Lecture 9-10: Computer Network

Practical: Design of Physical Layer Components

New 2020 Syllabus

Unit –I

Computer Network: Definitions, goals, components, Architecture, Classifications & Types.Layered Architecture: Protocol hierarchy, Design Issues, Interfaces and Services, ConnectionOriented & Connectionless Services, Service primitives, Design issues & its functionality. ISOOSI Reference Model: Principle, Model, Descriptions of various layers and its comparison with TCP/IP. Principals of physical layer: Media, Bandwidth, Data rate and Modulations

Unit-II

Data Link Layer: Need, Services Provided, Framing, Flow Control, Error control. Data Link Layer Protocol: Elementary &Sliding Window protocol: 1-bit, Go-Back-N, Selective Repeat, Hybrid ARQ. Protocol verification: Finite State Machine Models & Petri net models. ARP/RARP/GARP

Unit-III

MAC Sub layer: MAC Addressing, Binary Exponential Back-off (BEB) Algorithm, Distributed Random Access Schemes/Contention Schemes: for Data Services (ALOHA and Slotted- ALOHA), for Local-Area Networks (CSMA, CSMA/CD, CSMA/CA), CollisionFree Protocols: Basic Bit Map, BRAP, Binary Count Down, MLMA Limited Contention Protocols: Adaptive Tree Walk, Performance Measuring Metrics. IEEE Standards 802 series & their variant.

BIRLA INSTITUTE OF APPLIED SCIENCES
BIAS

New 2020 Syllabus

Unit-IV

Network Layer: Need, Services Provided, Design issues, Routing algorithms: Least CostRouting algorithm, Dijkstra's algorithm, Bellman-ford algorithm, Hierarchical Routing,Broadcast Routing, Multicast Routing. IP Addresses, Header format, Packet forwarding,Fragmentation and reassembly, ICMP, Comparative study of IPv4 & IPv6

Unit-V

Transport Layer: Design Issues, UDP: Header Format, Per-Segment Checksum, CarryingUnicast/Multicast Real-Time Traffic, TCP: Connection Management, Reliability of DataTransfers, TCP Flow Control, TCP Congestion Control, TCP Header Format, TCP TimerManagement. Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

About Course Instructor

- PhD from Gran Sasso Science Institute, Italy
- PhD Supervisor Prof Paolo Prinetto from Politecnico Di Torino, World Rank 13 in Electrical Engineering
- MTech from Indian Institute of Information Technology, Gwalior
- Scopus Profile: https://www.scopus.com/authid/detail.uri?authorId=57203239026
- Google Scholar: https://scholar.google.com/citations?user=UZ_8yAMAAAAJ&hl=hi
- Contact: gyancity@gyancity.com, +91-7428640820 (For help in this Subject @ BIAS and Guidance for future MS from Europe and USA after BIAS)

