Integrated Circuits Design by FPGA

م.م. أحمد مؤيد عبدالحسين
جامعة الفرات الأوسط التقنية / الكلية التقنية الهندسية / نجف

Lecture 15

System Design

Digital Filters

Objectives of this Lecture

- To define **Digital Filters**
- To implement **Digital Filter (FIR)** in an example design.

Contents of this Lecture

- Introduction
- **FIR** filter Example

Introduction

- Digital signal processing (DSP) finds innumerable applications in the fields of audio, video, and communications, among others. Such applications are generally based on LTI (linear time invariant) systems, which can be implemented with digital circuitry.

Any LTI system be represented by the following equation:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

- Where a_k and b_k are the filter coefficients, and x[n - k], y[n - k] are the current (for k = 0) and earlier (for k > 0) input and output values, respectively.
- To implement this expression, registers are necessary to store x[n- k] and/or y[n - k] (for k > 0), besides multipliers and adders, which are well-known building blocks in the digital domain.

Introduction

- The impulse response of a digital filter can be divided into two categories: **IIR** (infinite impulse response) and **FIR** (finite impulse response).
- The **IIR** corresponds to the general case described by the equation below, while the **FIR** occurs when N = 0.

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

- Only **FIR** filters can exhibit linear phase, so they are indispensable when linear phase is required, like in many telecom applications. With N = 0, the equation above becomes:

$$y[n] = \sum_{k=0}^{M} c_k x[n-k]$$

Introduction

- Where $c_k = b_k/a_0$ are the coefficients of the **FIR** filter. This equation can be implemented by the system of figure 12.8, where D (delay) represents a register (flip-flops), a triangle is a multiplier, and a circle means an adder.

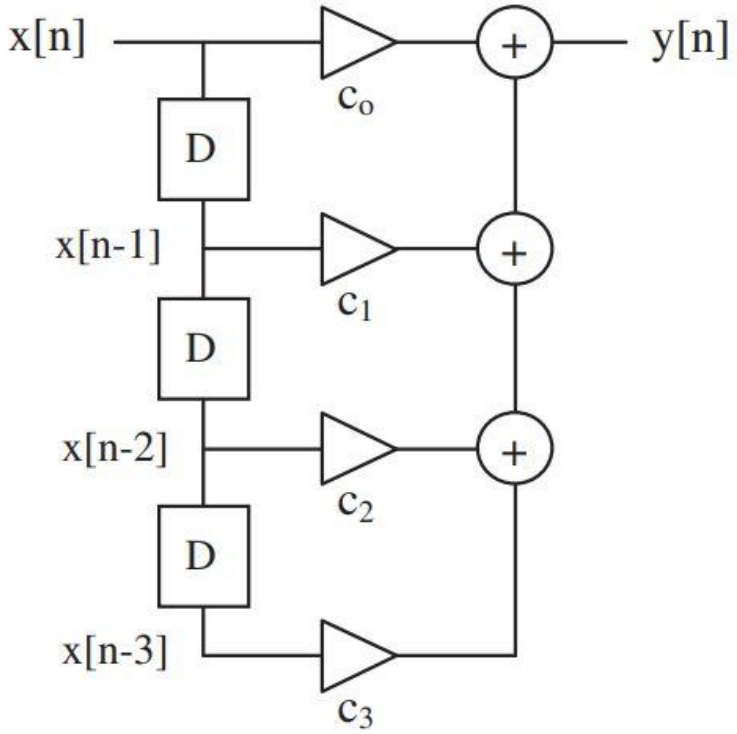


Figure 12.8
FIR filter diagram (with 4 coefficients).

Introduction

- An equivalent RTL representation is shown in figure 12.9. As shown, the values of x are stored in a shift register, whose outputs are connected to multipliers and then to adders.

