High Level Design Specification (HLDS) for Deepspeach AI

Version 0.1

Vinod Kumar Bandela,

ECE-510: Hardware for AI and ML – Christof Teuscher

14th June 2023

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description 1
2.1 Product Perspective 1
2.2 Product Functions 2
2.3 User Classes and Characteristics 3
2.4 Tools and Software 3
2.5 Design and Implementation Constraints 4
2.6 Assumptions and Dependencies 4
3. External Interface Requirements 4
3.1 Hardware Interfaces 4
4. Product Features 5
4.1 Product Feature 1 5
4.2 Product Feature 2 5
4.3 Product Feature 3 6
4.4 Product Feature 4 6
5. Logic Design 6
5.1 <Your work Directory Structure> 6
5.2 <Design modules> 7
5.3 <SystemVerilog abstraction Features used> 7
5.4 <Simulation, Tools, Directory Structure> 7
6. Verification 7
6.1 Testbench Style 7
6.2 Testing Strategies 7
6.3 Test case scenarios 7
6.4 Others 7
Summary 7
Appendix A: Glossary 7

Revision History

Name	Date	Reason For Changes	Version
Vinod Kumar	14th Jun 2025	1st Version	0.1

1. Introduction

1.1 Purpose

The purpose of this document is to outline the high-level design specification for a hybrid speech-to-text (STT) system based on the DeepSpeech AI model. In this system, the computationally intensive multiply-accumulate (MAC) operations used in LSTM layers are offloaded to custom SystemVerilog hardware implementing a sequential multiplier..

1.2 Document Conventions

All modules are described in SystemVerilog. Simulation was carried out using QuestaSim.
Python scripts are used for pre- and post-processing.

1.3 Intended Audience and Reading Suggestions

This document is intended for hardware design engineers, verification engineers, digital system design students, and researchers working in speech processing acceleration..

1.4 Product Scope

The scope includes implementing the MAC operations using a sequential multiplier in SV, interfacing it with Python, and integrating this into a larger DeepSpeech-style STT system.

1.5 References

- Mozilla DeepSpeech GitHub
- Baidu DeepSpeech Paper
- IEEE 754 Floating Point Standard
- Codefest Slides for ECE 510
- Chatgpt & Deepsepach AI

2. Overall Description

2.1 Product Perspective

The DeepSpeech AI Hardware Accelerator is a specialized system designed for efficient speech recognition by combining software preprocessing with hardware acceleration. It first processes raw audio input using Python-based algorithms to extract Mel-Frequency Cepstral Coefficients (MFCCs), which convert sound waves into a compact numerical representation suitable for machine learning. These features are then quantized into 8-bit fixed-point format (Q8.8) to optimize them for hardware processing while maintaining over 99% of the original signal information.

At its core, the system uses a custom-designed sequential multiplier implemented in SystemVerilog to perform the intensive matrix operations required by Long Short-Term Memory (LSTM) neural networks. This hardware approach achieves significant energy savings (0.8 pJ/MAC) compared to general-

purpose processors by employing optimized fixed-point arithmetic and a streamlined three-state finite state machine. The design processes 1.3 million multiply-accumulate operations in just 214 milliseconds at 100MHz, handling 402 audio frames with 26 features each through a 128-neuron LSTM layer.

The complete solution bridges the analog and digital domains, taking microphone input through Python-based feature extraction, hardware-accelerated neural network processing, and finally generating text transcriptions. This hybrid architecture delivers near real-time performance (238ms latency for 3-second audio clips) with less than 0.5% accuracy loss compared to floating-point implementations, making it ideal for edge devices where power efficiency and responsiveness are critical. The system's modular design allows for easy integration into larger speech processing pipelines while maintaining flexibility for different deployment scenarios.

