LINKÖPING UNIVERSITY
Department of Electrical Engineering

TSIU03, SYSTEM DESIGN

LECTURE 10

Kent Palmkvist
Kent.Palmkvist@liu.se

Slides by: Mario Garrido Gálvez (mario.garrido.galvez@liu.se)

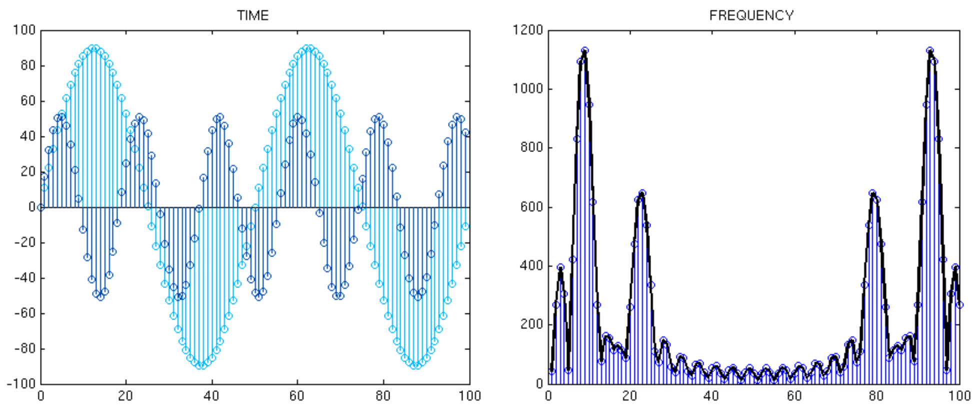
Linköping, 2021

TODAY

- Time and frequency domains.
- Parameterizing and replicating hardware in VHDL.
- Resources for the project.

TIME vs FREQUENCY

- In the frequency domain (to the right) we see the spectrum of both signals together happening together. This may be two musical notes played at the same time.
- How can you see the frequency and the amplitude in the frequency domain?



DISCRETE FOURIER TRANSFORM (DFT)

- Transforms a signal in the time domain into the frequency domain.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}$$

- x[n] are the input data, k = 0…N-1 are the output frequencies, X[k] are the values at each frequency.
- How many complex sums an multiplications does the DFT need? -> O(N^2)

FAST FOURIER TRANSFORM (FFT)

- Example: 4-point FFT.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}$$

$$X[0] = x[0] + x[1] + x[2] + x[3]$$

$$X[1] = x[0] + x[1]e^{-j\frac{\pi}{2}} - x[2] + x[3]e^{+j\frac{\pi}{2}}$$

$$X[2] = x[0] - x[1] + x[2] - x[3]$$

$$X[3] = x[0] + x[1]e^{+j\frac{\pi}{2}} - x[2] + x[3]e^{-j\frac{\pi}{2}}$$

- How many operations do we need? How can we reduce operations?

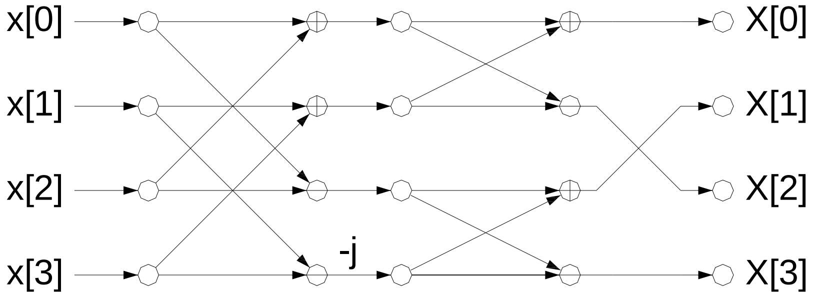
FAST FOURIER TRANSFORM (FFT)

