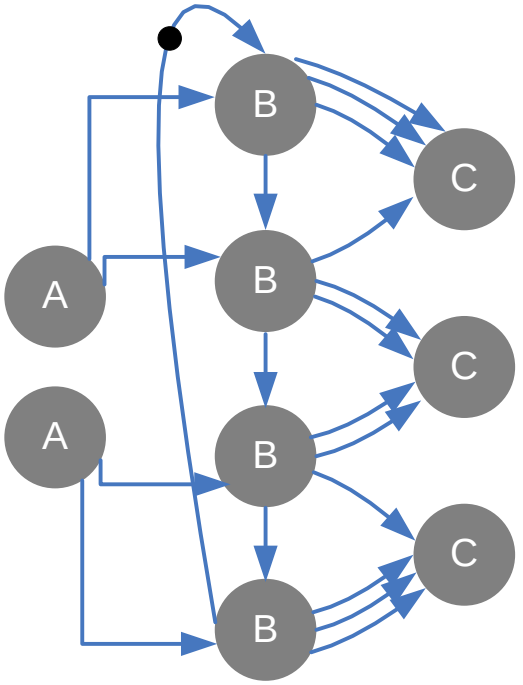
Homogeneous SDF (HSDF)

- Takes one sample at the input and generate one sample at the output

SDFG to HSDF conversion (a) SDFG consisting of three nodes A, B, and C (b) An equivalent HSDG

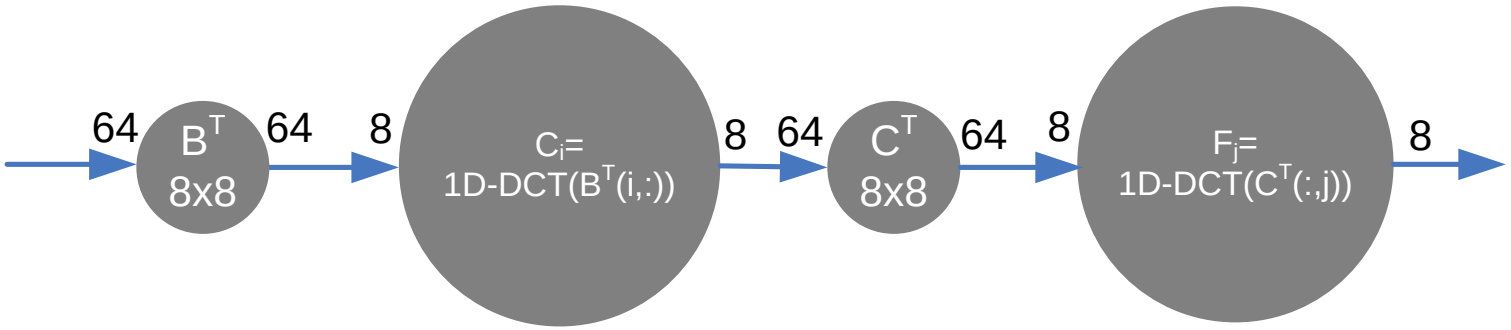


(a)

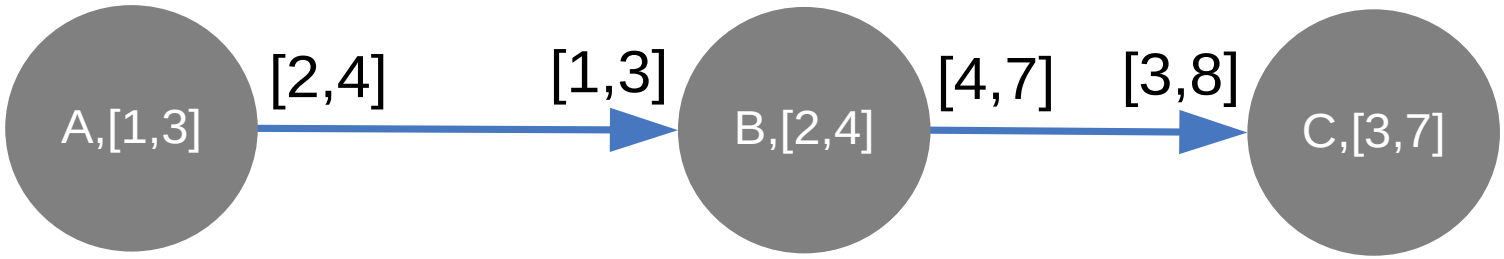


(b)