About Course Outline

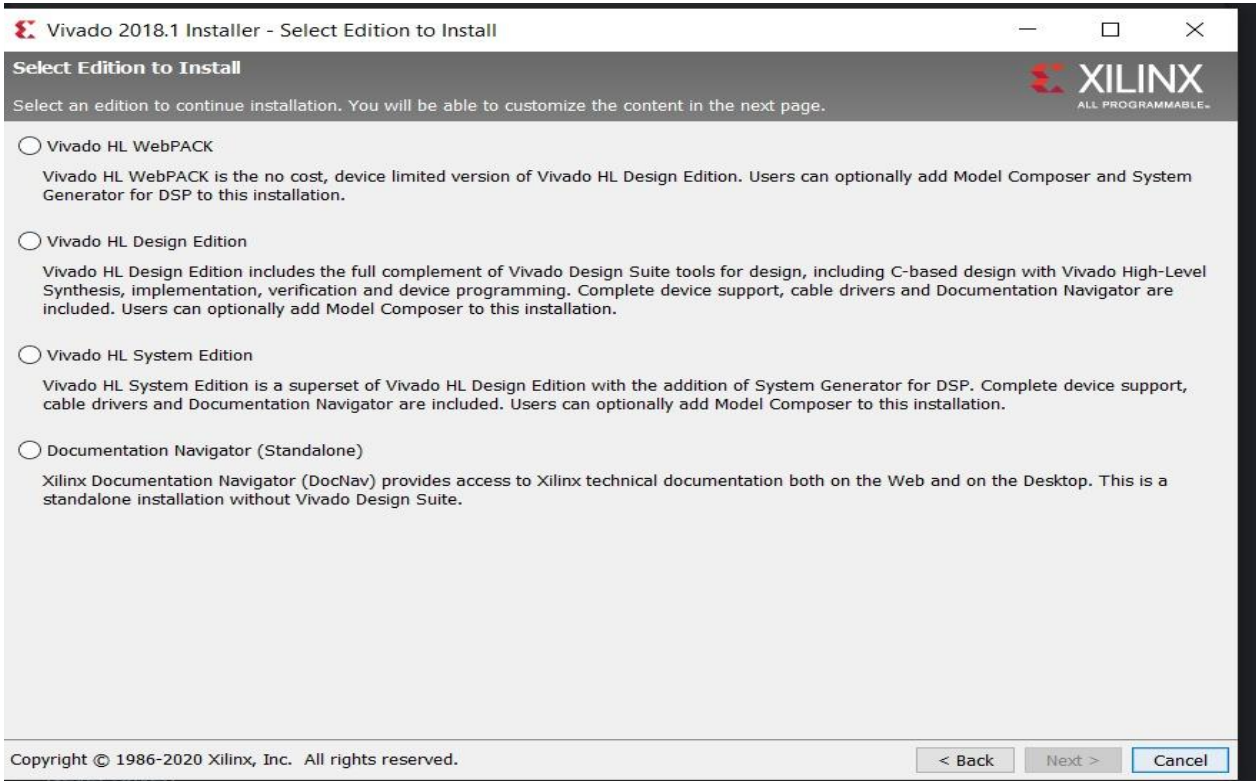
- UNIT 1: Lecture No 1-4
- UNIT 2: Lecture No 5-8
- Practical of UNIT 1 and 2: Lecture 9-10 on Vivado ISE
- UNIT 3: Lecture No 11-14
- UNIT 4: Lecture No 15-20
- Practical of UNIT 3 and 4: Lecture 21-22 on Packet Tracer
- UNIT 5: Lecture No 23-27
- Lecture No 28-35 to Discuss University Question Paper of Previous Years, or Gate Question of Previous Years or Presentation of Research Papers
- Out of 35 Lectures: Some will delivered by Professor From Foreign University

LECTURE 9-10: OUTLINE

- Practical: Physical Layer Component Design
  - Packet Counter
  - UART: Universal Asynchronous Receiver and Transmitter
  - FIR Low Pass Filter
  - Key Generator: Fibonacci Generator
  - Parity Generator,
  - CRC Module

Vivado

- Download Link: https://www.xilinx.com/member/forms/download/xef-vivado.html?filename=Xilinx_Vivado_SDK_Web_2018.1_0405_1_Win64.exe



Select

Vivado HL WebPACK

Verilog Code of Packet Counter

```
module packetcounter(packetin,
clock, count, packetout);
    input [31:0] packetin;
    input clock;
    output [5:0] count;
    output [31:0]  packetout;
    reg [5:0] count;
    reg [31:0]  packetout;

always@(posedge clock)
begin
count<=count+1;
packetout<=packetin;
end
endmodule
```

Verilog Code of UART

```
module uart (reset,
txclk,ld_tx_data,tx_data,tx_enable,tx_out,tx_empty,rxclk,uld_rx_d
ata,rx_data,rx_enable,rx_in,rx_empty);
// Port declarations
input        reset          ;
input        txclk          ;
input        ld_tx_data     ;
input  [7:0] tx_data        ;
input        tx_enable      ;
output       tx_out         ;
output       tx_empty       ;
input        rxclk          ;
input        uld_rx_data    ;
output [7:0] rx_data        ;
input        rx_enable      ;
input        rx_in          ;
output       rx_empty       ;
```

Verilog Code of UART

```
// Internal Variables
reg [7:0]   tx_reg        ;
reg         tx_empty      ;
reg         tx_over_run   ;
reg [3:0]   tx_cnt        ;
reg         tx_out        ;
reg [7:0]   rx_reg        ;
reg [7:0]   rx_data       ;
reg [3:0]   rx_sample_cnt ;
reg [3:0]   rx_cnt        ;
reg         rx_frame_err  ;
reg         rx_over_run   ;
reg         rx_empty      ;
reg         rx_d1         ;
reg         rx_d2         ;
reg         rx_busy       ;
```

