Laboratory Exercise 6

SERDES (part I)
Oversampling CDR

The purpose of this laboratory exercise is to design a digital SERDES. In the first part of the lab, you will design all the required subblocks for the digital SERDES and simulate them. The design will be verified by a separate functional test for each block. Functional verification verifies basic functionality, i.e. counting for counter; shifting for shifter. In part II of the lab you will integrate the blocks for transmitter and receive and run simulations to verify the functionality of the designed SERDES.

Introduction[1]

SERDES are used for point-to-point data transfer operations in high–speed data networking applications such as routers, backplanes and access switches, and storage area network equipment. SERDES devices provide a high–speed bus without a lot of connections on a backplane or cables between boxes. The SER stands for Serializer. It takes parallel data and serializes it into a serial bit stream. The input is typically 8 parallel data, which is, encoded with an optional 8B/10B encoder. This encoding scheme converts the 8 bits data into a 10-bit format that is transmitted over a serial output "Link". The data rate on the link, in this 1 Gigabit per second data input example (Figure 1.), is therefore 1.25 Gigabaud. The deserializer, or DES, works in reverse as it takes the serial data, decodes it and converts it back to a parallel data interface along with a "recovered" data clock.

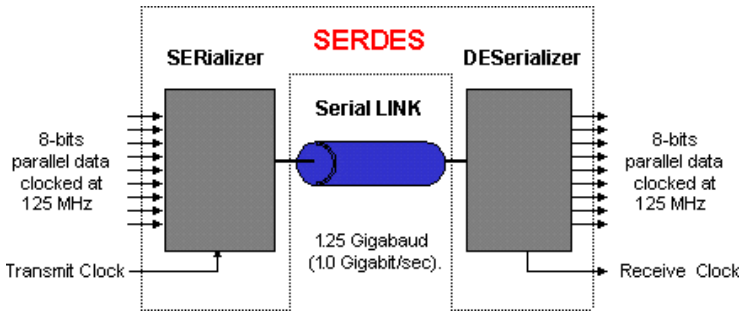

Figure 1. SERDES Transceiver

A generic functional diagram of a SERDES link is shown in Figure 2. The transmitter portion of a SERDES chip has a parallel digital interface, FIFO, 8B/10B encoder (that can be used or bypassed),

[1] From: http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01435940582350NbkQ

and serializer. The transmitter output drives a differential signal into, typically, 50-Ohm media (100 Ohm Differential) and requires no external termination or interface components.

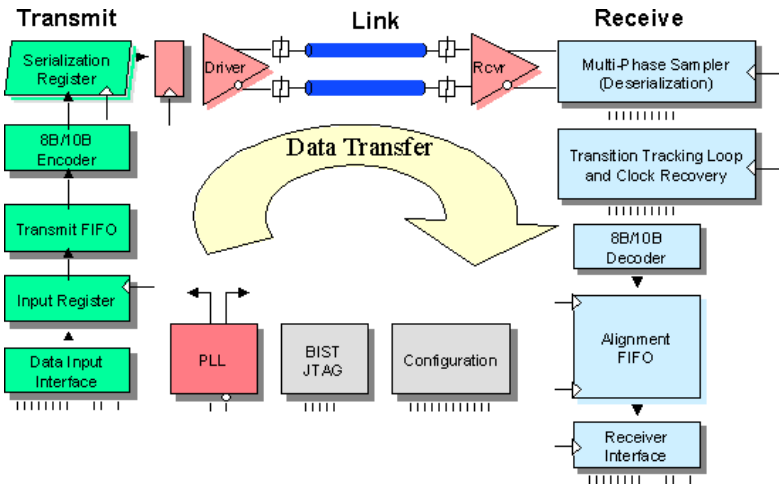

Figure 2. SERDES Functional Diagram

The receiver performs the deserializer function. It has a transition–tracking loop that does the data and clock recovery, along with byte alignment, an 8B/10B decoder, word alignment FIFO and digital parallel data interface.

A simplified block diagram for each SERDES channel with the common chip clock is shown in more detail in Figure 3.