$$X[0] = (x[0] + x[2]) + (x[1] + x[3])$$

$$X[1] = (x[0] - x[2]) + (-j)(x[1] - x[3])$$

$$X[2] = (x[0] + x[2]) - (x[1] + x[3])$$

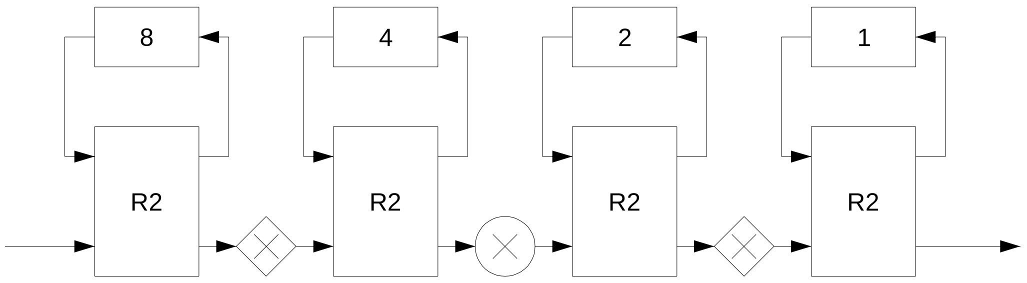
$$X[3] = (x[0] - x[2]) + (-j)(x[1] - x[3])$$



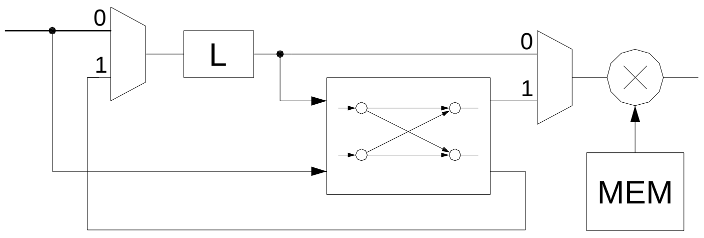
- How many operations do we need now? -> O (Nlog$_2$N)

FFT ARCHITECTURE

- This is an example of an FFT hardware architecture:



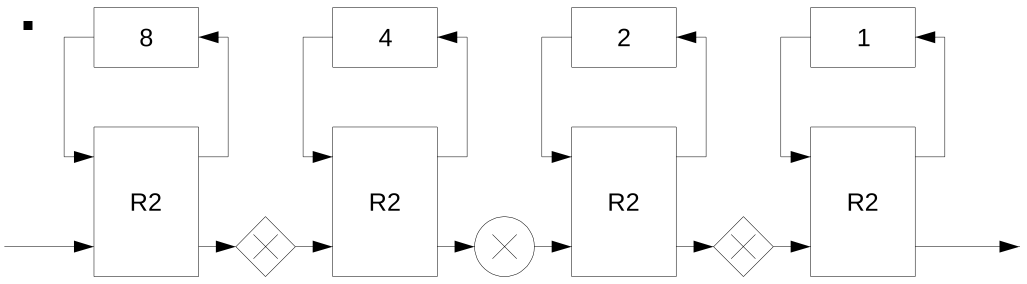
- This is how a stage looks like:



7

PARAMETERIZATION

- We want to implement an FFT architecture whose length, word length and type of multiplexer can be configured.
-



- It can have any number of stages, n, and the stages differ in:
  - Length of the buffers: In each stage the length is a different power of 2.
  - Word length: It increases in every stage.
  - Type of multiplier: Complex rotator vs trivial rotator.
- How can we implement it efficiently in VHDL?

8

GENERIC

1) Use **generic** for the parameters that may change. They are used internally as constants in the architecture.

```
entity FFTstage is
  generic(n:  integer:= 6;    -- Total number of stages.
      s:       integer:= 5;    -- Index of the current stage.
      WL:      integer:= 8;    -- Input word length.
      DeltaWL: integer:= 0);   -- Increment of WL in the stage.
  port(reset: in std_logic;
      clk:     in std_logic;
      counter: in std_logic_vector(n-1 downto 0);
      datain : in std_logic_vector(2*WL -1 downto 0);
      dataout: in std_logic_vector(2*(WL+DeltaWL) -1 downto 0));

end FFTstage;
```

- We only need to write one .vhd file for all the stages!!

