ECE1373S: Project Presentation

FPGA Shell

Omar Samhan
Nariman Eskandari
Meng Liu

Introduction

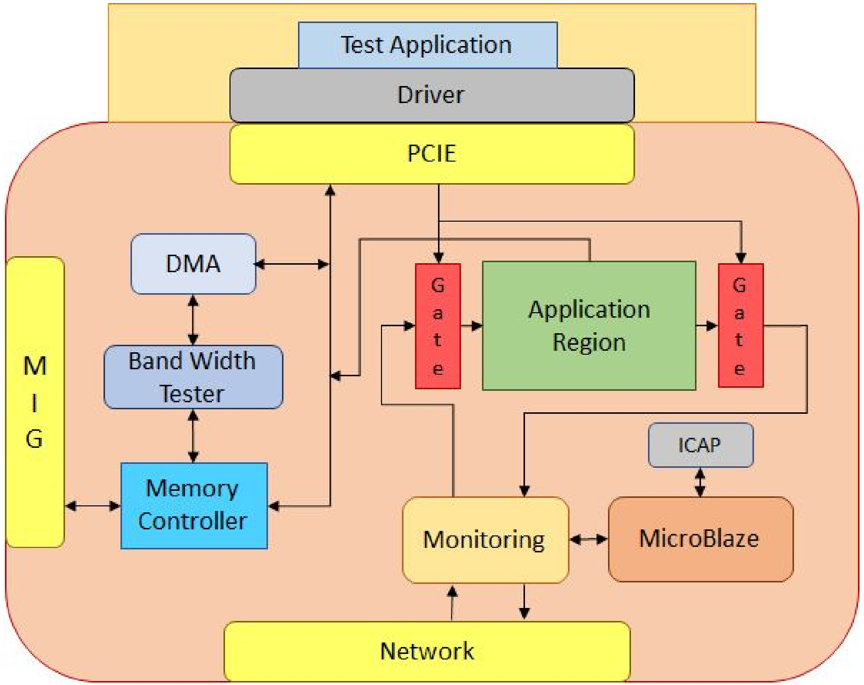
- FPGAs are useful in computation world
  - Low power
  - High performance
  - Microsoft Catapult
- Why FPGAs are not widely used?
  - Special purpose!
  - Hard to program in comparison with general purpose computation resources
  - Interfacing!
- Solutions?!
  - High Level Synthesis
  - FPGA shell

FPGA shell Goals

- Make life easier for FPGA users!
    - Implementing all the interfaces that a user may need
        - DDR memory
        - PCIe
        - Network
    - Providing monitoring information
        - PCIe bandwidth
        - Network load
        - Temperature
    - Providing Hardware debug capability
        - Xilinx Integrated Logic Analyzer
        - Xilinx Virtual Input output

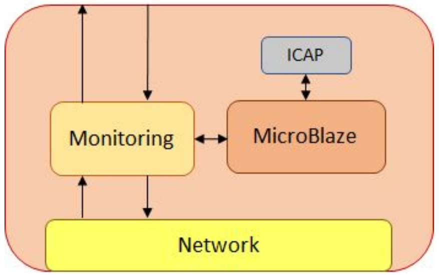
In This Shell

- This shell is built for Alpha Data 8k5 boards.
- Interfaces are implemented:
  - DDR memory
  - PCIe
  - Network
    - Ethernet
    - UDP
    - TCP
- Monitoring
  - Network
  - PCIe