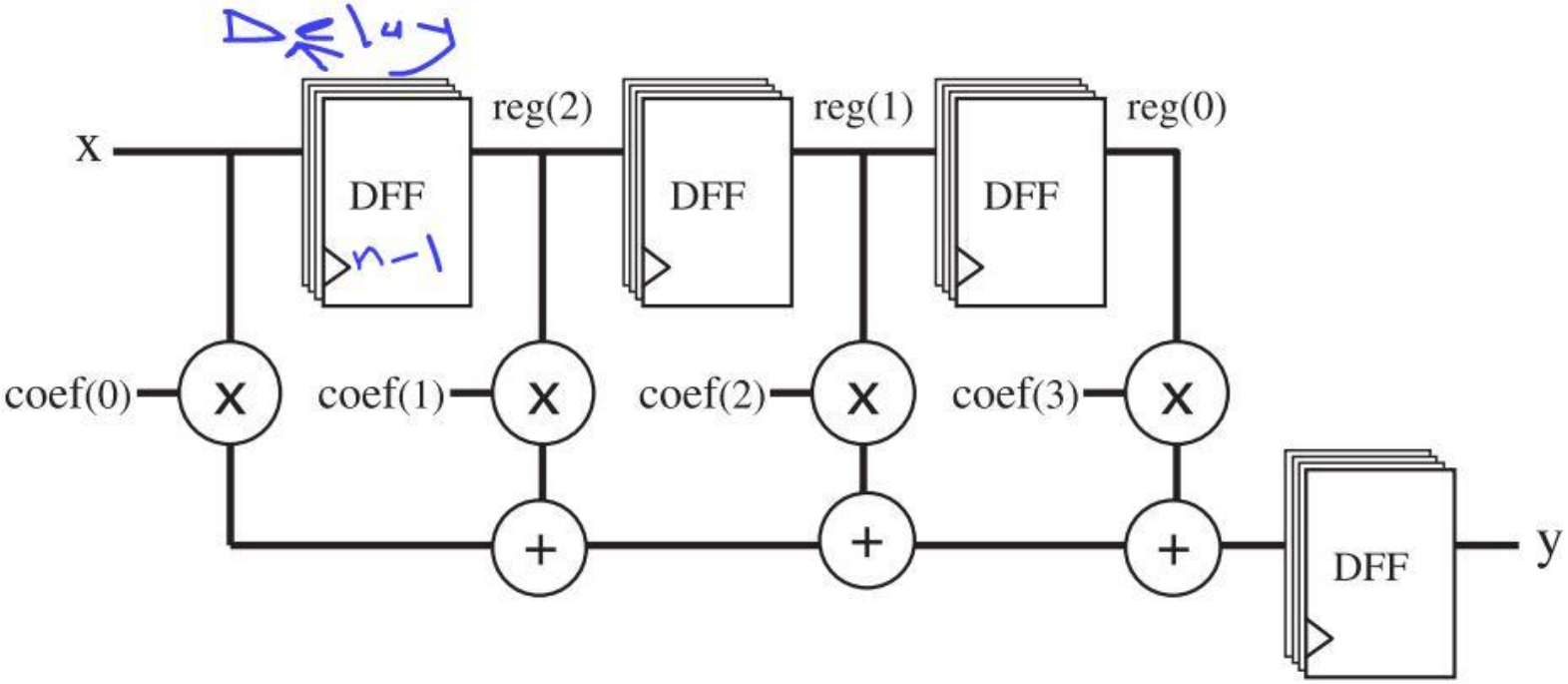


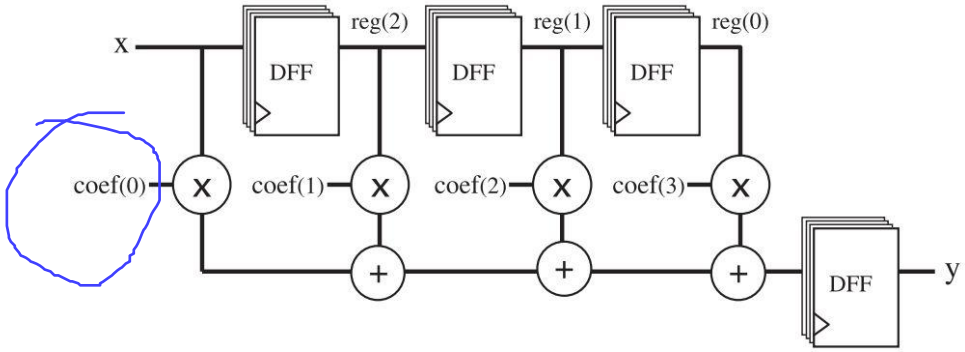
Figure 12.9
RTL representation of a FIR filter.

