Virtual Vaccine Passport Scanner for Remote Entry Approval

ECE 532 Group 3:

Eduardo Stecca Ortenblad

Guoxian Wu

Xuening Dong

Mustafa Kanchwala

Project Overview

- What's out there:
  - A Human scans the vaccine passport and allows or denies entry
- Why that’s bad:
  - Risk of Exposure – Very High!
  - Conflicts arising from enforcement – High!
- What can be done:
  - A Hardware based - remote server-controlled scanning & verification

MK

Initial Goal v/s Final Implementation

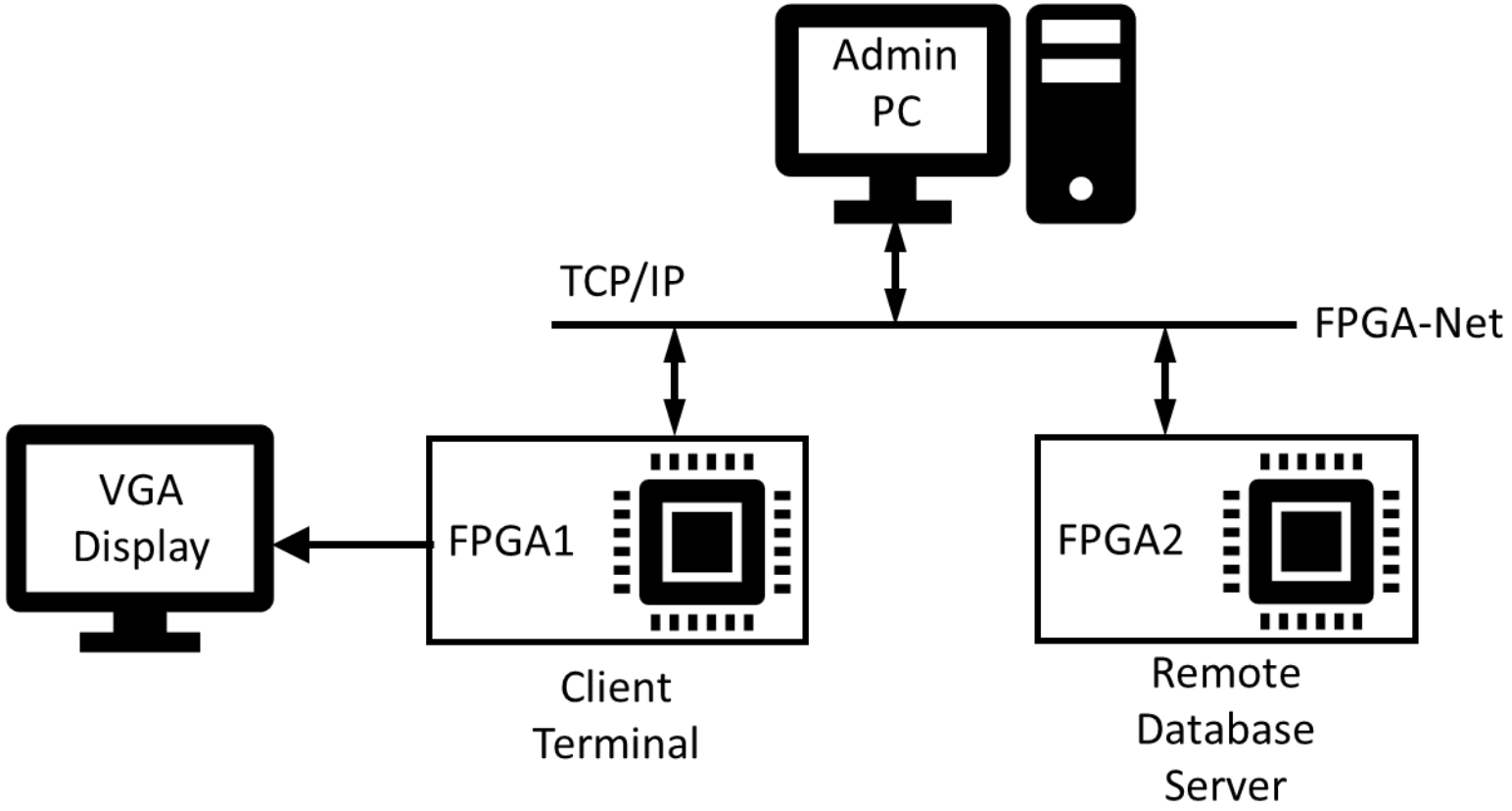
Initial Goal

- *“The aim of our project is to implement a digital, cloud-based database system, combined with a QR code decoder to mimic and try to improve upon the current vaccination passport system, being widely adopted throughout Canada and the world.”* – as stated in the project proposal

Final Implementation

- We met all aspects our initial proposed Goal with the exception of using a DataMatrix instead of QR code!