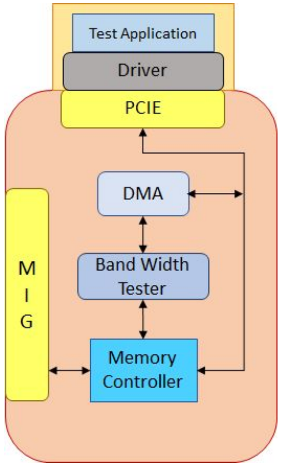


Project Organization

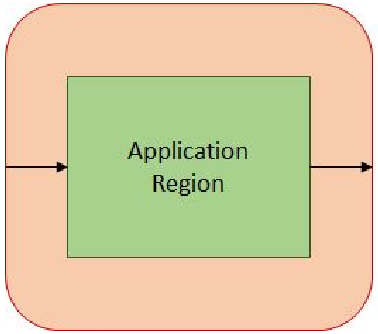
- This project is divided into three parts
  - Network and Monitoring
  - PCIe and DDR
  - Min Sum Decoder Application



Nariman



Meng



Omar

Testing: Shell

- The system is too big!
- Simulation of PCIe and Network is tricky because of their inputs!
- Xilinx ILA and VIO for debugging
- Python scripts for sending UDP packets, and loopback application is used in hardware.
- Python scripts for sending TCP packets, and an echo application is used in hardware.
- C code for PCIe testing to write to DDR memory and read data back and compare.

Testing: Application

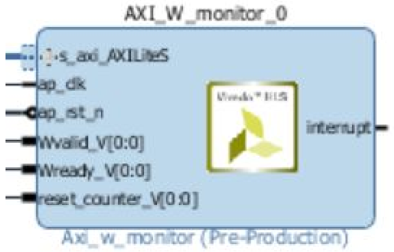
- Compare decoder estimates from min-sum Matlab model
  - Use the Matlab values inputs and outputs
  - Use the same Matlab inputs to the HLS decoder
  - Compare decoder output
- ILA
- Testbench for UDP

Testing: PCIe Gen3 to DDR4

- Apply Xilinx IP of DMA/Bridge Subsystem for PCIe: 8 lanes x 8GT/s, Memory mapped AXI interface
- Apply Xilinx IP DDR4 MIG with configuration file from Alpha data: 64/72bits with ECC, 2133MT/s
- Simulation between PCIe block and BRAM passed
- Nariman test PCIe Gen3 to DDR4 on hardware and works

Testing: PCIe Gen3 to DDR4

- Write bandwidth tester with Vivado HLS
- The test module measure reading or writing bandwidth by calculating total number of cycles during valid period of Ready&Valid on AXI bus.
- Reset_counter signal is used to output the counter value to s_axilite register to be read by Microblaze and restart the counters.
- Reset_counter signal is connected to timer's interrupt and the interrupt is disabled by MiroBlaze after reading the counter register.
- Write c code on Miroblaze to initial timer, timer ISR code to read counters from tester and disable reset_counter signal.



HLS interesting aspects

- Vivado HLS data flow model has a bug after 2015.1
- Use struct for stream interface
    - Defining “last”, “keep”, and “dest” signal in struct
    - Automatic mapping to signals of stream protocol
- Use void pointer to convert from int to float without casting
    - We couldn’t use union. Not supported by HLS.
- Used a class in min-sum decoder to implement a *circular shifted memory*

HLS Usage

- Network
  - Layer 3: IP
  - Layer 4: Transport
  - Layer 5: Applications
- PCIe
  - Bandwidth measurement
- Monitoring
  - Minitor
- Applications
  - Network test
  - Min Sum Decoder