BIRLA INSTITUTE OF APPLIED SCIENCES
BIAS

Verilog Code of UART

```
// UART RX Logic
always @ (posedge rxclk or posedge reset)
if (reset) begin
  rx_reg        <= 0;
  rx_data       <= 0;
  rx_sample_cnt <= 0;
  rx_cnt        <= 0;
  rx_frame_err  <= 0;
  rx_over_run   <= 0;
  rx_empty      <= 1;
  rx_d1         <= 1;
  rx_d2         <= 1;
  rx_busy       <= 0;
end else begin
  // Synchronize the asynch signal
  rx_d1 <= rx_in;
  rx_d2 <= rx_d1;
```

Verilog Code of UART

```
// Uload the rx data
if (uld_rx_data) begin
  rx_data  <= rx_reg;
  rx_empty <= 1;
end
// Receive data only when rx is enabled
if (rx_enable) begin
  // Check if just received start of frame
  if (!rx_busy && !rx_d2) begin
    rx_busy       <= 1;
    rx_sample_cnt <= 1;
    rx_cnt        <= 0;
  end
  // Start of frame detected, Proceed with rest of data
  if (rx_busy) begin
     rx_sample_cnt <= rx_sample_cnt + 1;
     // Logic to sample at middle of data
     if (rx_sample_cnt == 7) begin
```

Verilog Code of UART

```
if ((rx_d2 == 1) && (rx_cnt == 0)) begin
  rx_busy <= 0;
end else begin
  rx_cnt <= rx_cnt + 1;
  // Start storing the rx data
  if (rx_cnt > 0 && rx_cnt < 9) begin
    rx_reg[rx_cnt - 1] <= rx_d2;
  end
  if (rx_cnt == 9) begin
     rx_busy <= 0;
     // Check if End of frame received correctly
     if (rx_d2 == 0) begin
       rx_frame_err <= 1;
     end else begin
       rx_empty     <= 0;
       rx_frame_err <= 0;
       // Check if last rx data was not unloaded,
       rx_over_run  <= (rx_empty) ? 0 : 1;
```

Verilog Code of UART

```
                end
              end
            end
          end
        end
      end
    end
    if (!rx_enable) begin
      rx_busy <= 0;
    end
  end
```

Verilog Code of UART

```
// UART TX Logic
      always @ (posedge txclk or posedge reset)
      if (reset) begin
        tx_reg       <= 0;
        tx_empty     <= 1;
        tx_over_run  <= 0;
        tx_out       <= 1;
        tx_cnt       <= 0;
      end else begin
        if (ld_tx_data) begin
          if (!tx_empty) begin
            tx_over_run <= 0;
          end else begin
            tx_reg   <= tx_data;
            tx_empty <= 0;
          end
        end
```

Verilog Code of UART

```
        if (tx_enable && !tx_empty) begin
                tx_cnt <= tx_cnt + 1;
                if (tx_cnt == 0) begin
                  tx_out <= 0;
                end
                if (tx_cnt > 0 && tx_cnt < 9) begin
                   tx_out <= tx_reg[tx_cnt -1];
                end
                if (tx_cnt == 9) begin
                  tx_out <= 1;
                  tx_cnt <= 0;
                  tx_empty <= 1;
                end
              end
              if (!tx_enable) begin
                tx_cnt <= 0;
              end
            end
endmodule
```

Verilog Code of FIR Low Pass Filter

```
module FIR_Gaussain_Lowpass(Data_out, Data_in, clock, reset);
parameter order=8;
parameter word_size_in=8;
parameter word_size_out=2*word_size_in+2;
parameter b0=8'd7;
parameter b1=8'd17;
parameter b2=8'd32;
parameter b3=8'd46;
parameter b4=8'd52;
parameter b5=8'd46;
parameter b6=8'd32;
parameter b7=8'd17;
parameter b8=8'd7;
```

Verilog Code of FIR Low Pass Filter

```
output [word_size_out-1:0] Data_out;
input [word_size_in-1:0] Data_in;
input clock, reset;
reg [word_size_in-1:0] Samples[1:order];
integer k;
assign Data_out=b0*Data_in
+ b1*Samples[1]
+ b2*Samples[2]
+ b3*Samples[3]
+ b4*Samples[4]
+ b5*Samples[5]
+ b6*Samples[6]
+ b7*Samples[7]
+ b8*Samples[8];
```

Verilog Code of FIR Low Pass Filter

```
always @(posedge clock)
if(reset==1)
begin
for(k=1; k<=order; k=k+1)
Samples[k]<=0;
end
else
begin
Samples[1]<=Data_in;
for(k=2; k<=order; k=k+1)
Samples[k]<=Samples[k-1];
end

endmodule
```

