Reconfigurable Synchronization Technique

X’CON 2005 Beijing
cawan@ieee.org
Overview

- Transceiver Modules
- Synchronization Technique
- FPGA Based System Design
- New Hybrid-Reset Algorithm In FPGA
- Soft Processor Interfacing Technique
- Advantages of Reconfigurable System
- Demo

Lets Start The Journey of Hardware System Hacking !!!
Transceiver Modules
Synchronization Unit

- Deserializer
- CRU (CDR)
- Reference Clock
- Rx PLL

- Receiver PLL
  - Generates CRU Reference Clock
- CRU (CDR)
  - Clock Recovery Unit
Synchronization Technique (1)

A method to ensure the data regeneration is done with a decision circuit that samples the data signal at the optimal instant indicated by a clock.

Clock 2 = \text{F\{Communication Channel, Local Clock Source\}}
Synchronization Technique (2)

Essential of Transceiver Unit
Commonly Known As CDR Or CRU
Examples of Standard Interfaces:

- 10/100-Based-T
  - http://www.holmea.demon.co.uk/Ethernet/EthernetRx.htm

- ASI
Implementation Methods

Two Methods:

- PLL Based
  - Hardcore PLL
  - Softcore PLL (ADPLL)
- Oversampling Based
  - Frequency Triggered
  - Phase Triggered
PLL Based Synchronization

- **Hardcore PLL**

- **Softcore PLL (ADPLL)**
Oversampling Based

- **Frequency Triggered**

- **Phase Triggered**
PLL Based CRU Model

PLL Uses Transitions in the Serial Data to Generate a Recovered Clock
Oversampling Based CRU Model

- Oversampling Algorithm Only Extracts Recovered Data From Serial Data. Clock Is Not Recovered.
- A Local Clock Is Used As System Sampling Clock
- Data_Valid Prompts The Availability of Recovered Data
FPGA Based System Design

A Programmable Logic Platform To Implement Digital Systems

Integrated Hard Processor, RAM, PLL, etc.

EP2S60

EP1S40
HDL Design In Verilog

- Started From 1981
- Hardware Description Language
- True Abstract Behavior Modeling
- Hardware Structure Modeling
- C-Like Syntax
- Defined In IEEE Standard 1364
New Hybrid-Reset Algorithm In FPGA

- A New Algorithm (Hybrid-Reset) Is Developed
- Problem Statements of Current Synchronization Techniques
  - PLL Based
    - Hardcore PLL
      - Fixed Architecture
    - Softcore PLL
      - High Resource Utilization
  - Oversampling Based
    - Frequency Triggered
      - High Sampling Frequency
      - No Recovered Clock
    - Phase Triggered
      - High Cost of Frequency Synthesis
      - No Recovered Clock
Inefficiency of Current Oversampling Algorithm (1)

**4x Oversampling**

- Sample data
- Edges found? (Y/N)
- Update reference pointer
- Determine valid samples

**Jitter Tolerance = +/- 0.25UI**
Inefficiency of Current Oversampling Algorithm (2)

Why Recovered Clock Is Not Generated In Oversampling?
Because The Center of Data Eye Is Missing !!!

\[ \text{Max } (\Delta t_2 - \Delta t_1) \approx T_{CLK} \]

Recovered Clock Is Probably Leading Or Lagging \(0^\circ < \theta < 90^\circ\)

If \( \Delta t_1 < T_{su} \), Metastability Is Generated

The Recovery Time of Metastability Is Always Less Than \(T_{CLK}\)

The Recovered Bit of Metastability Is Randomly “1” or “0”
Two Level of Flip-Flop to Block The Spreading of Metastability

- Use second level of flip-flop
- Blocks the spreading of metastability to the following sequential or combinational logic.
- The following cycle after the metastability might not be the same as the original.
Hybrid-Reset Mechanism (1)

- Combine The Design Concept of Asynchronous And Synchronous Logic
- Utilize The Nature of Asynchronous Logic For Fast Reset
- The Recovered Clock Is Always Lagging $\Delta T_{co}$ From The Center of Data Eye
- Jitter Tolerance +/- 0.25
Hybrid-Reset Mechanism (2)

Snap-Shot of Source Code In Verilog:

```verilog
// define input and output
input data_in;
input clk;
input rst;
output data_buf;
// asynchronous edge detector
wire reset = (rst & ~data_in & capture_buf);
// data oversampling module
reg capture_buf;
always @ (posedge clk or negedge rst)
if (rst == 0)
capture_buf <= 0;
else
capture_buf <= data_in;
// edge detection module
reg [1:0] mclk_div;
always @ (posedge clk or negedge reset or posedge reset)
if (reset == 0)
mclk_div <= 2'000;
else
mclk_div <= mclk_div + 1;
// capture at data eye and put into a 16-bit buffer
reg [15:0] data_buf;
always @ (posedge mclk_div[1] or negedge rst)
if (rst == 0)
data_buf <= 0;
else
data_buf <= {data_buf[14:0], capture_buf};
```
Soft Processor Interfacing Technique (1)

**What Is Soft Processor?**
- A Synthesizable Core That Can Be Targeted Into Different Semiconductor Fabrics
- Advantages of Soft Processor:
  - Flexible Implementation (In Verilog)
  - Adaptive Platform
- Examples:
  - Nios (Altera)
  - Microblaze (Xilinx)

**Question: Hard Processor Always Faster Than Soft Processor?**
- Not Really, Processor Performance Is Often Limited By How Fast The Instruction And Data Can Be Pipelined From External Memory Into Execution Unit
Soft Processor Interfacing Technique (2)

How To Interface? Bus Specification!!!

Principle Design Goals:
✧ Low Resource Utilization For Bus Logic
✧ Simplicity
✧ Synchronous Operation

Avalon Bus Specification With Nios Processor

Avalon Bus

32-Bit Nios Processor

Address (32)
Read
Write
Data In (32)
Data Out (32)

Switch PIO

LED PIO

7-Segment LED PIO

Network

User-Defined Interface

CRU

ROM (with Monitor) UART Timer
Advantages of Reconfigurable System

- Offer The Possibility To Penetrate Into A Customize System, example:
  - 12Mbps, NRZ, TDM with 4 Time Slots
- Hardware Reconfigurable Through Network
- TFTP
- True Parallel Processing Architecture
- DSP out of the topic in hacking !!!
- Custom Instructions good for cracking purpose !!!

http://www.cl.cam.ac.uk/users/rnc1/descrack/index.html
Custom Instructions

- Hardware Based Instruction Set
- Accelerates Software Algorithms
- Very Suitable To Implement Library of Cracking Instruction Set
Demo

- Synchronize Into A Customize System
- By Knowing The Communication Spec...
  - 12Mbps
  - NRZ Coding of PCM Audio
  - TDM With 4 Time Slots
- Lets Start...
Thanks You

Q & A