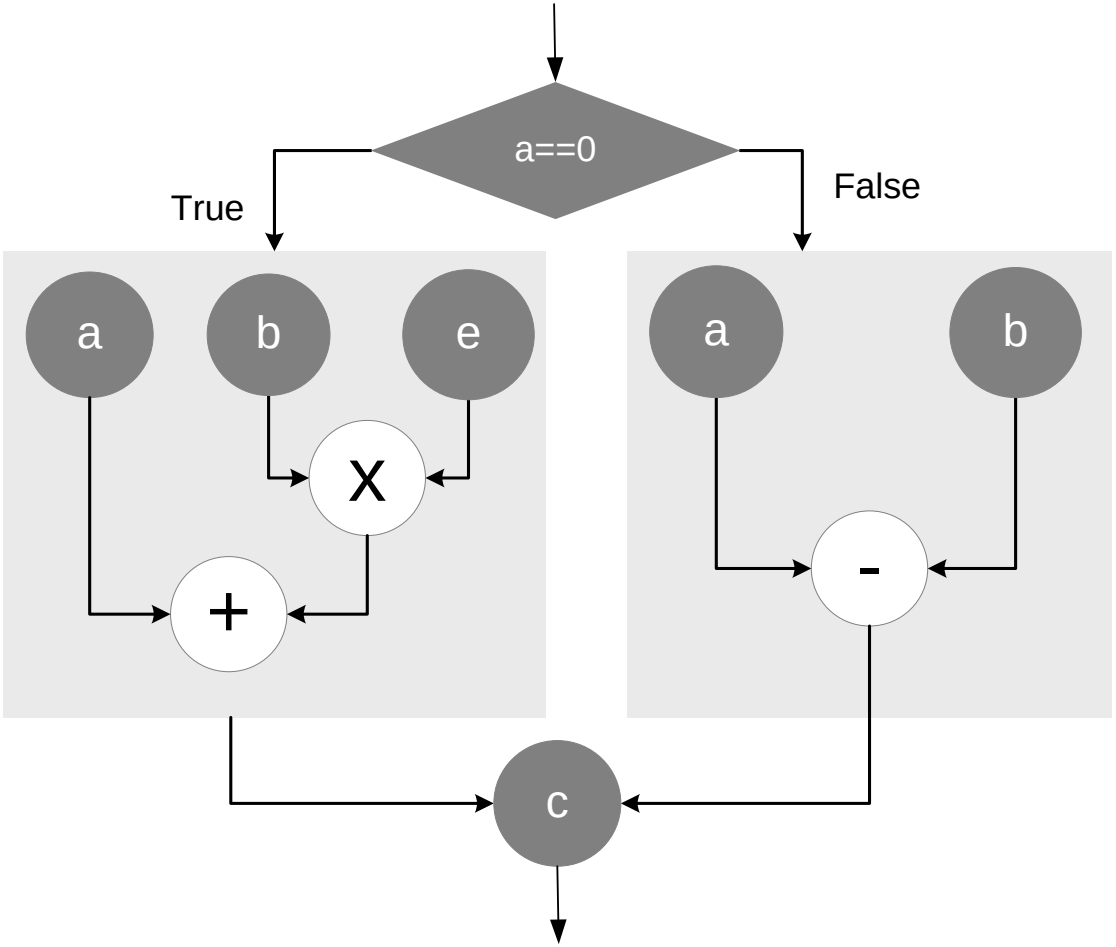
Computing two-dimensional DCT in a JPEG algorithm presents a good example to demonstrate a Multi-rate DFG



