Wiki »

EXERCISE: OPTIMIZING ARITHMETIC

Exercise Goals

This exercise will through simulation of different multiplication methods introduce you to:

- Get experience with solving problems caused by limited max frequency, area a.o.
- Understand how terms as pipelining, clock frequency and latency works in real life.

Prerequisites

- Quartus II and ModelSim software must be installed and working.
- Template code checked out from repository (exercise_multiplier)

Introduction

In this exercise we will investigate how our selection of multiplier implementation affects our design in terms of:

- Maximum Clock Frequency
- Latency
- Resource Usage

Compiling our design in Quartus* will generate a .vho file, a VHDL file that describes the actual gate level design. The gate level design includes the actual logic elements, memory blocks and other components to which to design is compiled. You can use this file for simulation in ModelSim, thus getting to simulate the actual implementation.

-) Quartus must be configured to do so in assignments->Settings->EDA Tool Settings->Simulation. It has been prepared in the template project.

Exercise Steps

The exercises do not require a lot of programming, but they require a lot of understanding! In the first exercise you will only have to change a few parameters and comment/uncomment a few lines of code. It is important however, that you write down what you observe.

1. Multiplying with the "*" operator

A) Open the existing project

- Open MultiplierTest.vhd in Quartus.
- Check that following lines are uncommented/commented:
  outAB <= std_logic_vector(unsigned(inA_reg) * unsigned(inB_reg));
  -- outAB <= MultAB;
- Check the constant value: MULT_PIPELINE_DEPTH := -1

The Multi_proc procedure puts a register on inputs inA and inB and multiplies the two registered inputs together. Multiplication is done inside the process and an output register will therefore be added.

B) Investigate the result:

- Compile the design in Quartus
- Open the output in the “RTL Viewer” and the “Technology Map Viewer”

How does the implementation look? Is the design as we expected? Is it pipelined?

C) Look at the compilation report:

- Open the “Compilations Report” Tab in Quartus

How many Logical Elements, Registers and embedded multipliers have we used?

D) How is timing performing?

- Open the “Time Quest Analyzer” folder in the Compilation ReportIs the timing analyzer successful? What is the design’s Fmax (Slow Model-Fmax Summary)?

E) Simulate the multiplier.

- Open ModelSim
- Create a new project in design folder
- Add the two files MultiplierTest.*vho *, MultiplierTest_tb.vhd
- Compile, Simulate, add signals to waveform
- Run simulation for 300ns

How much latency do we have? Remember that the input- and output registers already gives us a latency of two clock cycles.

Conclusion? Are we satisfied with this multiplying solution? How can we improve performance?

2. Multiplying with LPM functions

A) Next we’ll try the same as in the previous step, but using LPM components instead.

- Re-comment the previously uncommented line in MultiplierTest.vhd
- Uncomment/comment the lines:
  outAB <= MultAB;
  -- outAB <= std_logic_vector(unsigned(inA_reg) * unsigned(inB_reg));
- Set the pipeline depth to ‘1’: MULT_PIPELINE_DEPTH := 1;

The LPM_MULT component input is mapped to inA_reg and inB_reg and MultAB. When uncommenting the line above, we use the LPM_MULT component instead.

B) Run through all the steps from the previous section. Note what effects the introduced pipelining has on Fmax, latency, area a.o. – Use the same ModelSim project as before!

C) Add more pipelining stages by setting MULT_PIPELINE_DEPTH to a higher value. There is an upper limit, where additional pipeline stages do not have an effect on Fmax, since other delays become the limiting factor.

Conclusion? Is this better? -if so how?

Note! The LPM components can be created using the MegaWizard in Quartus. The MegaWizard creates a component with a name.vhd chosen by you. The content is a simple instantiation of an lpm component, but with generics defined according to your settings in the MegaWizard. In MultiplierTest.vhd the lpm_mult component is instantiated directly. The lpm library is added at the top of the module.