Introduction

- The coefficients must also be stored on chip. However, if the coefficients are always the same (that is, if it is a dedicated filter), their values can be implemented by means of logic gates rather than registers (we just need to store CONSTANTS).
- On the other hand, if it is a general purpose filter, then registers are required for the coefficients. In the architecture of figure 12.9, the output vector (y) was also stored, in order to provide a clean, synchronous output.
- Notice that the lower section of the filter contains a MAC (multiply-accumulate) pipeline. This circuit is closely related to the MAC circuit discussed in section 12.3. Here too, overflow can happen, so an add/truncate procedure must be included in the design.
- With n = m = 4, the synthesized circuit required 20 flip-flops (four for each stage of the shift register, plus eight for the output)

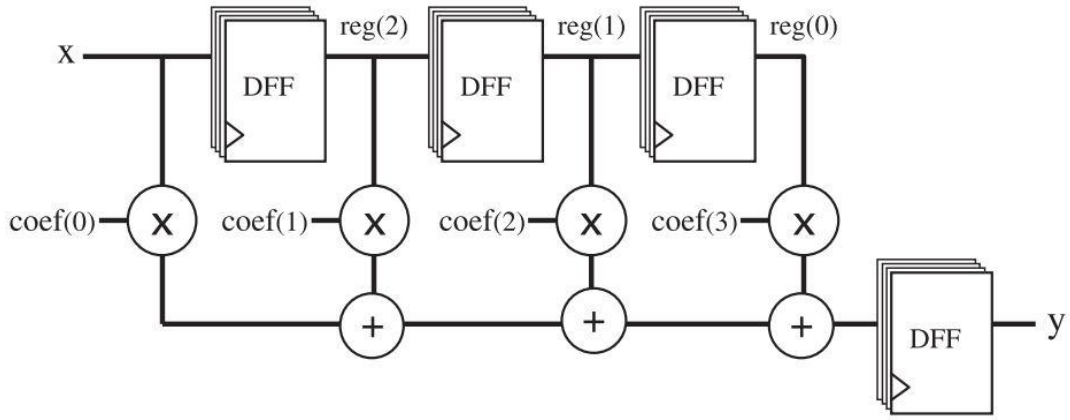
FIR filter example

```
1  ------------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_arith.all;  -- package needed for SIGNED
5  ------------------------------------------------------
6  ENTITY fir2 IS
7     GENERIC (n: INTEGER := 4; m: INTEGER := 4);
8     -- n = # of coef., m = # of bits of input and coef.
9     -- Besides n and m, CONSTANT (line 19) also need adjust
10    PORT ( x: IN SIGNED(m-1 DOWNTO 0);
11           clk, rst: IN STD_LOGIC;
12           y: OUT SIGNED(2*m-1 DOWNTO 0));
13 END fir2;
14 ------------------------------------------------------
15 ARCHITECTURE rtl OF fir2 IS
16    TYPE registers IS ARRAY (n-2 DOWNTO 0) OF
17                             SIGNED(m-1 DOWNTO 0);
18    TYPE coefficients IS ARRAY (n-1 DOWNTO 0) OF
19                               SIGNED(m-1 DOWNTO 0);
20    SIGNAL reg: registers := ("0000", "0000", "0000");
21    CONSTANT coef: coefficients := ("0001", "0010", "0011",
22                                    "0100");
```



FIR filter example

```
23 BEGIN
24    PROCESS (clk, rst)
25       VARIABLE acc, prod:
26          SIGNED(2*m-1 DOWNTO 0) := (OTHERS=>'0');
27       VARIABLE sign: STD_LOGIC;
28    BEGIN
29       ----- reset: -------------------------
30       IF (rst='1') THEN
31          FOR i IN n-2 DOWNTO 0 LOOP
32             FOR j IN m-1 DOWNTO 0 LOOP
33                reg(i)(j) <= '0';
34             END LOOP;
35          END LOOP;
36       ----- register inference + MAC: -------
37       ELSIF (clk'EVENT AND clk='1') THEN
38          acc := coef(0)*x;
39          FOR i IN 1 TO n-1 LOOP
40             sign := acc(2*m-1);
41             prod := coef(i)*reg(n-1-i);
42             acc := acc + prod;
```