Verilog Code of Key Generator: Fibonacci Generator

```
module FibGen(clk, rst, enb, out);
input clk, rst, enb;
output [16:0] out;
reg [16:0] out;
   // states
   parameter S0 = 3'h000;
   parameter S1 = 3'b001;
   parameter S2 = 3'b010;
   parameter S3 = 3'b011;
   parameter S4 = 3'b100;

   // used to initialize registers
   parameter Zero_16 = 16'b0000000000000000;
   parameter One_16 = 16'b0000000000000001;
```

Verilog Code of Key Generator: Fibonacci Generator

```
reg [16:0] reg_0 = Zero_16;
reg [16:0] reg_1 = One_16;
reg [16:0] fib = Zero_16;
reg [2:0] State;
always @ (posedge rst or posedge clk)
begin
   if( rst == 1 )
   begin
      reg_0 <= Zero_16;
      reg_1 <= One_16;
      fib <= Zero_16;
      State <= S0;
      out <= Zero_16;
   end
   else
```

Verilog Code of Key Generator: Fibonacci Generator

```
    begin
case( State )
  S0:
                begin
    // determine next state
    if( enb == 1 )
      State <= S1;
    else
      State <= S0;
    // assign output value
    out <= reg_0;
    fib <= reg_0;
  end
  S1:
                begin
```

Verilog Code of Key Generator: Fibonacci Generator

```
// determine next state
        if( enb == 1 )
           State <= S2;
        else
           State <= S1;
// assign output value
        out <= reg_1;
        fib <= reg_1;
     end
     S2:
                    begin
        // determine next state
        if( enb == 1 )
           State <= S2;
        else
```

Verilog Code of Key Generator: Fibonacci Generator

```
            State <= S3;
          // update values and assign output value
          out <= reg_0 + reg_1;
          fib <= reg_0 + reg_1;
          reg_0 <= reg_1;
          reg_1 <= reg_0 + reg_1;
        end
        S3:
                        begin
          // determine next state
          if( enb == 1 )
            State <= S2;
          else
            State <= S3;
         // assign output value
          out <= fib;
        end
                    endcase
    end
  end
endmodule
```

Publication In Green Communication: Suggested Reading

- Singh, Sunny, et al. "Thermal aware low power **universal asynchronous receiver transmitter** design on FPGA." *2014 International Conference on Computational Intelligence and Communication Networks*. IEEE, 2014.
- **Kumar, Keshav,** et al. "Effect of Different Nano Meter Technology Based FPGA on Energy Efficient **UART** Design." *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2018.
- Singh, P. R., et al. "Output load capacitance based low power implementation of **UART** on FPGA." *2014 International Conference on Computer Communication and Informatics*. IEEE, 2014.
- Sandhu, Amanpreet, et al. "Thermally aware LVCMOS based low power **universal asynchronous receiver transmitter** design on FPGA." *Indian Journal of Science and Technology* 8.20 (2015): 1-4.

Publication In Green Communication: For Reading

- Sharma, Rashmi, et al. "Input–Output Standard-Based Energy Efficient **UART** Design on 90ánm FPGA." *System and Architecture*. Springer, Singapore, 2018. 139-150.
- Sharma, Rashmi, et al. "Voltage Scaling Based Wireless LAN Specific UART Design Based on 90nm FPGA." *International Journal of Smart Home* 10.3 (2016): 131-138.
- Kumar, Tanesh, B. Pandey, and Teerath Das. "Voltage scaling based energy efficient **FIR filter** design on FPGA." International Journal of Current Engineering and Technology. Special Issue-3(2014): 200-204.
- **Keshav Kumar,** Bishwajeet Pandey, D M Akbar Hussain, “Effect of Frequency on Energy Efficient **Transceiver** Design”, Gyancity Journal of Engineering and Technology, Vol.5, No.2, pp. 14-18, July 2019 ISSN: 2456-0065 http://gjet.gyancity.com/Vol5No2/GJET_Vol5No2_July2019_002.pdf

Publication In Green Communication: For Reading

- Pandey, Bishwajeet, et al. "Scaling of output load in energy efficient **FIR filter** for green communication on ultra-scale FPGA." *Wireless Personal Communications* 106.4 (2019): 1813-1826.
- Kumar, Abhishek, et al. "Low Voltage Complementary Metal Oxide Semiconductor Based Energy Efficient **UART** Design on Spartan-6 FPGA." *2019 11th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2019.
- **Kumar, Keshav**, et al. "Low Power **UART** Design Using Different Nanometer Technology Based FPGA." *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2018.
- Gupta, Isha, et al. "28nm FPGA based Power Optimized **UART** Design using HSTL I/O Standards." *Indian Journal of Science and Technology* 8.17 (2015): 1-6.