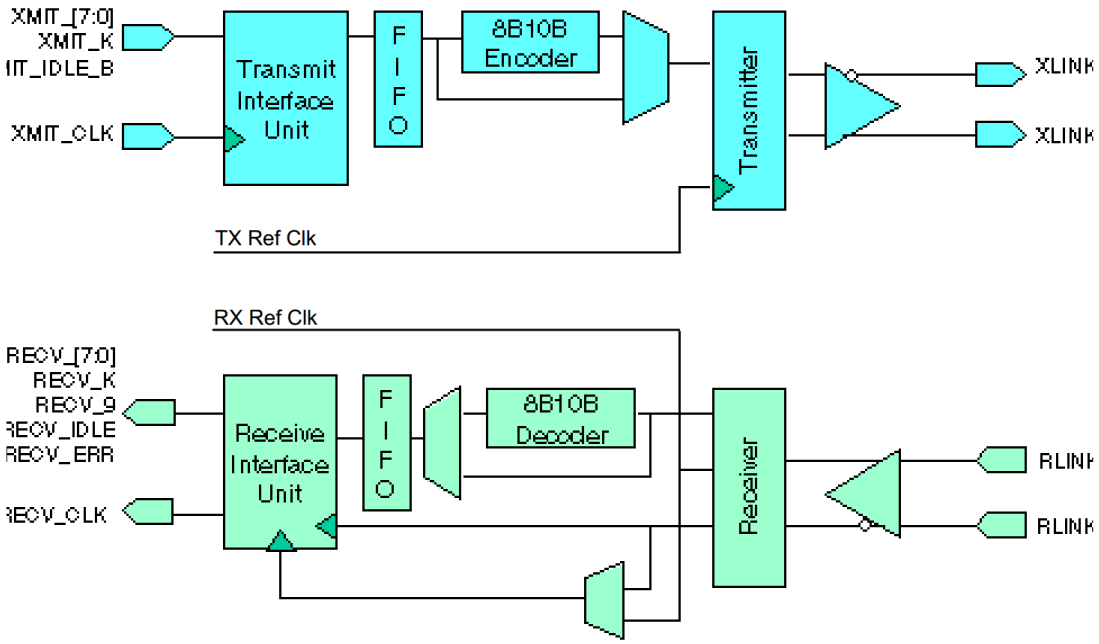

Figure 3. SERDES Block Diagram

Clock and Data Recovery (CDR)

Clock and Data Recovery (CDR) circuits are used to recover serial data from an incoming data stream that often follows a Non-Return to Zero (NRZ) signaling scheme. The main component of a CDR or the Data Recovery (DR) circuit is the Phase-Locked Loop (PLL). Therefore, CDR circuits are broadly classified according to the type of the Phase Detector (PD) used in the PLL. Linear CDR circuits use a Hogge's style Phase Detector and numerous such implementations (full-rate and reduced-rate) have been presented in literature. Non-linear CDR circuits employ Bang-Bang Phase Detectors (BBPDs) that have become popular more recently for multi-Gb/s applications. Traditional CDR circuits (linear or non-linear) perform a 2*x*-oversampling of the incoming data [1]. One of the clock edges is aligned with a PLL to a data edge as the other clock edge samples the data.

Oversampling Clock Data Recovery (CDR) Circuit

For the final project of the course you will be designing an oversampling serial-deserializer (SERDES) based on a multiple rotating clock phase (MRCP) architecture [2]. It is strongly recommended to read references of this lab [1-3]. In a synchronous digital circuit all of the registers in the circuit operate in the same clock domain. That is to say, each of the registers updates at the same time as all of the other registers in the circuit. In larger digital circuits and systems several different clock domains exists and getting data safely across the boundary from one clock domain to another is typically the job of a SERDES. A SERDES' duty is to synchronize data to the systems clock or in some cases even recover the original clock from the serial data stream. If the SERDES has done its job the newly synchronized data should be an exact copy of the original data. An oversampling SERDES is a special type of SERDES that uses a faster clock to track the data. By sampling each bit in the incoming data stream several times it is possible to predict where the next bit transition will occur in the serial input stream. Once the edge can accurately be predicted the recovery of the original data is trivial. The question becomes, how does one predict the timing of an edge, Ahmed and Kwasniewski describe one method in [1, 2] which will be used in this lab.