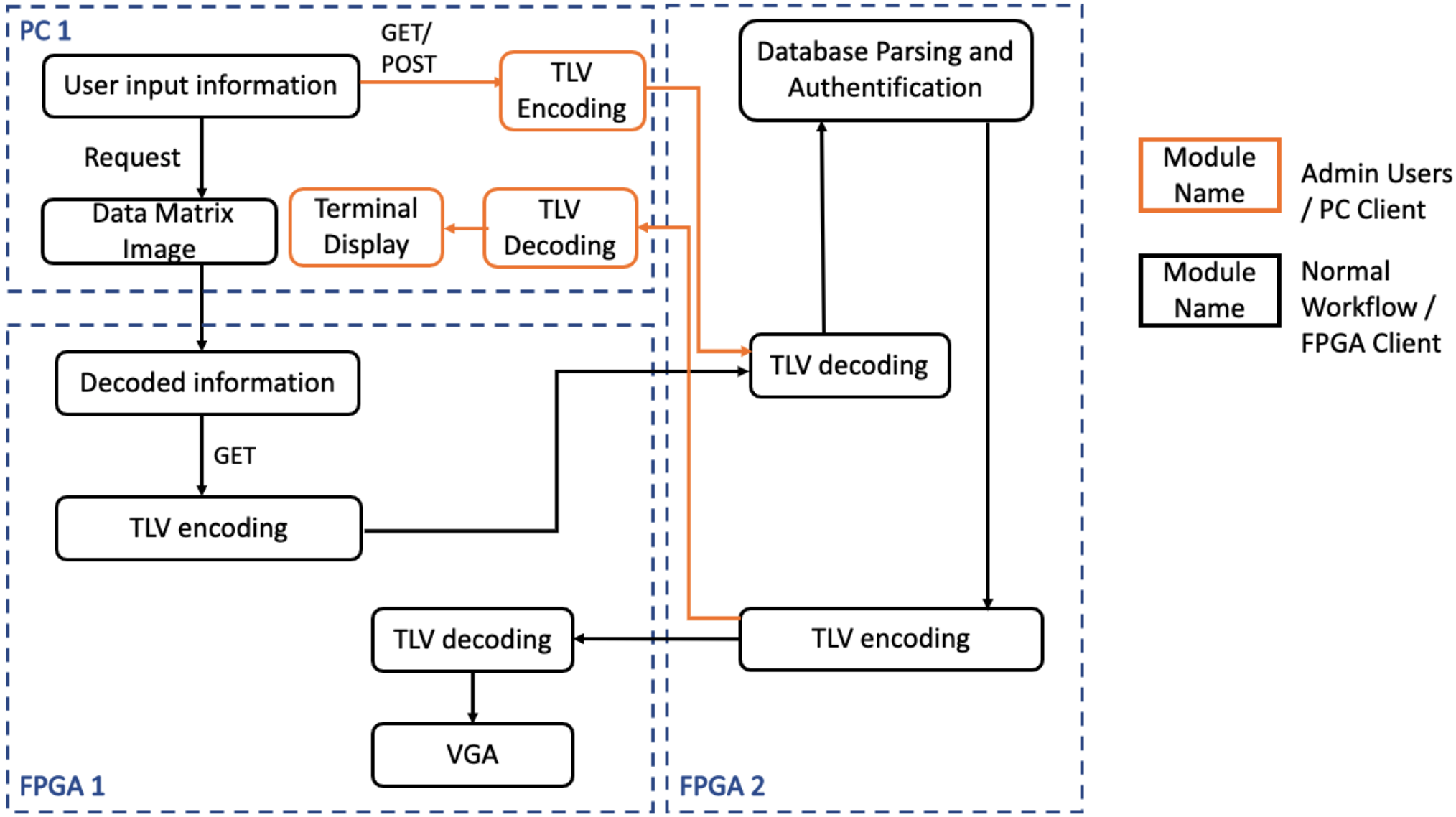
System Connection Diagram



- A standalone FPGA based Remote Database Server
- VGA Display connected to Client Terminal
- A single Admin access PC on the network
- All nodes connected to the FPGA-Net