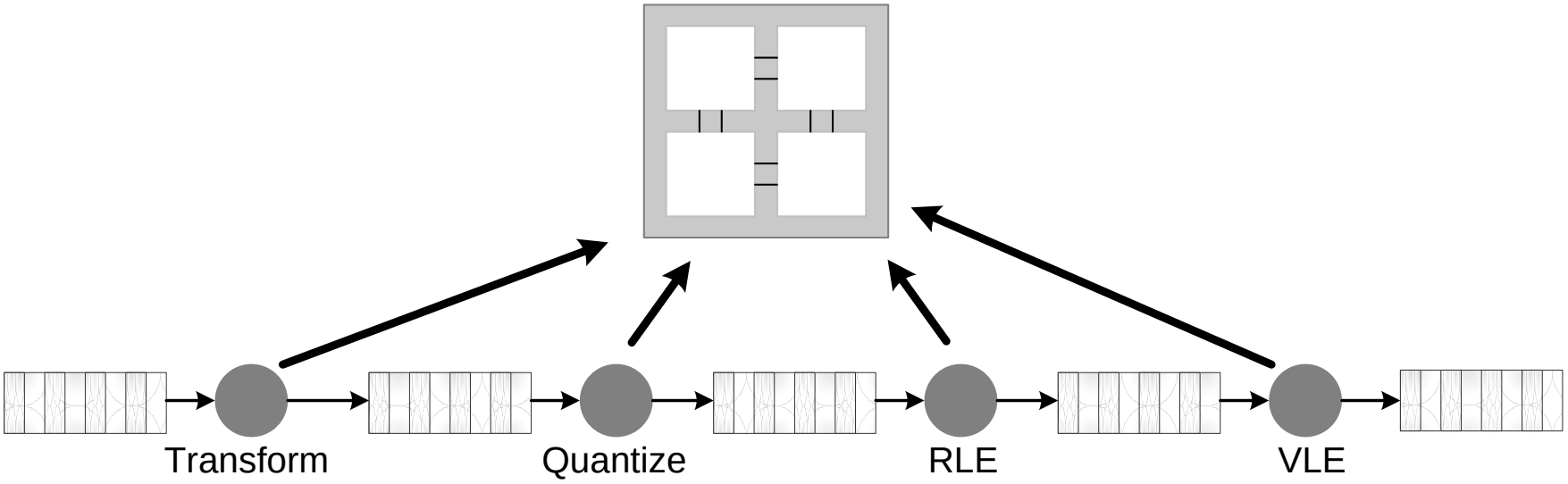
An example of a CSDFG, where node A first fires and takes 1 time unit and produces 2 tokens, and then in its second firing it takes 3 time units and generates 4 tokens, the node B in its first firing consumes one token in 2 time unit and in its second firing takes 4 time unit and consumes 3 tokens, it respectively produces 4, and 7 tokens, similarly behavior of node C can be inferred from the figure



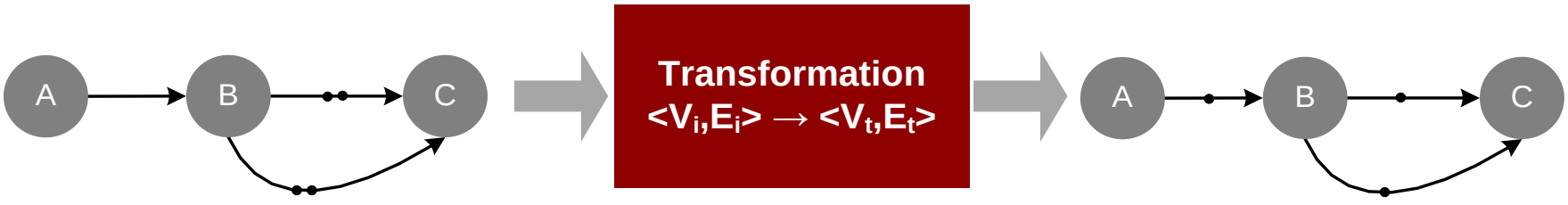
CDFG implementing conditional firing



A dataflow graph is a good representation for mapping on run-time reconfigurable platform