MRCP Architecture

The multiple rotating clock phase (MRCP) architecture uses three rotating clock phases supplied by a phase generator and the original data stream to recover data and clock (Figure 4 and Figure 5). In this architecture three rotating phases, a fixed distance apart (Figure 4) are used to sample the incoming data. The Early, Edge and Late Phases as shown in Figure are used to clock the three front-end flip flops used for sampling the incoming data stream. This implements a 3xoversampling all digital CDR architecture. The phase detector circuit compares the sampling of these three flip-

flops and decides on rotating the edge phase to right or left. The following sub-sections explain the various blocks of the MRCP architecture in detail. The block diagram of the MRCP CDR architecture appears in Figure 6.

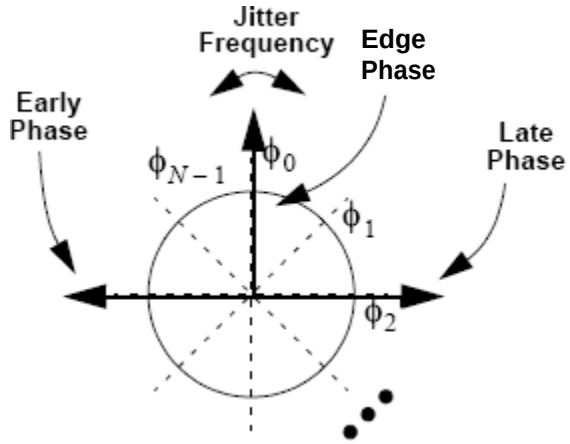

Figure 4. Phase rotation Concept [1]

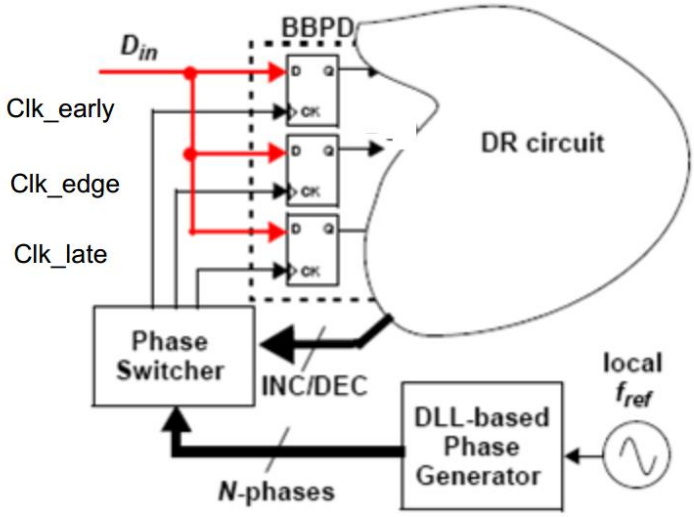

Figure 5. CDR Architecture [1]