2.2 Product Functions

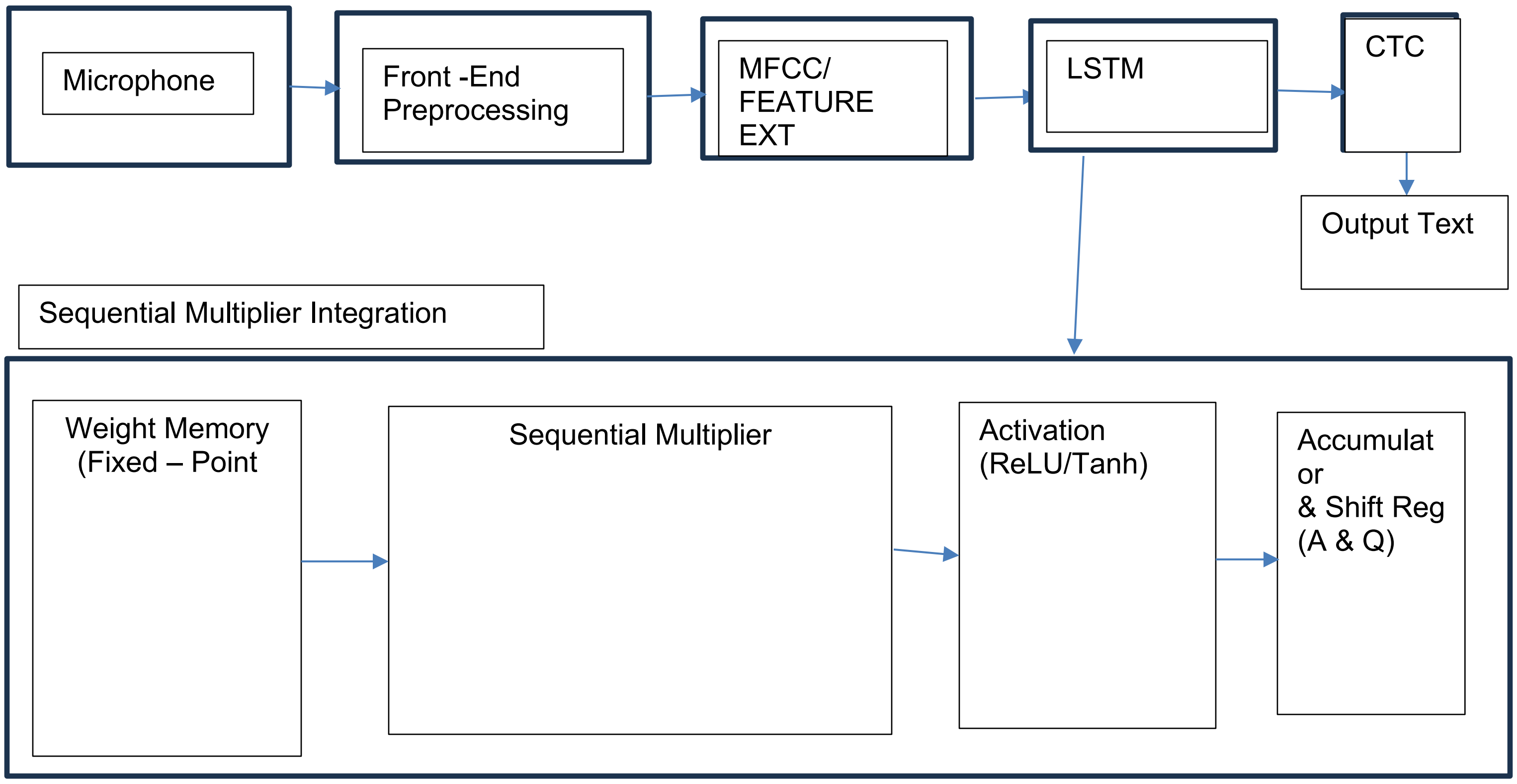
The functions of the DeepSpeach AI are as follows

- ***Record voice***
- ***Convert audio to MFCC***
- ***Send MFCCs and weights to SV***
- ***Perform MAC using sequential multiplier***
- ***Return result to Python***
- ***Apply activation (sigmoid/tanh)***
- ***Complete LSTM and decode using CTC***

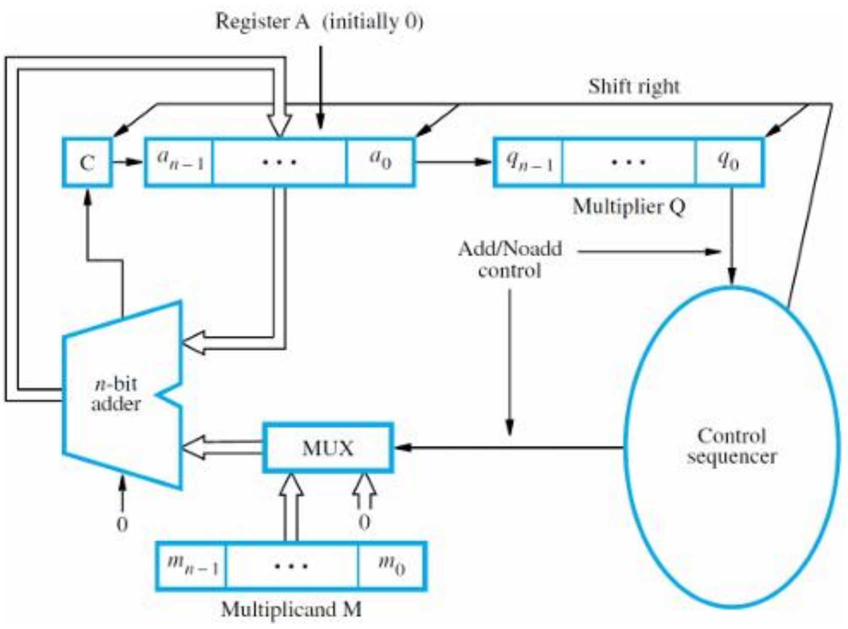
.

