# High-Level Synthesis Based Acceleration of LLaMA2 Inference on FPGA
# 基於高階合成的LLaMA2於FPGA推論加速設計

組別：62

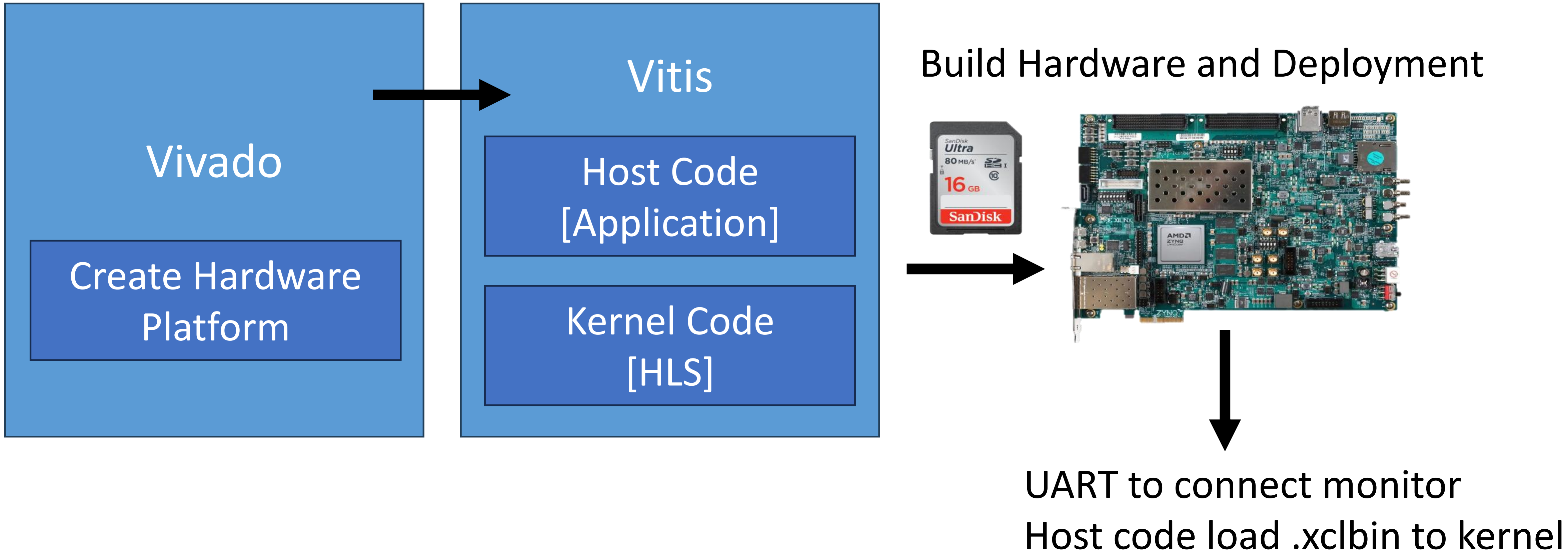
組員：趙堉安、金以凡

指導教授：陳添福 教授

## INTRODUCTION

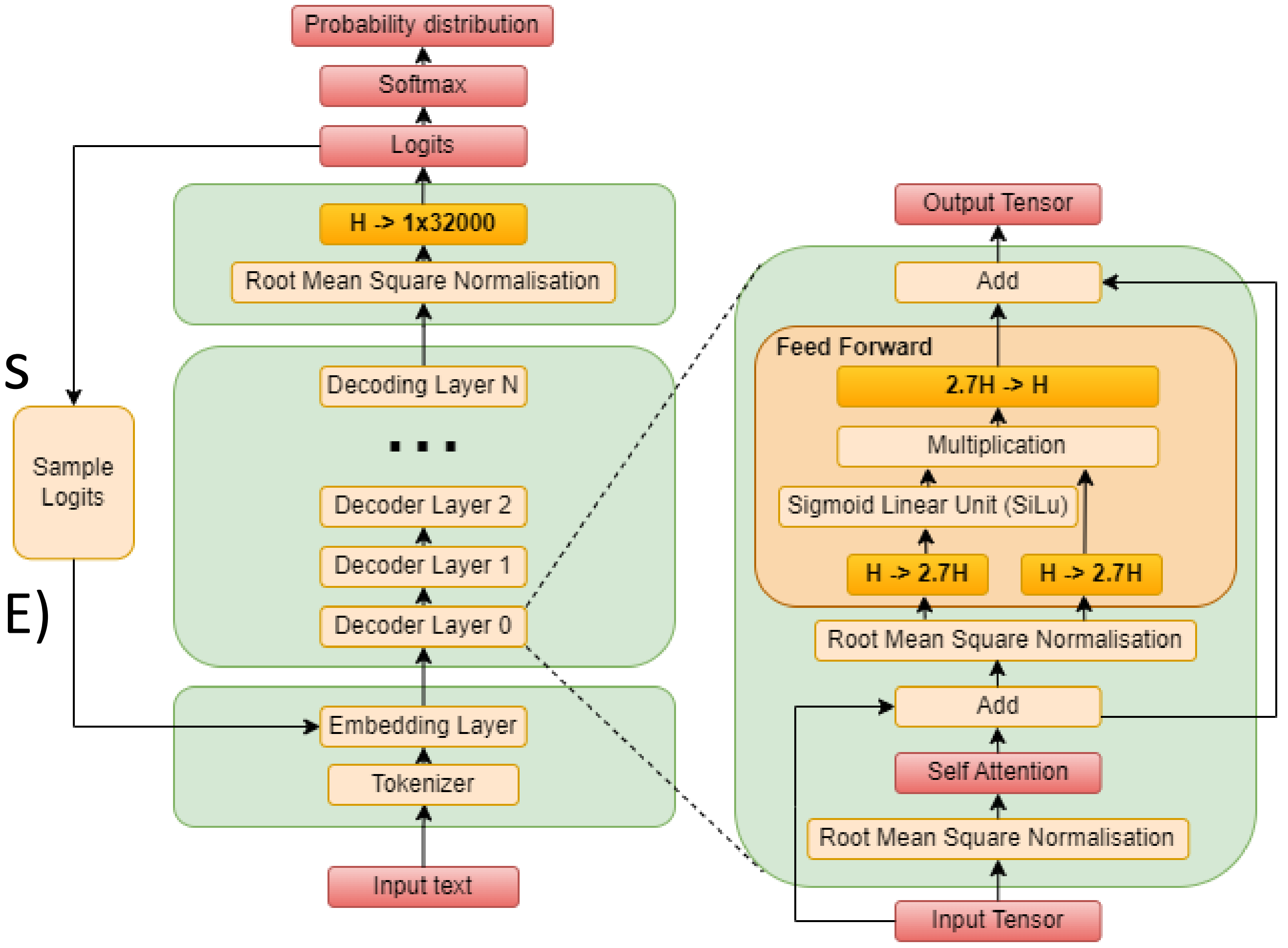
- Background:
  - LLM rely heavily on matrix operations, making them ideal candidates for hardware acceleration
  - While cloud GPUs are powerful, they are not ideal for embedded devices or edge deployment. FPGA gives a better solution for accelerating inference
- Our main goal:
  - To accelerate inference speed by offload operations to FPGA kernels
- We use:
  - LLaMA2.c
  - AMD Zynq UltraScale+ MPSoC ZCU106 board
  - Xilinx Vitis 2024.2
  - High-Level Synthesis (HLS)
  - Xilinx Runtime (XRT) API

## WORKFLOW


Vivado
Create Hardware Platform
Vitis
Host Code [Application]
Kernel Code [HLS]
Build Hardware and Deployment
UART to connect monitor
Host code load .xclbin to kernel

## CONCEPTS

- LLaMA2 model structure
- What we offload is six operations
  - RMS Normalization
  - Matrix Multiplication
  - Rotary Positional Embedding (RoPE)
  - Self Attention
  - Residual
  - SwiGLU (an activation function)



## METHODOLOGY

- We designed and optimized the following key components for FPGA to accelerate inference:
- RMS Normalization Block