HLS help

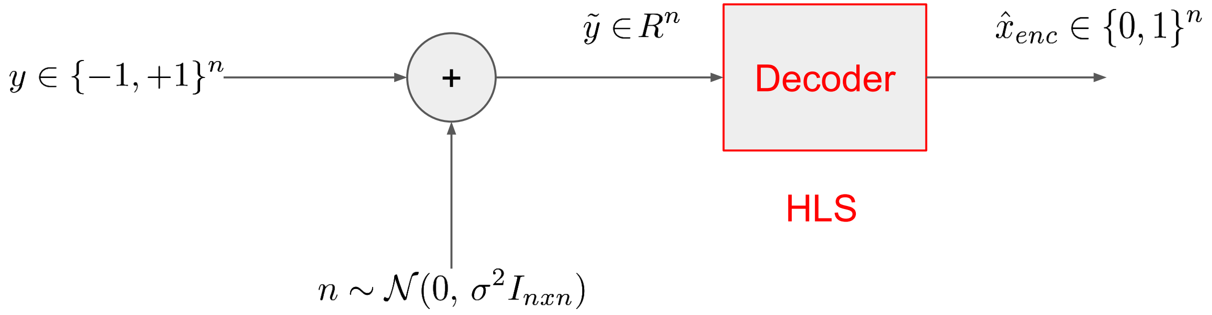
- High level applications are far more easier to implement using HLS (Network)
- Pragmas could help to explore design space in a very short amount of time!
- Faking high level inputs could be done with HLS cores.
- User friendly especially for debugging purposes.
- Floating Point in FPGA!
- Utilizing FPGA resources automatically!

HLS vs HDL time

- Implementing monitoring is better in HDL because it provides more low level access
- Min-sum is a simple decoding algorithm. Writing it in HDL would have taken maybe twice the time.

Min-sum decoder

- What is the min-sum decoder?
  - Graphical inference algorithm used for decoding Low-Density Parity-Check (LDPC) codes
  - Uses the **real-valued noisy** output of the channel to update the **likelihood** of a transmitted codeword
- Why is it popular?
  - Hardware friendly
  - Easy to implement
  - Achieves near Shannon channel capacity

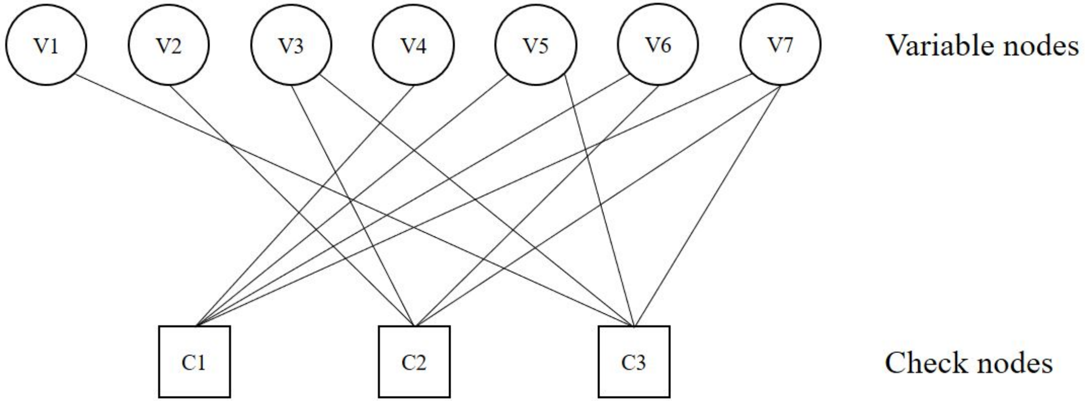


$y \in \{-1, +1\}^n$
+
$\tilde{y} \in R^n$
Decoder
$\hat{x}_{enc} \in \{0, 1\}^n$
HLS
$n \sim \mathcal{N}(0,\, \sigma^2 I_{nxn})$