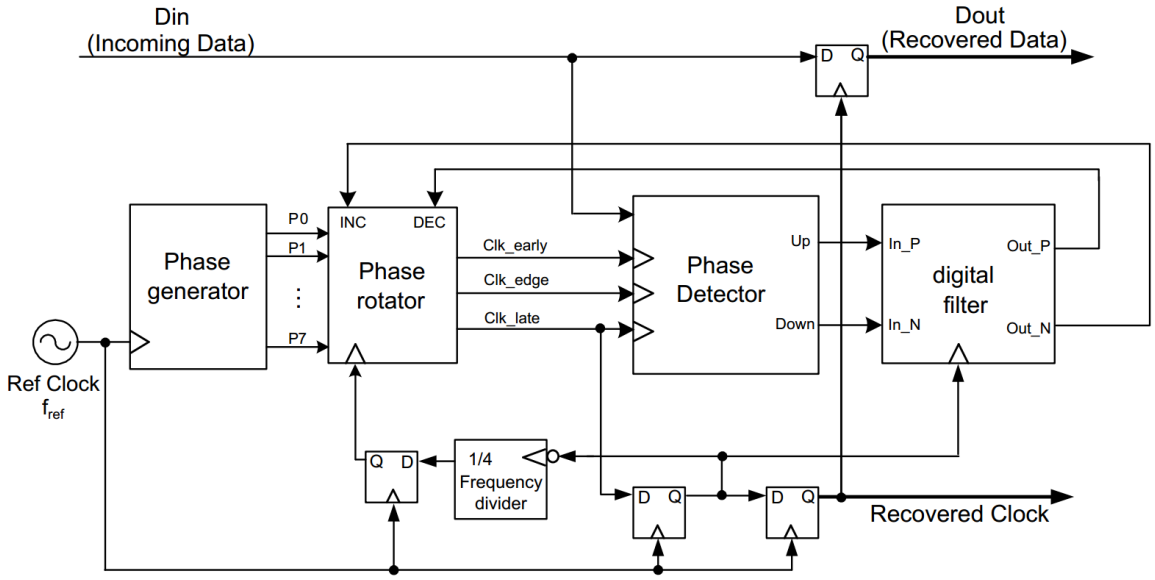

Figure 6. CDR Block Diagram

Phase Generator

The phase generator block provides 8 clock phases (P0, P1, …, P7) required in the phase rotator block. A VCO or DLL can be used to generate the 8 clock phases. In order to simplify the circuit in this lab we use an 8-bit ring counter (rst sets one flip-flop and resets all the rest) to generate the 8 clock signal phases. The counter is clocked at f_{ref} frequency and the frequency of each of the P0, P1, …, P7 clocks is $f_{ref}/8$.

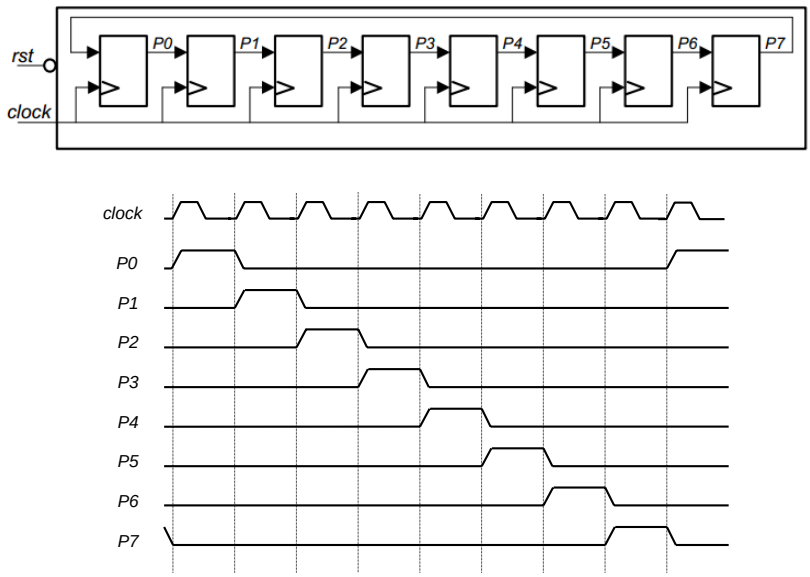

Figure 7. 8-phase clock system

Phase Rotator and Clock Divider

One of several available clock phases is chosen using a cyclic phase rotator (Figure 8). This block operates at a lower frequency (CLK/*4*). Create a clock divider block to generate output clock with the frequency a quarter of input clock (or period four times of input clock). The phase rotator can rotate the clock phases left or right as a result of the INC/DEC command respectively. As the jitter accumulates, a decision is made by the BBPD/Digital Filter to either advance/delay (DEC/INC) the phase. Depending on the direction of the jitter as signalled by INC/DEC, the clock phases are selected on a rotating basis from the eight available clock phases. The routing of the 8 phases to the three multiplexers defines the phase difference between early, edge and late clock phases. In this implementation we assume 90 degrees phase difference which corresponds to a 2 unit difference between the phases.

To maintain glitch-free Clk_early, Clk_edge and Clk_late, the increasing and decreasing of the phase should take place between the falling edge of Clk_late and the rising edge of Clk_early.