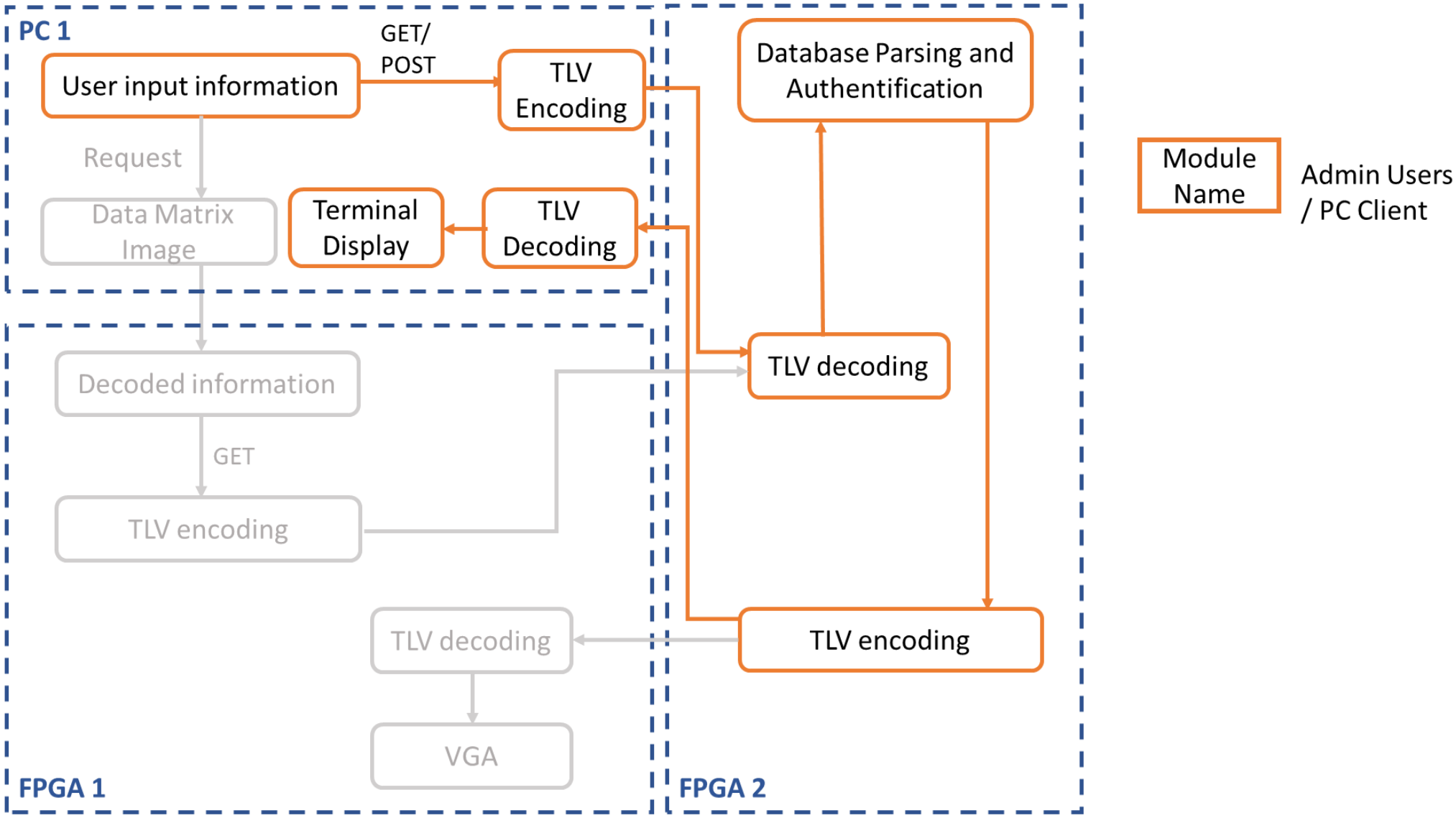
XD

System Overview - Overall



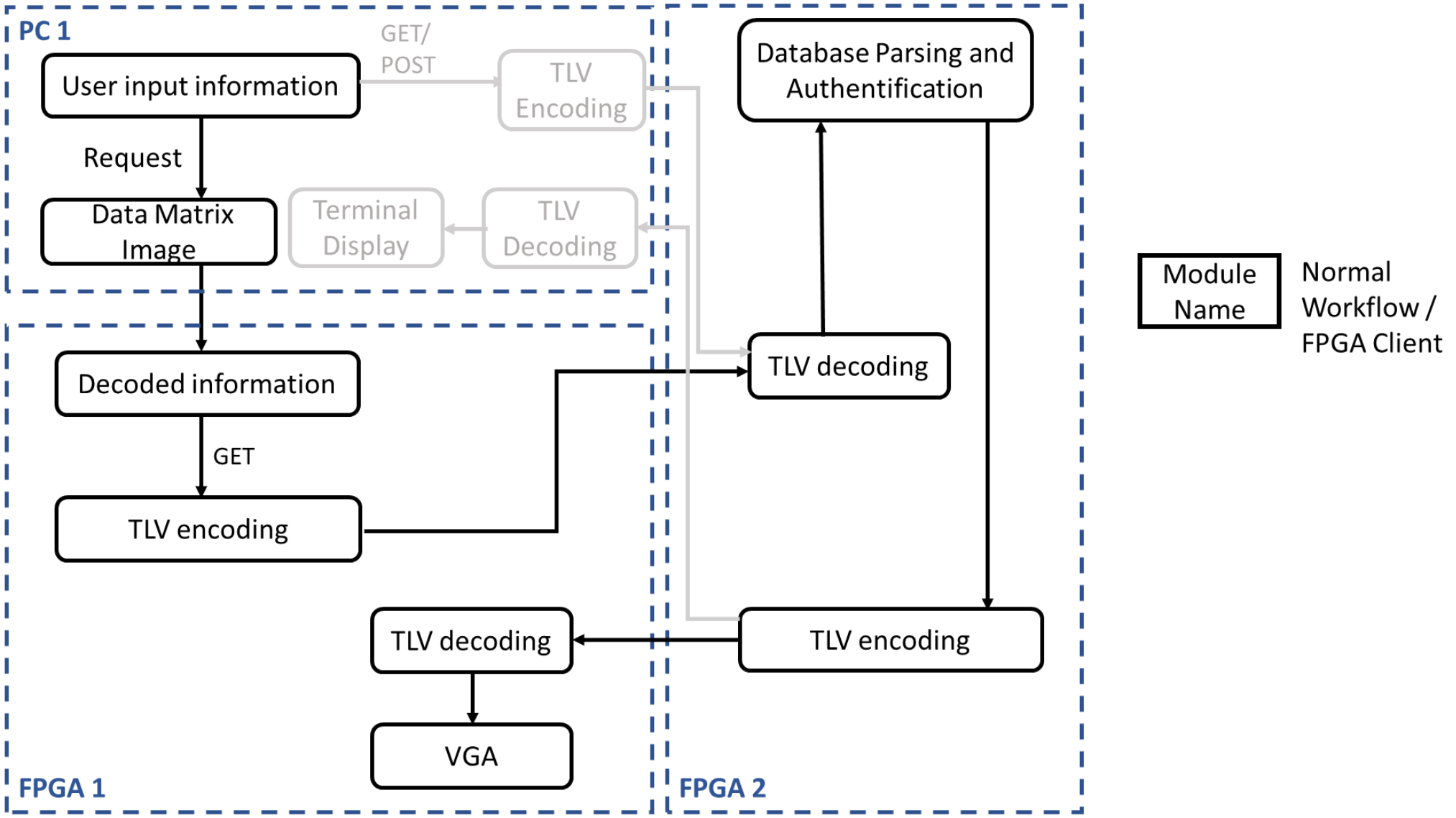
XD

Admin Mode Operation



XD

Normal Mode Operation



XD

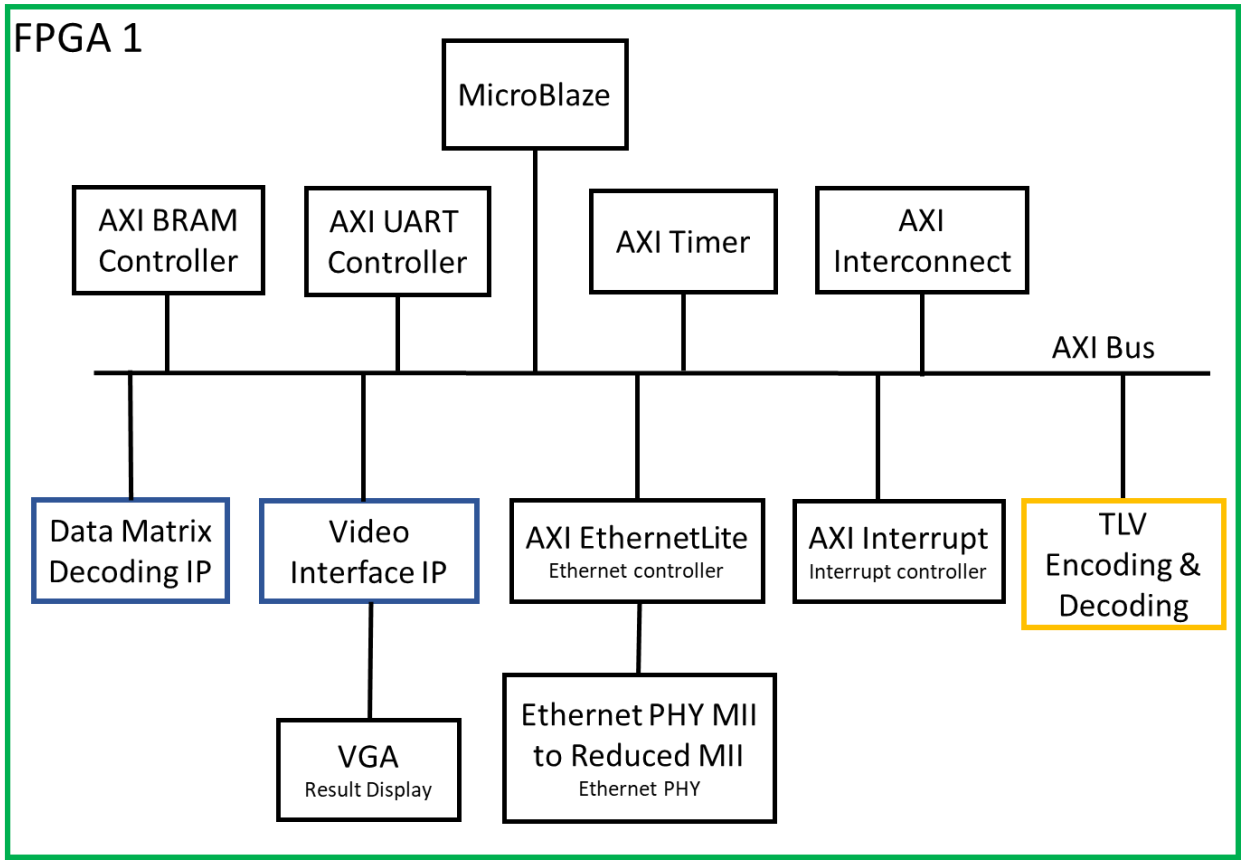
Proposed Block Diagram

Module Name — Custom IP

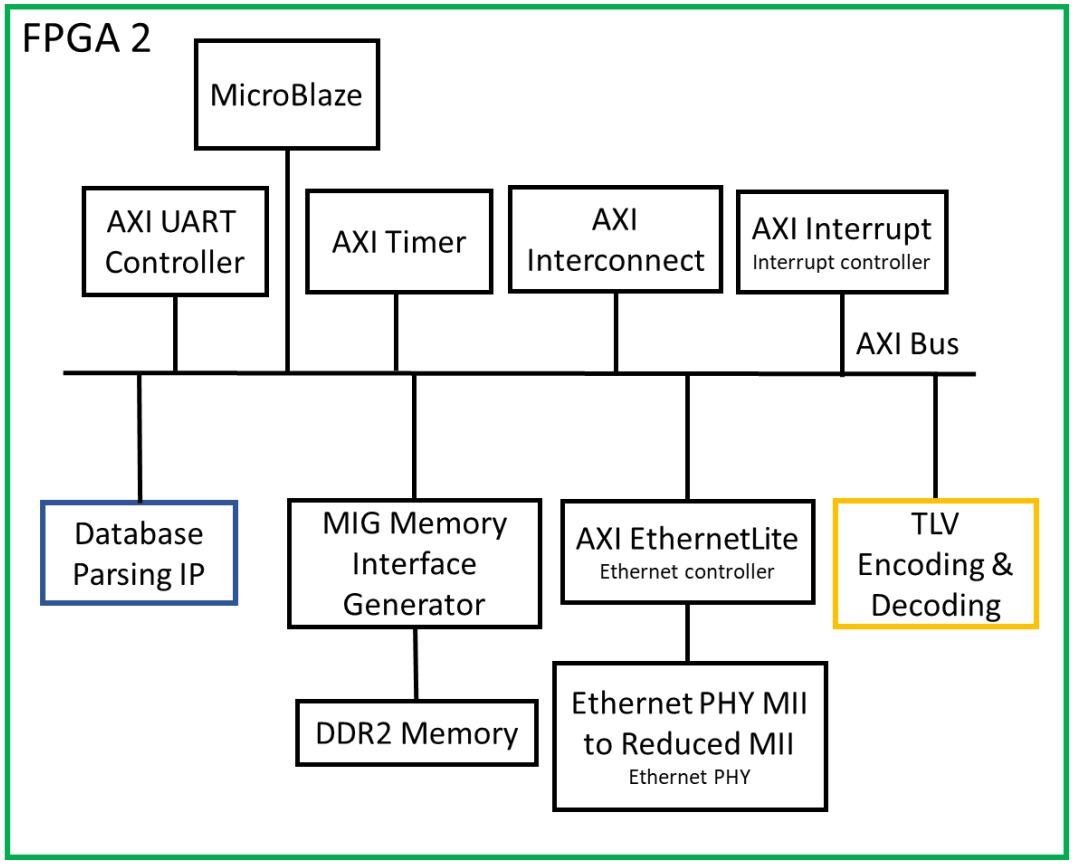
Module Name — Tentative IP

Module Name — Existing IP

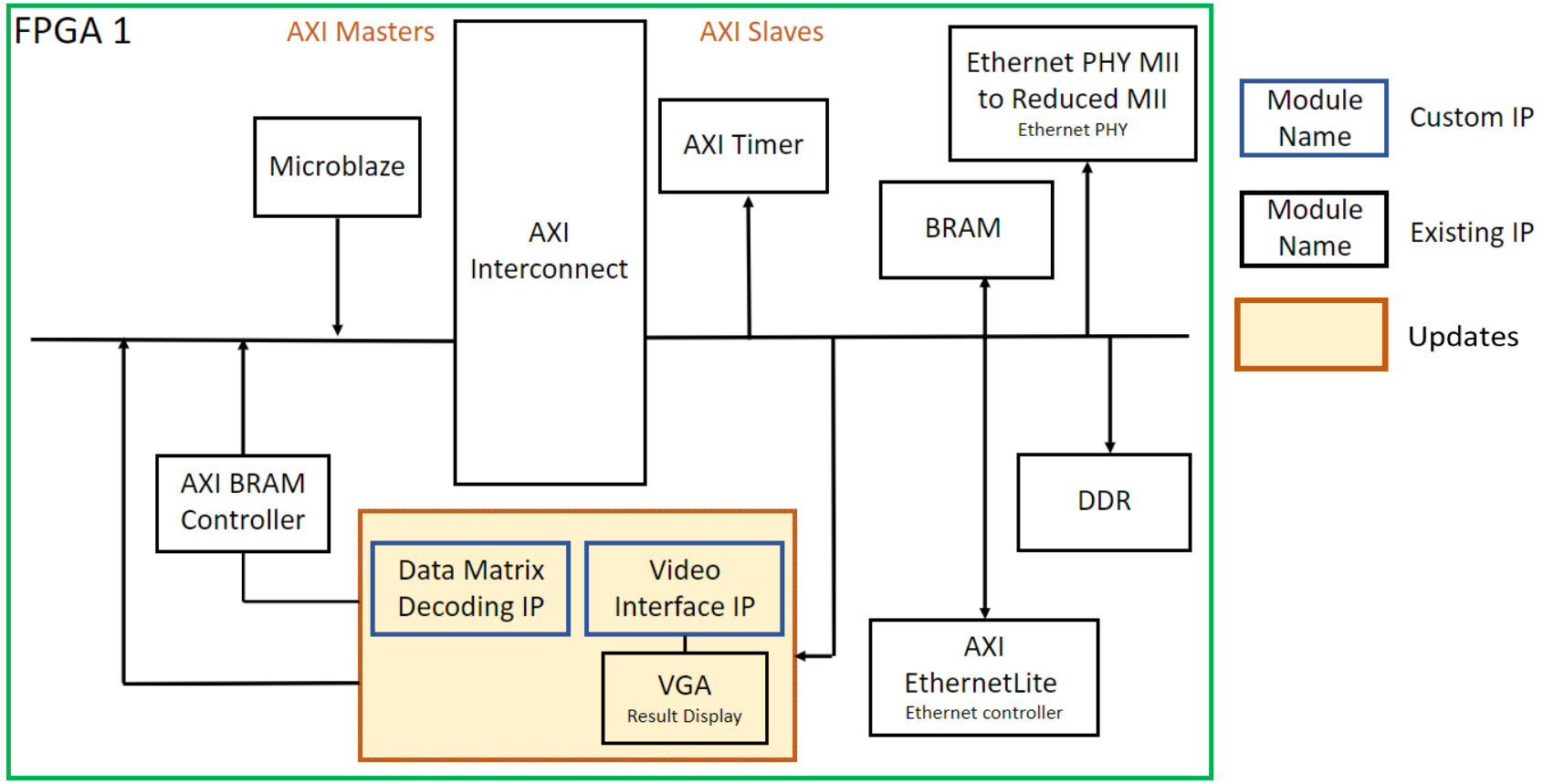
Client



Server

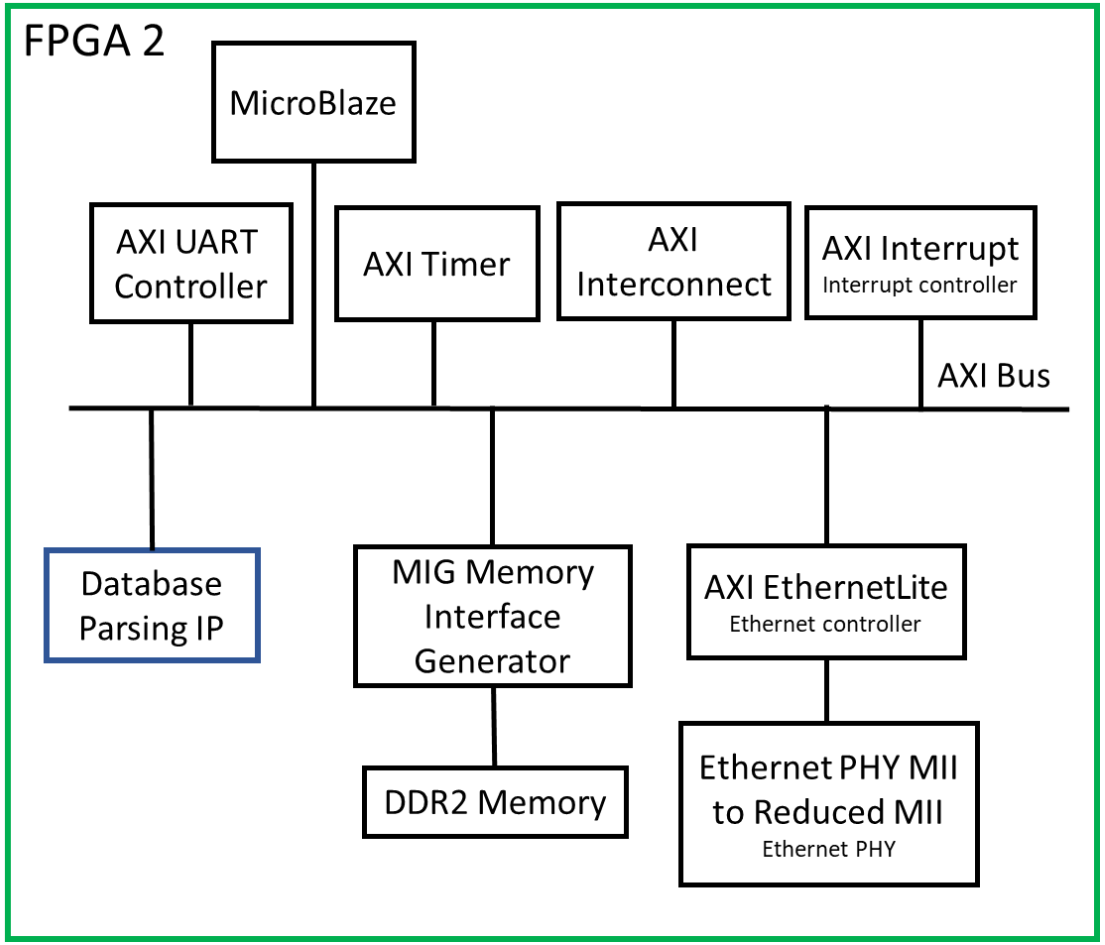


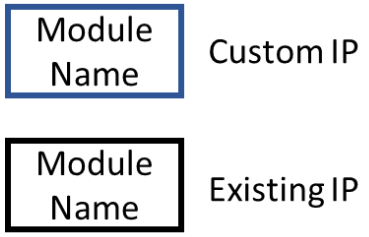
Final Block Diagram (Client)



- Reason for the updates:
  - TLV block removed as encoding and decoding is performed in Software
  - Datamatrix decoder and VGA are integrated because they are controlled by the same FSM and enable signals.

Final Block Diagram (Server)