Mathematical transformation changing a DFG to meet design goals



Performance Measures

- Iteration Period
- Sample period and Throughput
- Latency
- Power Dissipation

Power Dissipation

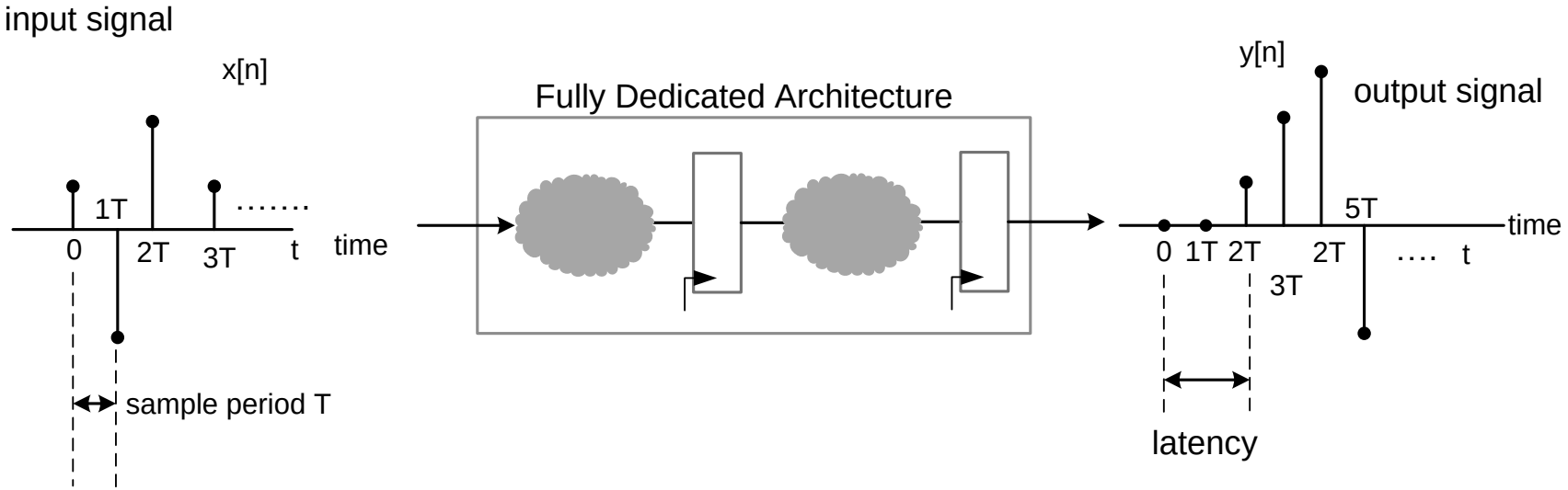
- Power is a major consideration
  - Specially for battery operated devices
- There are two classes of power dissipation in a digital circuit
  - Static power dissipation is due to the leakage current in digital logic
  - Dynamic power dissipation is due to all the switching activity
    - Major source of power dissipation
    - Stop the clock in areas that are not active
    - The dynamic power consumption in a CMOS circuit

$$P_d = \alpha C V_{DD}^2 f$$

α and C are switching activity and physical capacitance of the design
V_{DD} is the supply voltage
f is the clock frequency.