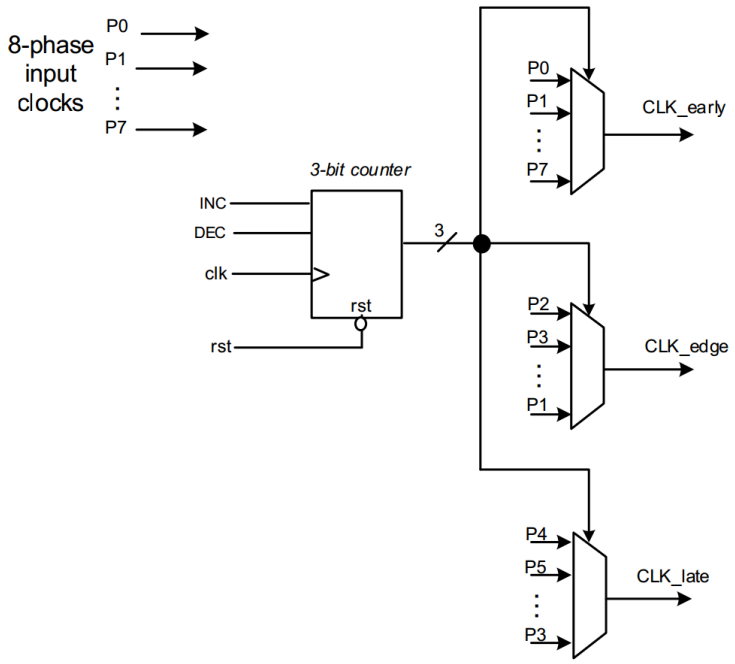

Figure 8. Phase rotator circuit

Bang Bang Phase Detector (BBPD)

The Up and Down outputs from the BBPD provide an indication of the early/late data transitions (Fig. 9). The Up and Down signals are synchronized with Clk_late. Fig. 10 shows the direction of phase rotation as a result of Up/Down signals. In Fig. 11 the conditions for asserting Up and Down signals are shown.

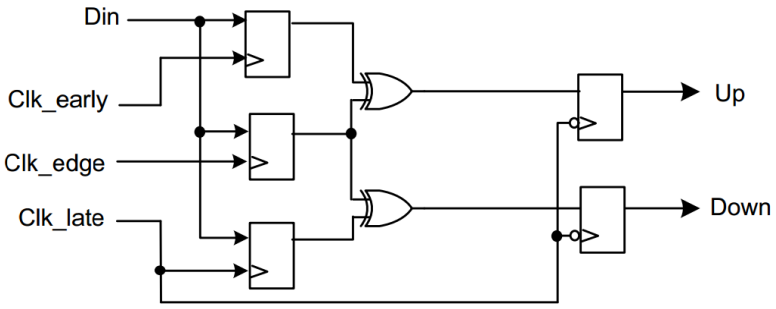

Figure 9. Bang-bang phase detector

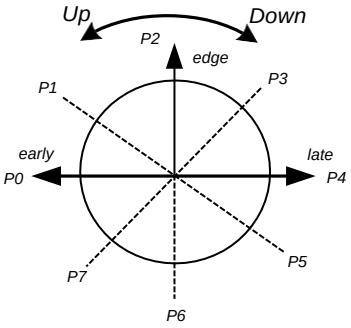

Figure 10. Phase rotation direction for Up/Down signals

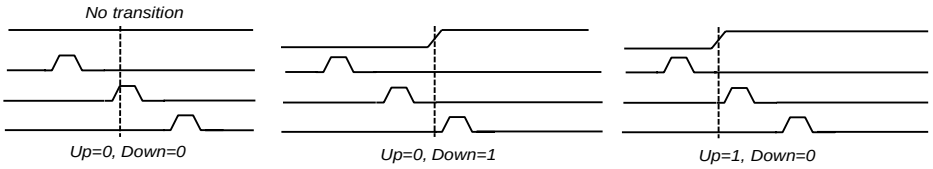

Figure 11. Phase shifting in CDR

Digital Filter

The digital filter is shown in Fig. 12. The outputs, that are never asserted simultaneously, control the direction of phase correction applied to the phase rotator.

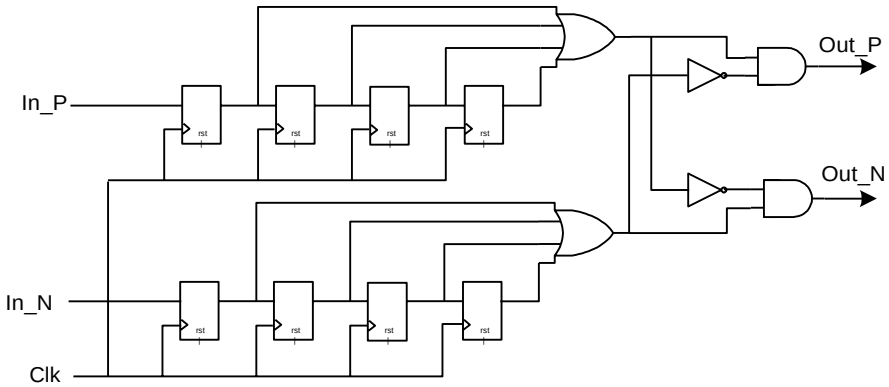

Figure 12. Digital filter

Laboratory Procedure

Part 1

Remotely connect to "kwasniewski" server from your lab station with your VLSI account and copy the Lab_6_Simulink.mdl Simulink model of CDR from the lab manual directory to your drive. Open the local copy in MATLAB 2014a.