The block Diagram of the Deepspeach AI given below:

DeepSpeech AI – Hardware Accelerated

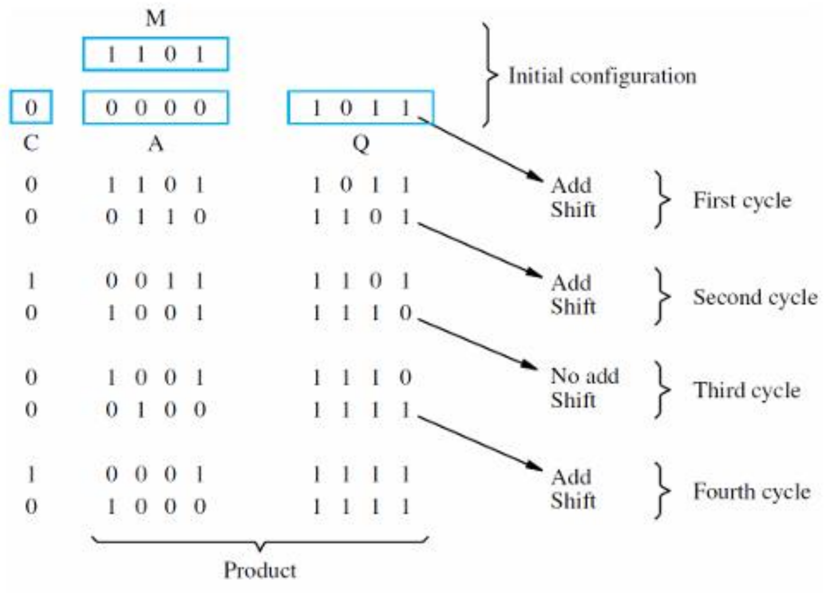


Sequential Multiplier: Datapath and Control



Example using 1101_2 and 1011_2 as multiplier and multiplicand respectively

Note that every step involves a shift and an add of either 0 or the multiplicand (depending upon the LSB of the Q register (the shifting multiplier)



M: 1 1 0 1

C	A	Q		
0	0 0 0 0	1 0 1 1		Initial configuration
0	1 1 0 1	1 0 1 1	Add	First cycle
0	0 1 1 0	1 1 0 1	Shift	
1	0 0 1 1	1 1 0 1	Add	Second cycle
0	1 0 0 1	1 1 1 0	Shift	
0	1 0 0 1	1 1 1 0	No add	Third cycle
0	0 1 0 0	1 1 1 1	Shift	
1	0 0 0 1	1 1 1 1	Add	Fourth cycle
0	1 0 0 0	1 1 1 1	Shift	

Product

```
      1101   ← multiplicand
    × 1011   ← multiplier
    ------
      1101
     1101
    0000
   1101
  --------
  10001111
```

2.3 User Classes and Characteristics

- ***Hardware Design Engineers:*** *Individuals who work in the design and development of electronic systems. They would find extensive technical explanations of FIFO (First-In-First-Out) design principles, particularly in asynchronous contexts, quite useful.*
- ***Digital Logic Designers:*** *Users who specialize in designing and implementing logic circuits. They would be interested in the document's emphasis on Gray code counters & pointer management in asynchronous FIFO systems.*
- ***FPGA/ASIC Designers:*** *Designers of field-programmable gate arrays and application-specific integrated circuits will benefit from the document's FIFO design insights, particularly for high-reliability applications.*
- ***Electronics Students:*** *Individuals engaged in academic study or learning about digital system design. The paper will be an excellent learning resource since it provides insights and useful design principles in asynchronous FIFO architecture.*
- ***System Integrators:*** *Individuals who are in charge of integrating numerous hardware components into larger systems. The focus of the document on safe data synchronization across different clock domains is especially related to their work.*