GENERIC MAP

2) Use **generic map** to assign the values of the generic when instantiating a component:

```
stage3: entity work.FFTstage
  generic map(n   => 8,  -- Total number of stages.
          s       => 3,  -- Index of the current stage.
          WL      => 8,  -- Input word length.
          DeltaWL => 0)  -- Increment of WL in the stage.
  port map(reset  => reset,
          clk     => clk,
          counter => counter,
          datain  => datain_stage3,
          dataout => datain_stage4);
```

- But in this case, we have to instantiate the stages one by one. Can we do it better?

FOR…GENERATE

3) Use **for…generate** to generate multiple instances of the same component (such as many multipliers for a filter). **for…generate** also admits to configure the generics of each instance specifically:

```
FFTStages: for i in 1 to n generate
begin
  stage: entity work.FFTstage
    generic map(n      => n,      -- Total number of stages.
            s       => i,      -- Index of the current stage.
            WL      => WL,     -- Input word length.
            DeltaWL => DeltaWL) -- Increment of WL.
    port map(reset  => reset,
            clk     => clk,
            counter => counter,
            datain  => datain(i),
            dataout => datain(i+1));
end generate;
```

IF…GENERATE

4) For conditional instantiation of code, we can use if…generate. For instance, the last stage of the FFT does not require a rotator. Then:

```
FFTStages: if s < n generate
begin
  mult: entity work.mult
    generic map(WL   => WL)
    port map(datain  => datainMult(s),
             dataout => dataoutMult(s+1));
 end generate;

FFTStages: if s = n generate
begin
   dataout <= datainMult(s);
end generate;
```

INTEGER AND REAL IN VHDL

- Integer and real numbers are very useful to do calculations with the parameters.
- Note that these calculations are done before the circuit is implemented, so these operations can be non-synthesizable, and we can use for them any mathematical operation.
- Integer numbers are included by default in VHDL. For real numbers we need to use the package **math_real**. It contains operators (non-synthesizable) such as: cos, sin, log, log2, ceil, floor, **,
- Example: how can we describe the entity of a memory where the number of addresses and the data word length are parameters (generic values)?.

INTEGER AND REAL IN VHDL

- Example: Entity of a memory where the number of addresses and the data word length are parameters (generic):

```
library ieee;
use ieee.math_real.all; -- Needs to be included to use real types.
entity mem is
 generic (nAddr : integer := 256
    WL: integer := 16)
 port (rst: in std_logic;
    clk: in std_logic;
    we : in std_logic;
    datain : in std_logic_vector (WL-1 downto 0);
    Addr: in std_logic_vector (integer(ceil(log2(real(nAddr))))-1 downto 0);
    dataout: out std_logic_vector (WL -1 downto 0));
end mem;
```

RESOURCES FOR THE PROJECT (1)

- Important documents on the web page:
  - Project Specification.
  - Guidelines to Write the Project Documents.
  - Templates for the Project Documents.
  - How to describe a HW circuit.

RESOURCES FOR THE PROJECT (2)

- Other useful documents:
  - Datasheet of the DE2-115 Board.
  - How to load a background image to the SRAM.
  - Petter´s small VHDL guide.
  - VHDL Packages [Rushton Appendix A].
  - VHDL Syntax Reference [Rushton Appendix B].

GOOD LUCK!!

CHECKLIST FOR LECTURE 10

- Time and frequency, DFT, FFT.
- Parameterization: generic, generic map, if…generate, for…generate, real, integer

AT HOME

- Review the checklist for lecture 10 and check that you understand all the concepts and you know how to use them.
- Go through the checklists of all the lectures to check that you understand all the concepts in the course.