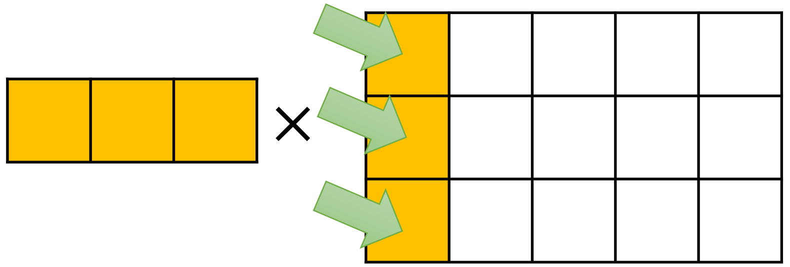
$$RMS(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

  - Optimized approach: Using multiplication by a precomputed inverse (INV_P_DIM) instead of division by P_DIM
  - Replaces a division with a faster multiplication by a constant
  - ***Speed Up 12.7x***
- Matrix Multiplication Block
- Self Attention Block

# METHODOLOGY

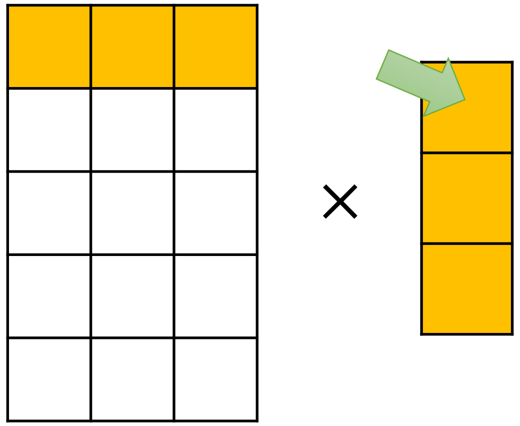
- RMS Normalization Block
- Matrix Multiplication Block
  - For example: x size: n * 1, weight size: n * m

$$o = x^T \times w$$



*Need n*m non-continuous access*

$$o^T = w^T \times x$$



*Need m non-continuous access*

  - Flatten the weight matrix to 1D
  - Mathematically equivalent but reduce hardware access times
  - ***Speed Up 10.6x***
- Self Attention Block

## METHODOLOGY

- RMS Normalization Block
- Matrix Multiplication Block
- Self Attention Block

$$attention\ score = Q \times K^T / \sqrt{d_k}$$

  - Optimized approach: this inverse square root value is precomputed and stored as a constant
  - The expression then becomes a simple multiplication
  - ***Speed Up 21.6x***

## METHODOLOGY

We applied several optimizations for blocks:

1. Memory Access Optimization and Preloading
   - Avoid multiple DRAM accesses by preloading data into BRAM
2. Array Partitioning and Unrolling for Parallelism
   - Use pragma HLS ARRAY_PARTITION, HLS UNROLL
   - Combined loop unrolling and array partitioning to maximize parallel access and system performance
3. Pipeline for Throughput
   - Use pragma HLS PIPELINE
   - Achieved fully pipelined execution that processes one input per clock cycle

***Achieved Total Speed Up: 13.4x***

## EXPERIMENTS

- Model we use: TinyStories 15M parameters
- Generate Stories: Once upon a time, there was a little boy named Timmy. Timmy was very brave and liked to climb trees. One day, Timmy saw a big wagon in the park. He wanted to climb on it, but he was scared …
- Result – Comparison of Inference Speed:

| Model | Tokens | Time Spent (s) | Speed (toks/s) |
|---|---|---|---|
| APU | 201 | 114.75 | 1.75 |
| **Logic Cells** | 225 | 25.83 | **8.71** |

***-> Improved performance by around 5x***

## CONCLUSIONS

- FPGA Hardware Limitations
  - Limited DDR memory capacity restricts the deployment of larger LLaMA2 models
  - Resource constraints (e.g., BRAM, DSP) affect scalability when increasing layer depth or hidden dimensions

- We successfully accelerated LLaMA2 inference speed by 5 times using FPGA-based hardware optimizations
- Provided a Demonstration of the feasibility of Transformer inference on resource-constrained edge devices