VU Advanced Digital Design
Lab Exercise 2

Robert Najvirt & Andreas Steininger & Florian Huemer

December 11, 2017

Part A

A(1)

Synthesize the design "nosync.vhd" and download it to a DE2-115 FPGA board provided in the ECSLAB. Do not forget to add timing assumptions (.sdc) to your design.

The design contains a data source and a data sink that can be independently clocked. The source produces a new data word (12 bit) every clock cycle; whose uppermost 8 bit field alternates between the values "0x55" and "0xAA" and whose 4 least significant bits implement a counter that increments by one with each step. The receiver's task is to capture this data word using its own clock. The received data word is available at port "data_dst". A two-phase encoded request line is used to indicate valid new data.

At the receiver's clock port "clk_dst" connect a clock with frequency of 60MHz (fixed), at the sender's clock port "clk_src" connect a clock with 60 MHz as well, but the latter clock shall be modulated by ±1 MHz with a modulation frequency of 100 Hz using a triangular modulation waveform. The required pulse generators are available in the lab; for the modulated clock use the Arbitrary Waveform Generator from Agilent; for the fixed clock you may also use the clock source on the FPGA board.

Be careful with the clock generator impedance settings: check whether the FPGA has a 50 ohm input (usually not) and set the generator impedance / output voltage accordingly. If you are not sure what you are doing, ask a supervisor or tutor **before** starting the signal generator.

Use the logic analyzer to observe the behavior of the circuit. You should notice two problem sources: missing flow control (skipped and doubly received datawords) and missing synchronization (wrong/inconsistent data). Introduce flow control to the design, removing the former problem. Now, find and display an event where an erroneous data word is received (screenshot ⇒ slides). Furthermore, observe the rate of erroneous data words at the receiver and compute the MTBU (⇒ measurement concept & result ⇒ slides) as well as the error probability (⇒ slides) for a transmission.

A(2)

Change the design to improve the MTBU to a value of at least 1000000 years, using a handshake protocol. For the presentation, prepare a block diagram and sketch the key concept behind it. Can you sustain the data rate? If not determine the new maximum data rate (both throughput and min./max./avg. latency ⇒ slides).

Part B

Build an even/odd synchronizer as outlined in [1]. Consider the following general hints:

- Note that in your setup you do not need prediction for the clock frequencies, which most of the paper is devoted to.
- In your assignments no flow control is required, so data words may be lost or duplicated. This is to simplify your task. Normally one would have to implement an appropriate flow control such that the sender and/or receiver (depending on who is faster) will have cycles where no data can be sent/received.
- When designing synchronizers that require a bounded phase drift, consider that the sender clock will run with 59 - 61 MHz and the receiver with 60 MHz. Set your synchronizer very conservatively to account for jitter and uncertainties resulting from FPGA prototyping.
- Delay elements can be realized by a chain of inverters. Use the "keep" attribute for the internal signals as used for the toggle bits in the source to prevent inverters from being optimized away.

In your presentation (slides!) explain the synchronizer principle, the key steps you had to take, mention pitfalls you may have experienced [once removed, we will not blame you for them :)]. Give a few representative screenshots from the logic analyzer demonstrating how your solution works. Also report the throughput and min./max./avg. latency that you were able to achieve. If applicable, report on the limits: margins you have chosen in the design, and that you have verified by measurements (you may change the generator settings for this, of course), assumptions you had to make.

References

[1] W. J. Dally and S. G. Tell. The even/odd synchronizer: A fast, all-digital, periodic synchronizer. In *2010 IEEE Symposium on Asynchronous Circuits and Systems*, pages 75–84, May 2010.