- Reason for the updates:
  - TLV block removed as encoding and decoding is performed in Software

EO

Resources Used – FPGA 1 (Client)

Custom IP / Software Code

- Custom IP created:
  - Data Matrix Decoding IP
  - VGA Display IP

- Software Code:
  - TLV request generation
  - TLV response parsing
  - TCP server to receive data matrix
  - TCP client to request data from FPGA 2 (Server)

Existing Library used

- EasyTLV – C Library for TLV encoding/decoding to/from raw bytes

EO

Resources Used – FPGA 2 (Server)

Custom IP / Software Code

- Custom IP created:
  - Database Parsing IP

- Software Code:
  - TLV request parsing
  - TLV response generation
  - TCP server to receive database request
  - Request handling logic and AXI interaction with Database IP

Existing Library used

- EasyTLV – C Library for TLV encoding/decoding to/from raw bytes

EO

Resources Used – PC Python Script

Software Code

- TLV request generation
- TLV response parsing
- Data matrix encoding
- CSV parsing
- Update entry logic
- User interface

Existing Library used

- Pystrich – Datamatrix generation
- uttlv – Python library for TLV encoding/decoding to/from raw bytes
- Python's csv library for csv parsing
- Pillow – read images as data matrix input
- Numpy - matrix manipulation

Data Matrix Decoder IP

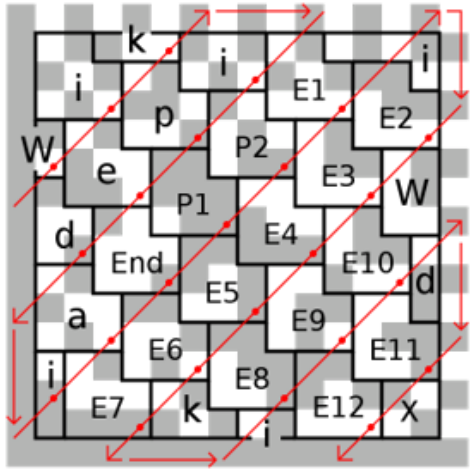
Software

Hardware Custom IP

Bitstream of Datamatrix

Name

UID (For database Entry)



Key function:
Locate the order of different characters (red line)
Locate X-Y Cordinate of each bit of a character