Publication In Green Communication: For Reading

- Kumar, Abhishek, et al. "Frequency Scaling and High Speed Transceiver Logic Based Low Power **UART** design on 45nm FPGA." *2019 11th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2019.
- Pandey, Bishwajeet, et al. "Performance evaluation of **FIR filter** after implementation on different FPGA and SOC and its utilization in communication and network." Wireless Personal Communications 95.2 (2017): 375-389.
- Pandey, Bishwajeet, et al. "Thermal mechanics based energy efficient **FIR filter** for digital signal processing." Applied Mechanics and Materials. Vol. 612. Trans Tech Publications Ltd, 2014.

Publication In Green Communication: For Reading

- Kumar, Tanesh, et al. "**Mobile DDR** IO standard based high performance energy efficient portable ALU design on FPGA." *Wireless Personal Communications* 76.3 (2014): 569-578.
- Musavi, Sayed Hyder Abbas, et al. "IoTs enable active contour modeling based energy efficient and thermal aware object tracking on FPGA." *Wireless Personal Communications* 85.2 (2015): 529-543.
- Shivani Sharma, “Energy Efficient Sustainable **Communication System** Design for Space Craft Operating in the Coldest Places of Solar System”, Gyancity Journal of Engineering and Technology Vol.1 No.1 January 2015 ISSN: 2456-0065 http://gjet.gyancity.com/Vol1No1/5.pdf
- Vandana Thind, Shivani Sharma, M H Minwer, D M Akbar Hussain, “Junction Temperature Aware Energy Efficient **Router** Design on FPGA”, Gyancity Journal of Engineering and Technology, Vol.1, No.1, January 2015 48 ISSN: 2456-0065 http://gjet.gyancity.com/Vol1No1/6.pdf

Publication In Green Communication: For Reading

- Vandana Thind, DM Akbar Hussain, "FPGA Based Low Power **Router** Design Using High Speed Transeceiver Logic IO Standard ", Gyancity Journal of Engineering and Technology, Vol.1, No.2, July 2015 24 ISSN: 2456-0065 http://gjet.gyancity.com/Vol1No2/5.pdf
- Tarun Singhal, Abhishek Shrivastava, Palash Jain, Rahul, Gaurav Verma, "**Smart Communication Network design** with application of Energy Efficient Digital Clock for Monitoring of Time -To-Live", Gyancity Journal of Engineering and Technology, Vol.3, No.1, pp. 23-28, January 2017 ISSN: 2456-0065 DOI: 10.21058/gjet.2017.31004 http://gjet.gyancity.com/Vol3No1/4.pdf
- **Keshav Kumar,** Bishwajeet Pandey, D M Akbar Hussain, "Power Efficient **UART** Design Using Capacitive Load on Different Nanometer Technology FPGA", Gyancity Journal of Engineering and Technology, Vol.5, No.2, pp. 1-13, July 2019 ISSN: 2456-0065 DOI: 10.21058/gjet.2019.52001 http://gjet.gyancity.com/Vol5No2/GJET_Vol5No2_July2019_001.pdf

Publication In Green Communication: For Reading

- **Keshav Kumar,** Bishwajeet Pandey, Jason Levy, Sri Chusri Haryanti, D M Akbar Hussain, “Design of Voltage Scaling Based Power-Efficient **UART** and its Implementation on 7-Series FPGA”, Gyancity Journal of Engineering and Technology, Vol.6, No.1, pp. 1-8, January 2020 ISSN: 2456-0065 http://gjet.gyancity.com/Vol6No1/GJETVol6No1002.pdf
-

Assignment

Total Marks: 15 Marks

Deadline of Submission: 15th September 2020

Q.1 Write a verilog code of any of the following circuits: (5 Marks)

- FIR Low Pass Filter
- Key Generator: Fibonacci Generator
- UART Circuit

Q.2 Compile Verilog code of physical layer component (Q.1) in Vivado and generate its Schematic. (2 Marks)

Q.3 Make a group of three students, read many research paper in computer network and make a presentation of 15 minutes on one research paper, present in group of three (5 minutes for every group member). (8 Marks)

OR

Q.3 Make Physical Layer component in Q.1 more Energy Efficient than the existing one. Finally write a research paper on that, and present it. (8 Marks)

BIRLA INSTITUTE OF APPLIED SCIENCES

BIAS