Latency and sampling period

- Latency is the time delay for the design to produce an output y[n] in response to an input x[n]
- Sampling period as the time difference between the arrivals of two successive samples

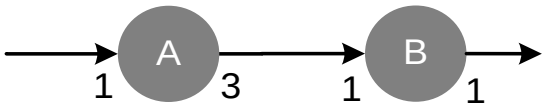


Mapping Multi-rate DFG in Hardware

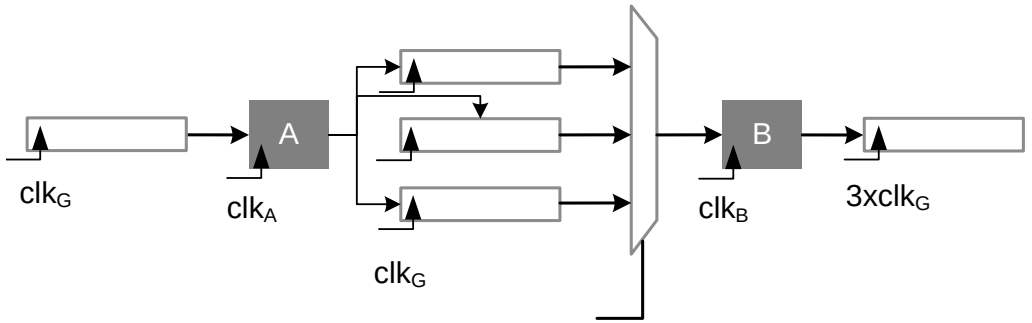
- Sequential Mapping
- Parallel Mapping

(Detail description of this part is covered in the book)

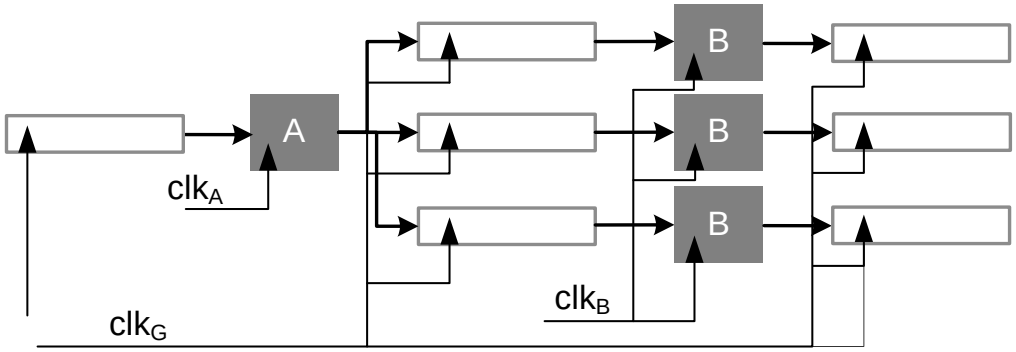
(a) Multi rate to single rate edge (b) Sequentially synthesis (c) Parallel Synthesis



(a)