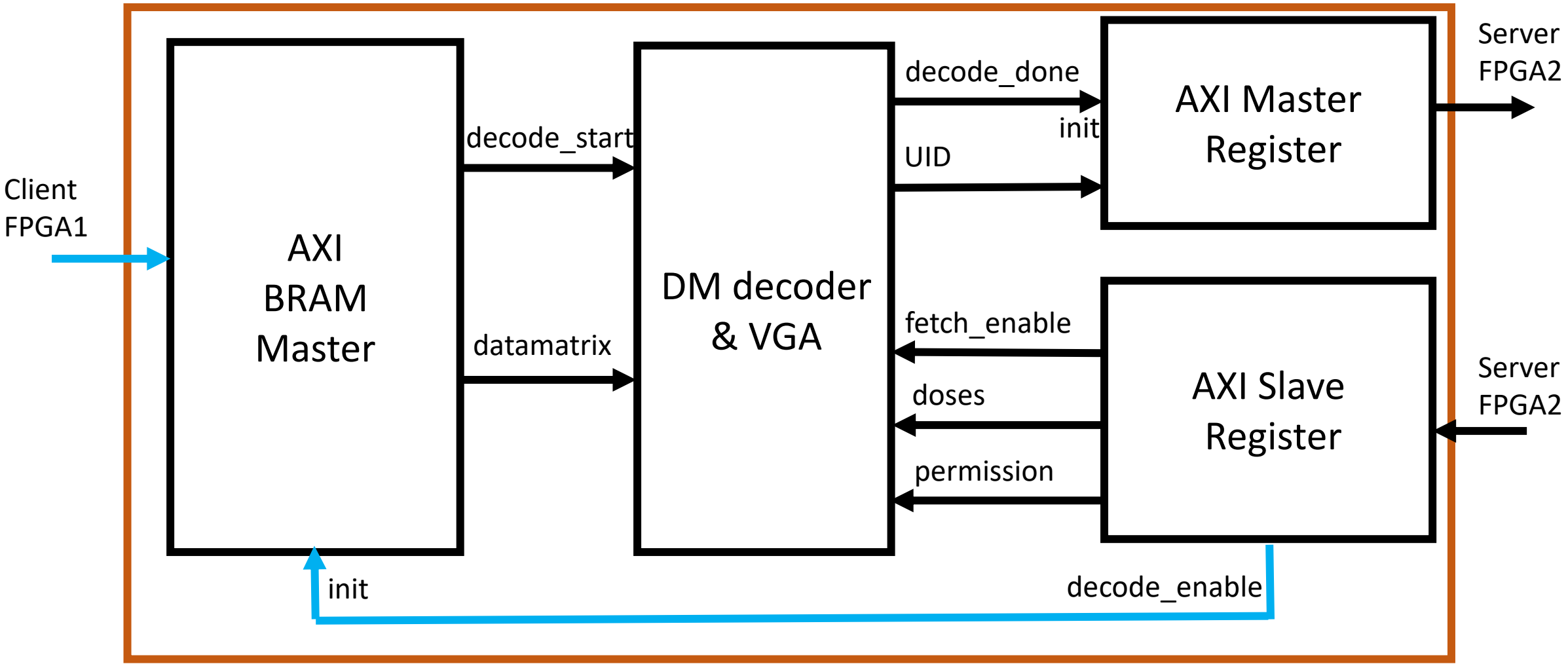
GW

FPGA1 FSM

Trigger signal of a state

Data/Control Signal sent in this state

IDLE state



Hardware block of FPGA1

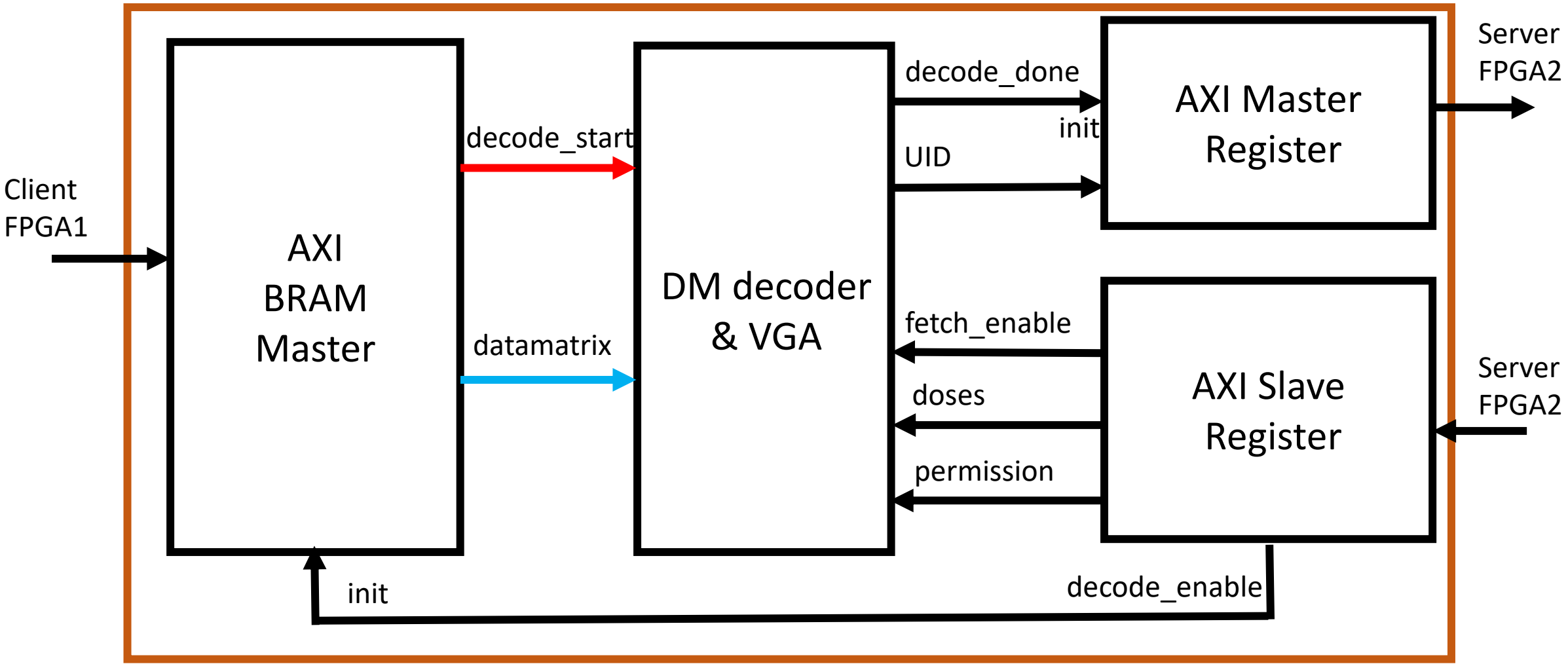
GW

FPGA1 FSM

Trigger signal of a state

Data/Control Signal sent in this state

DECODE state



Hardware block of FPGA1

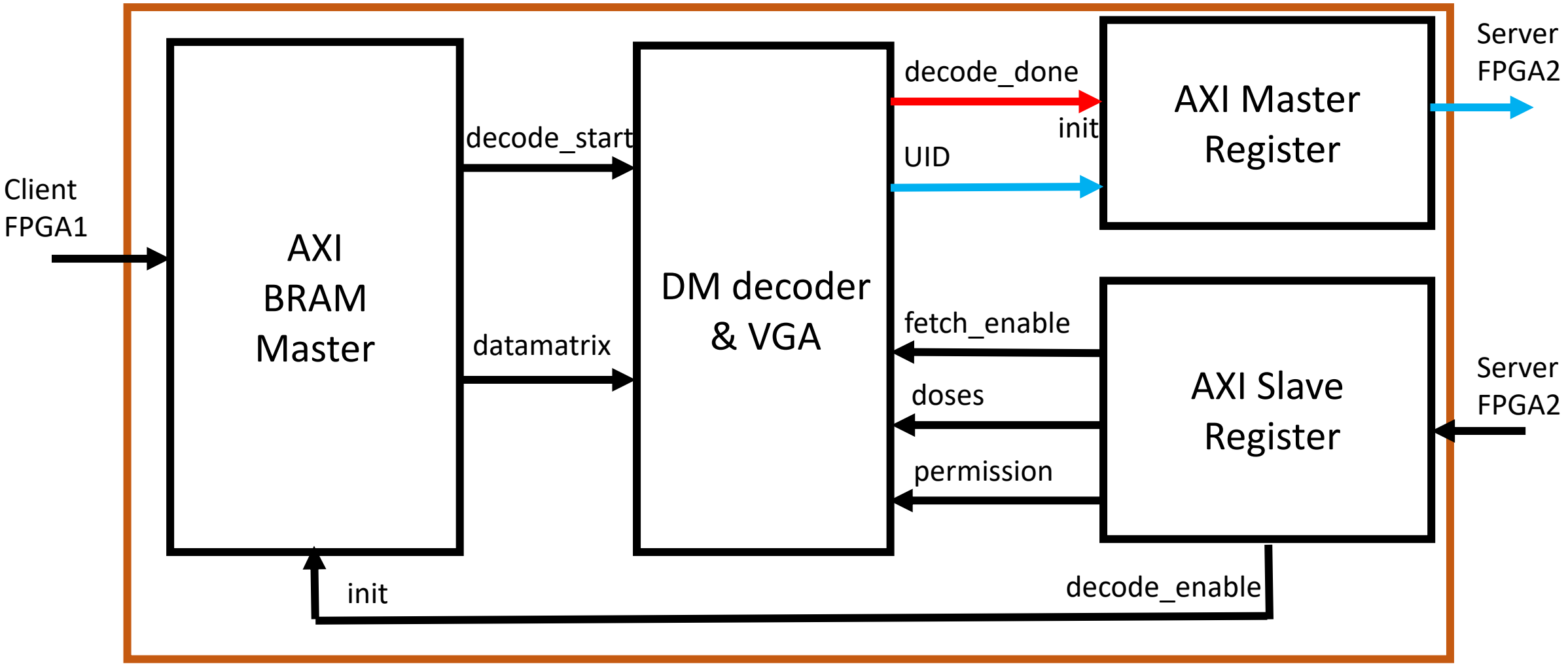
GW

FPGA1 FSM

Trigger signal of a state

Data/Control Signal sent in this state

SEND state



Hardware block of FPGA1

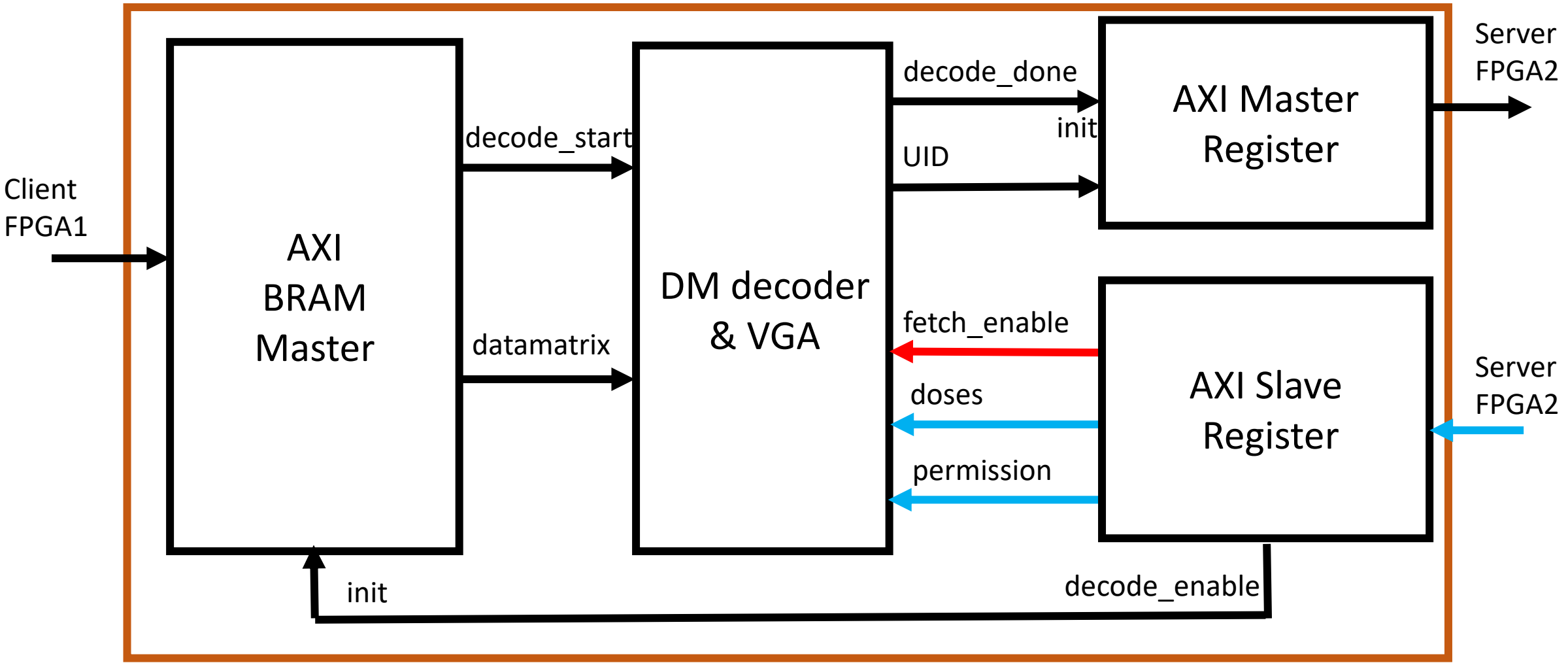
GW

FPGA1 FSM

Trigger signal of a state

Data/Control Signal sent in this state

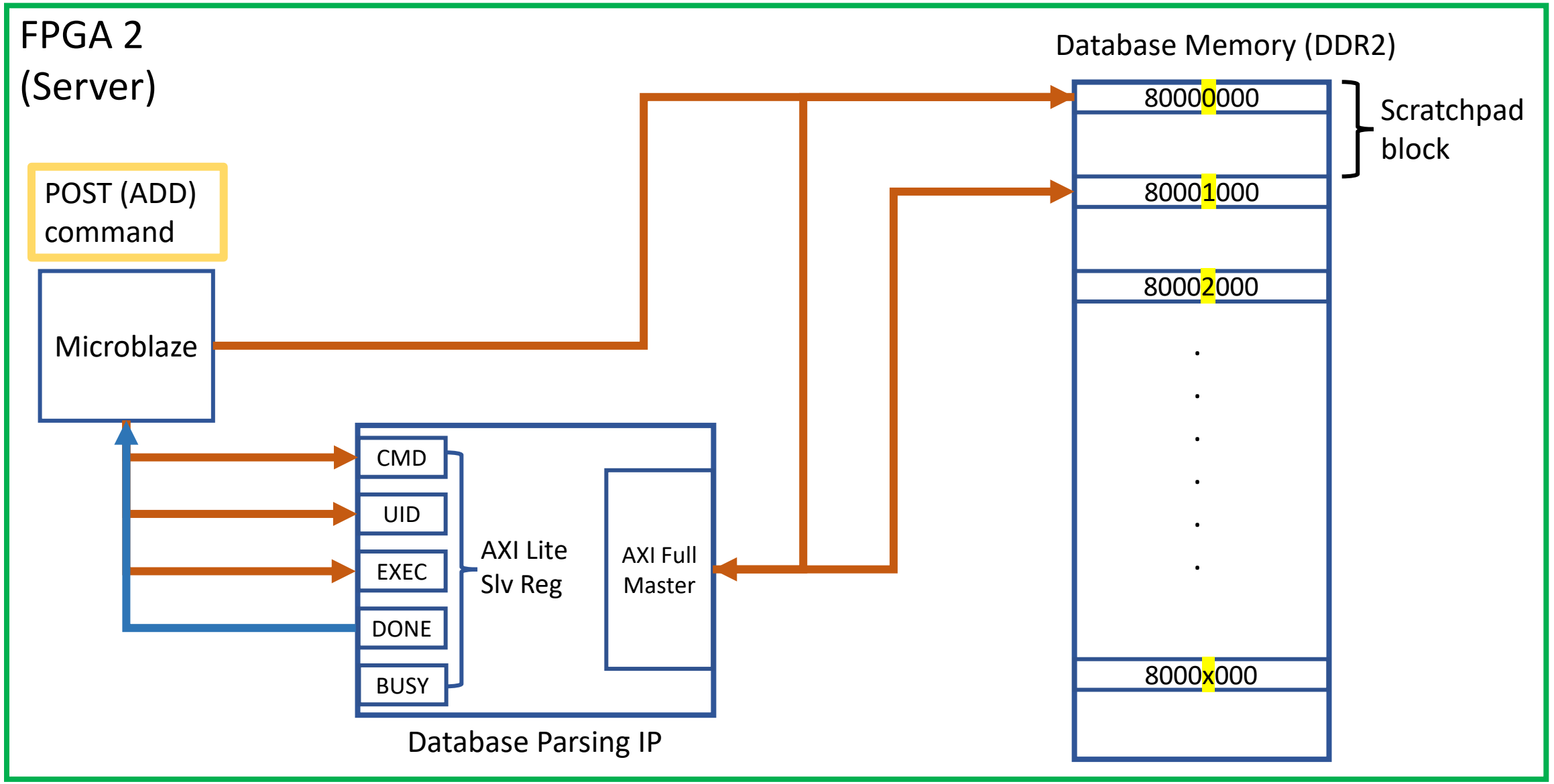
FETCH state



Hardware block of FPGA1

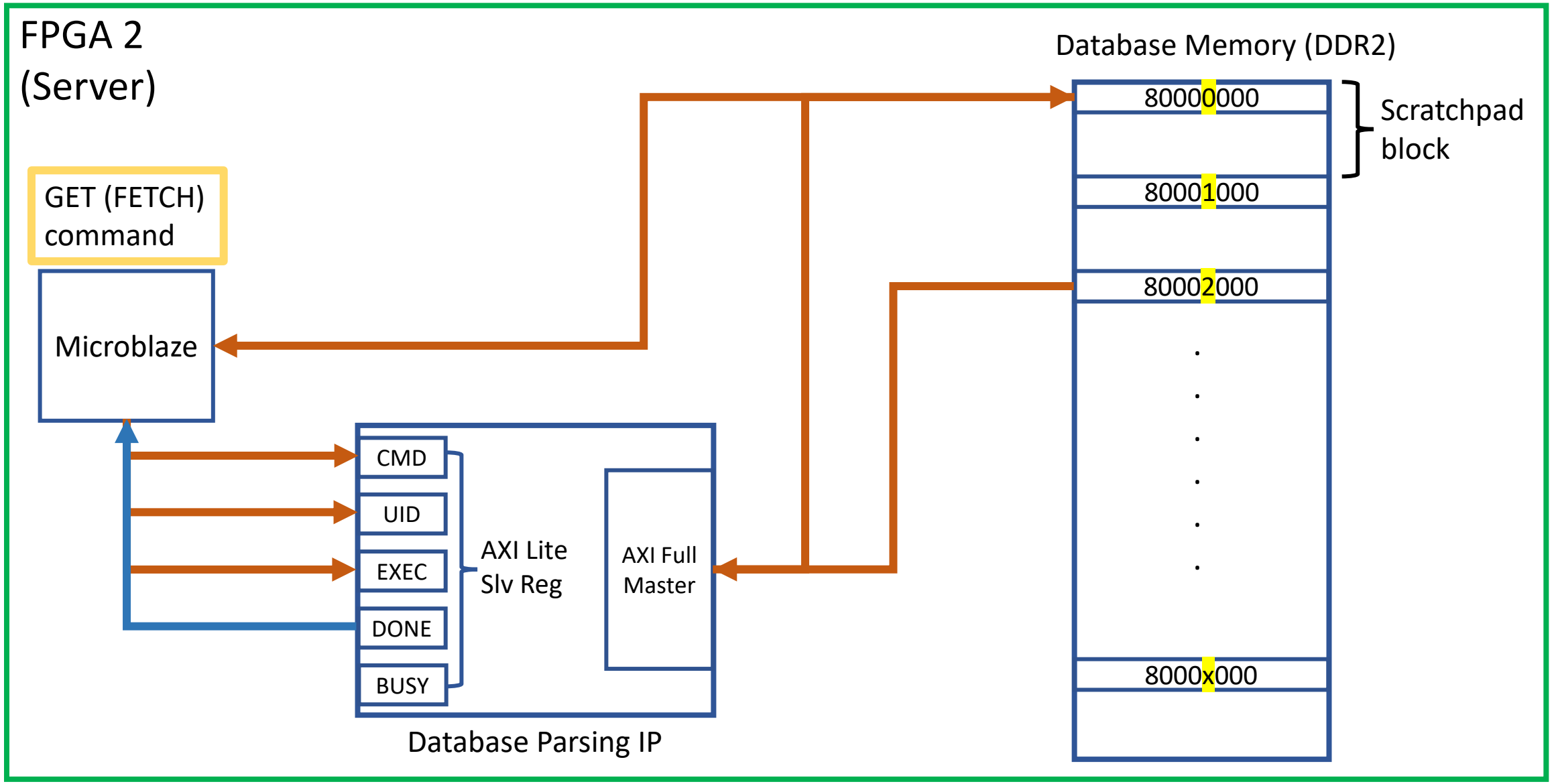
MK

Database IP



MK

Database IP



Challenges faced

- **Problem**: AXI Bus Integration with multiple masters and slaves
  - The auto connect feature is not reliable - messes up the AXI bus clock signals with different frequencies
  - AXI bus freezes if transaction response is not transferred as expected
- **Solution:** Manually connect clock signals and AXI connections. Monitor handshakes and transactions using VIP and ILA to ensure proper operation