Identify:

1. source of data (Tx Data – Bernouli Binary); the red traces are the input signals to all 3 FFs; the input signal can be arriving at the FFs inputs as
    - square binary shape & square binary shape with jitter
    - signal going through a "simple channel" & simple channel with jitter
2. recovered data output at Output of FF1
3. Phi_1 -> recovered clock
4. Shift Register (Loop filter) – Shift Register + AND; this block can be opened to see how it is constructed
5. Decimation and Phase Rotator Blocks; signal to the phase rotator can arrive through Shift Register or directly from XORs of the phase detector
6. Scope block on the far right; the signals top down are:
    - 3 phases of phase detector
    - Received signal
    - Up/down signal arriving at phase rotator input
7. Bottom of the page "Triggered Eye Display" allows observing a changing eye of the received data
8. Error Detection Block compares transmitted data with the recovered data and produces an error signal

Run simulation, observe behavior:

1. Clock aligning from initial value
    - Start simulation
    - Open right most 5 input scope
    - Observe how the second top trace (Rx sampling phase phi_1) aligns itself to the middle of he received data (4^{th} trace); it takes approx 3e-4 seconds (see bottom right time counter); note the time elapsed.
2. Eye diagram Shape in Time
    - Restart the simulation

- Observe the eye diagram (Triggered Eye Diagram) for 5e-4s
- On the eye diagram click “Axis” and “Refresh”. Wait for another 5e-4s
- Capture eye-diagrams and comment on their shape.

3. Eye Diagram without the noise and without the channel, keep jitter on
   - Disable noise and channel but not jitter
   - Observe behavior, capture eye-diagram and make notes on the shape
4. Eye Diagram without the noise and without the jitter, keep channel on
   - Disable noise and jitter but not channel;
   - Observe behavior, capture eye-diagram and make notes on the shape;
   - At how many different amplitude levels the transition to a new data symbol starts?
5. The effect of Loop Filter
   - Add the Loop Filter “Shift Register + AND” block
   - Observe behavior, capture eye-diagram and make notes on the shape
6. Propose two tests of your choice
   - State what behavior you are verifying/ observing and the justification of your selection.
   - Perform simulations and report results

Part 2

Implement (code) each one of the 5 blocks of the CDR (Phase Generator, Phase Rotator, Phase Detector, Filter, and Divider) as a component.
Simulate (functional) each component and troubleshoot until you can verify its proper operation. First, complete the phase generator. Then complete the clock divider. Then, use the completed phase generator and clock divider for driving and verifying your phase rotator. Then, use the combination of the phase generator, clock divider and phase rotator for driving your phase detector. The digital filter can be simulated stand-alone.
Demonstrate simulation waveforms for relevant test cases for each one of the 5 blocks.

Lab Report

- For part 1, include all captured eye-diagrams with comments on the shapes. Discuss the effect of loop filter. Propose two tests with simulation and results.
- For part 2, include function simulation of 5 blocks.

References

[1] S. I. Ahmed, Tad A. Kwasniewski, "Overview of Oversampling Clock and Data Recovery Circuits," Canadian Conference on Electrical and Computer Engineering, IEEE CCECE05, Saskatoon, May 2005.

[2] Ahmed, S. I. and Kwasniewski, T. A., “A Multiple Rotating Clock Phase Architecture for Digital Data Recovery Circuits using VerilogA,” in *BMAS*, 2005, pp. 112117.

[3] Y. Miki *et. al*, “A 50-mW/ch 2.5-Gb/s/ch Data Recovery Circuit for the SFI-5 Interface with Digital Eye-Tracking,” *IEEE J. Solid-State Circuits,* vol. 39, no. 4, Apr. 2004, pp. 613-621.