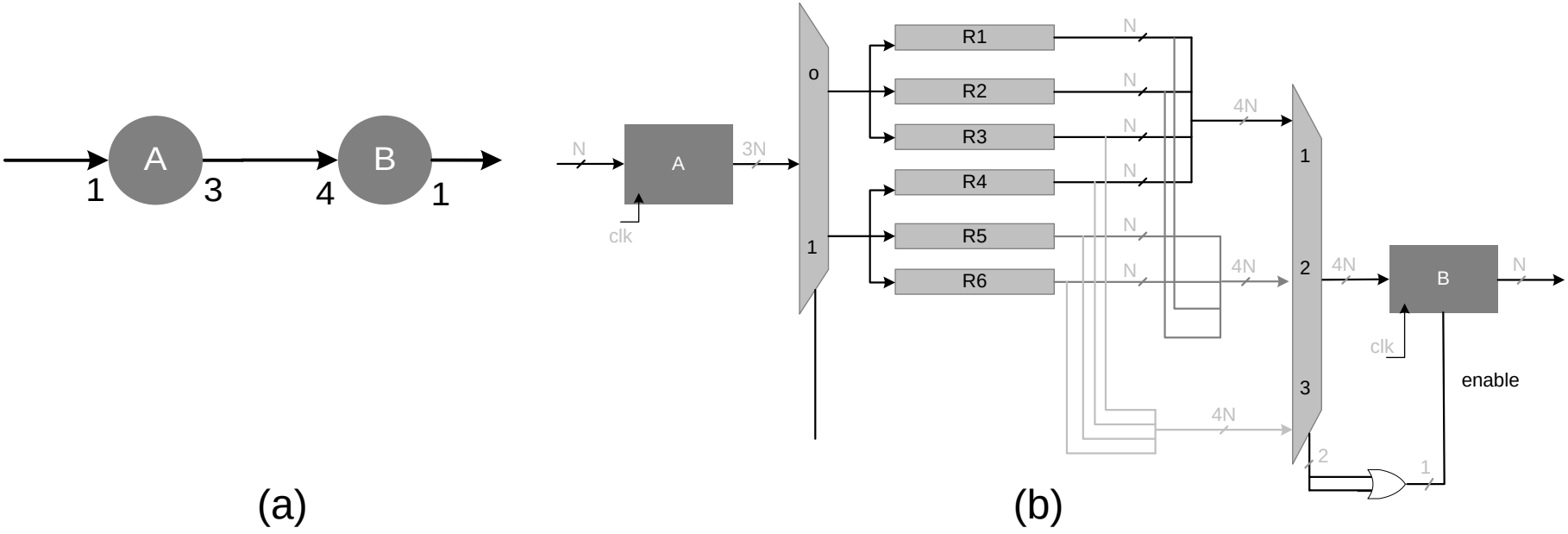


(b)



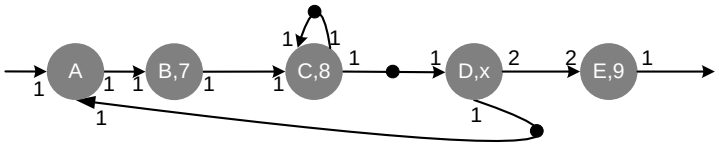
(c)

(a) A multi-rate DFG (b) Hardware realization

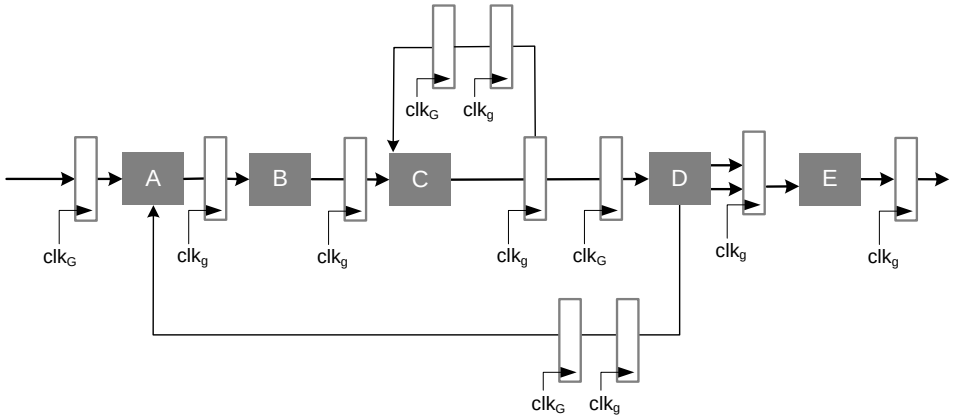


(a) (b)

(a) Hypothetical DFG with different types of node (b) HW realization



(a)



(b)

Summary

- The chapter defines synchronous digital design to process digital signals
  - Define 1-D, 2-D and video digital signals
- Introduces KPN to design top level architecture for streaming applications
  - Major limitation of KPN and its resolution is presented
- Presents Graphical representation of DSP algorithm as
  - Block diagram, signal flow graphs and DFGs
- Mapping and scheduling of multi rate DFG is done based on repetition vector or self timing firing
- For fully dedicated architecture one to one mapping for DFG to HW is performed for HSDFGs
- HW synthesis for multi rate DFG can be done as sequential or parallel designs