- **Problem:** Hardware IP integration with Microblaze program (both server and client sides) needed a lot of debugging
- **Solution:** Performed extensive testing using memory monitors and ILA. Made necessary changes to the IP to resolve the issue

- **Problem:** DDR memory writes did not work when using ethernet
- **Solution:** Identified issues with memory mapping, corrected the linker script and hardware address mapping to resolve the issue

EO

Design Process

- Defined clear specifications and expectations for what our final project should look like before starting implementation
- By the proposal deadline we all had a very clear idea of what had to be done to accomplish our goals
- Facilitated integration of different sub-systems
- Examples:
  - General system workflow was well-defined on project proposal
  - Database entries were designed on project proposal
  - TLV communication protocol was designed in Milestone 1

EO

Our Recipe for Success

- Divided the Project into the “client” side and the “server” side
- Xuening and Guoxian managed the Client
- Eduardo and Mustafa managed the Server
- Helped each other resolve bugs and issues
- Established communication channel on Teams to share files and stay informed of progress or issues faced
- Assigned individual tasks based on each team members skillset and preferences – this ensured strong motivation!

XD

Key take away from the Project

- A fully functioning virtual vaccine passport scanner for entry approval, with the approval displayed on a standalone screen remotely
- Constructed on a database with a maximum capacity of 16384 entries, large enough for normal companies and universities
- Registered 4KB data space per person (UID), allowing extra information to be added
- TLV encoding ensures more information can be seamlessly added to the database and variable size data-transmission

Skills Learned

- Learned about AXI-Lite and AXI-Full and implementing them with custom IPs
- Learned how to effectively map memory in hardware and software
  - Optimize linker script to store data effectively
  - Map memory addresses for fast access
- Explored the use of Microblaze cache memory and optimized it to ensure fast performance
- Developed working experience with Vivado and Xilinx FPGAs
- Learned about how to establish TCP/IP connection within the FPGA-Net and how to run both client and server on the side simultaneously

Demo Time!

Q&A