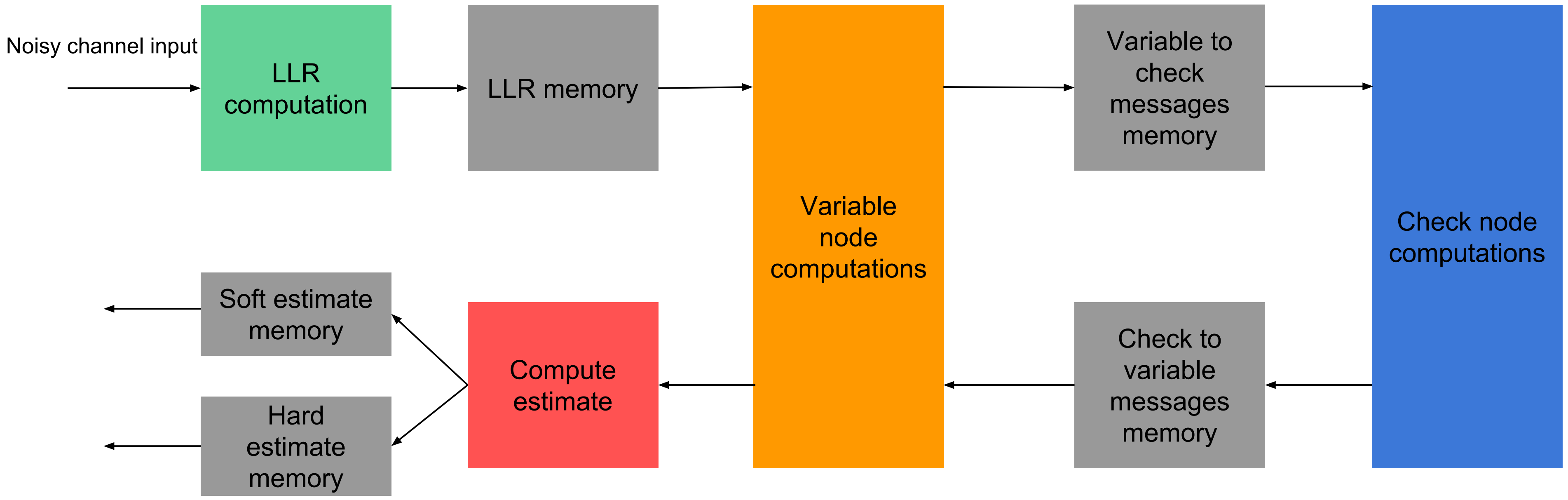
Tanner graph

- A binary linear block code can be specified by a parity-check matrix **H**
- The **H** matrix, in turn, has a graphical representation
  - Each column represents a **variable node** and each row a **check node**
  - A “1” in **H** represents an edge between a variable and check node

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

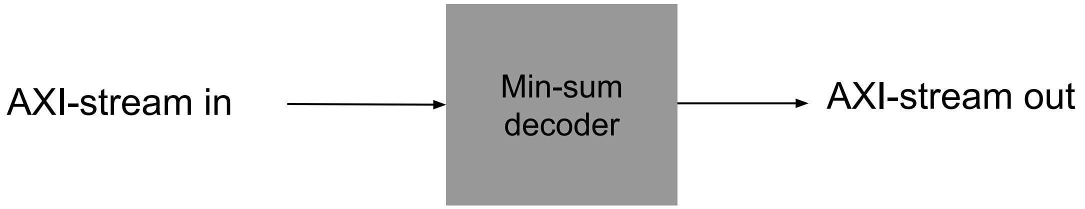


Min-sum hardware block diagram

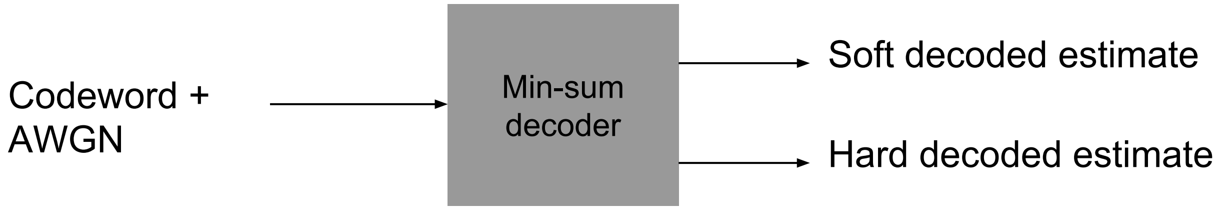


Decoder hardware interface

- AXI-stream for UDP



- AXI-full for PCIe


Codeword + AWGN
Min-sum decoder
Soft decoded estimate
Hard decoded estimate