FIR filter example

```
43              ---- overflow check: ------------
44              IF (sign=prod(prod'left)) AND
45                    (acc(acc'left) /= sign)
46                 THEN
47                 acc := (acc'LEFT => sign, OTHERS => NOT sign);
48              END IF;
49           END LOOP;
50           reg <= x & reg(n-2 DOWNTO 1);
51        END IF;
52        y <= acc;
53     END PROCESS;
54 END rtl;
55 ---------------------------------------------------------
```

Shift Right Process

FIR filter example

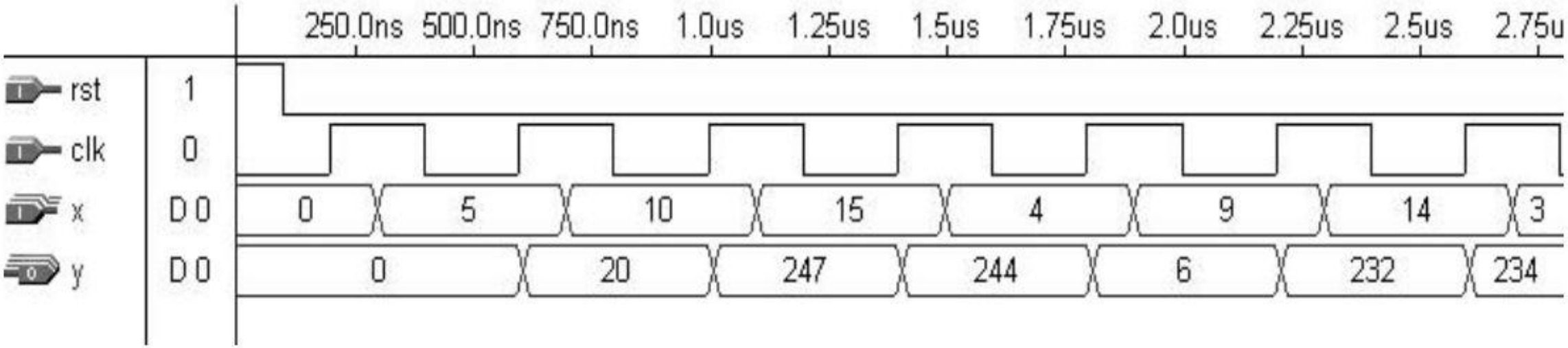
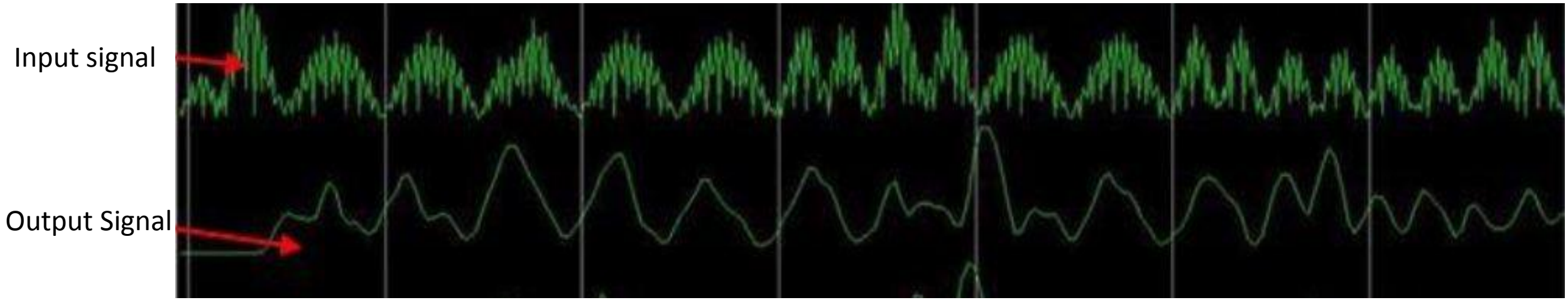


Figure 12.10
Simulation results of FIR filter of figure 12.9.

FIR filter example



Assignments

The assignments will be attached to your class room

End of lecture 15
Any Questions ?