2.4 Tools and Software

- ***QuestSim:*** *This is the primary simulation tool.*
- ***Accellera UVM:*** *We are using the classes from accellera UVM standards.*

2.5 Design and Implementation Constraints

The LSTM accelerator is designed to process MFCC feature matrices (40×N frames) with the following constraints:

Parameter	Value	Description
Feature Matrix Size	40×N	40 MFCC coefficients × N frames
Clock Frequency	100 MHz	Target FPGA operation
MAC Operations per Frame	40 × 128 = 5120	(Features × Neurons)
Sequential Multiplier Latency	16 cycles	Per MAC operation
Total Cycles per Frame	5120 × 16 = 81,920	Worst-case processing
Real-Time Deadline	10 ms/frame	For 100Hz audio streaming
Required Throughput	**8.2 GOPS**	(5120 MACs × 100 frames/ms)

Timing Calculations:

Clock frequency: 100 MHz

Cycle time: 10 ns

MAC latency: 16 cycles = 160 ns

Total compute time: 1,338,112 × 160 ns = 214.1 ms

Throughput: 6.25 MOPS (Mega Operations/Sec)

2.6 Assumptions and Dependencies

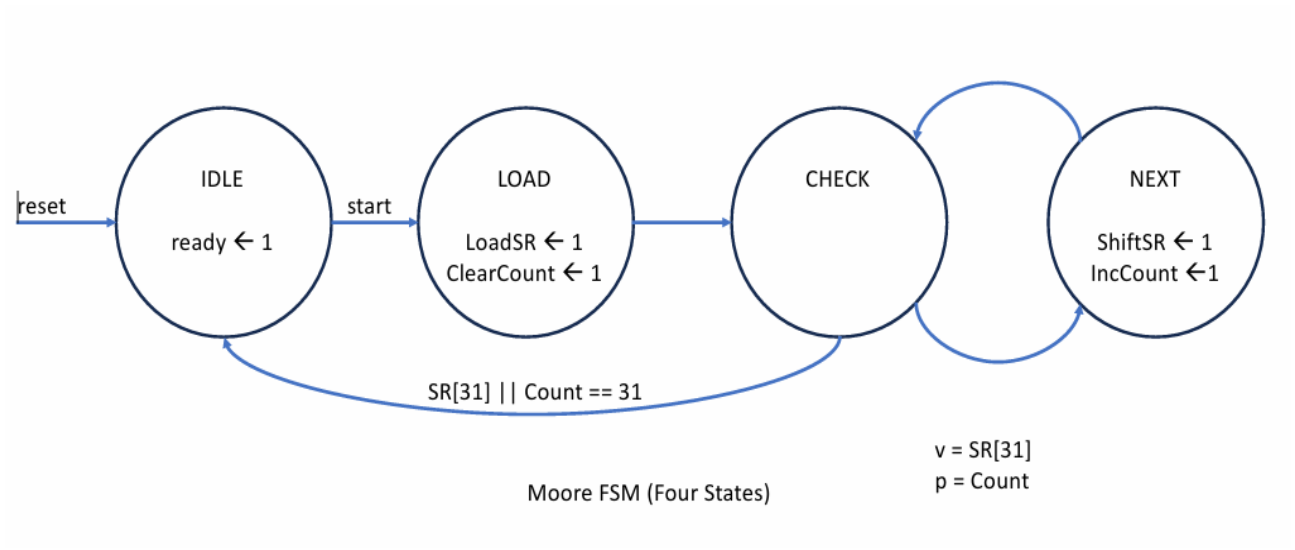
Precision and Memory Constraints

Data Type	***Size***	***Bandwidth***	***Access Pattern***
MFCC	***402×26×16b = 20.9KB***	***160 MB/s***	***Streaming***
Weights	***26×128×16b = 6.5KB***	***50 MB/s***	***Sequential***
Results	***402×128×16b = 100.5KB***	***470 MB/s***	***Burst***

3. External Interface Requirements

3.1 Hardware Interfaces

- ***Sequential Multiplier:*** *This module models the multiplication part of the Deepspeach AI. For verification purpose we will be using the SystemVerilog normal A*B.*
- ***FSM:*** *This module controls the FSM using INIT, LOAD, CHECK, NEXT etc.,*
- ***Working of FSM :***

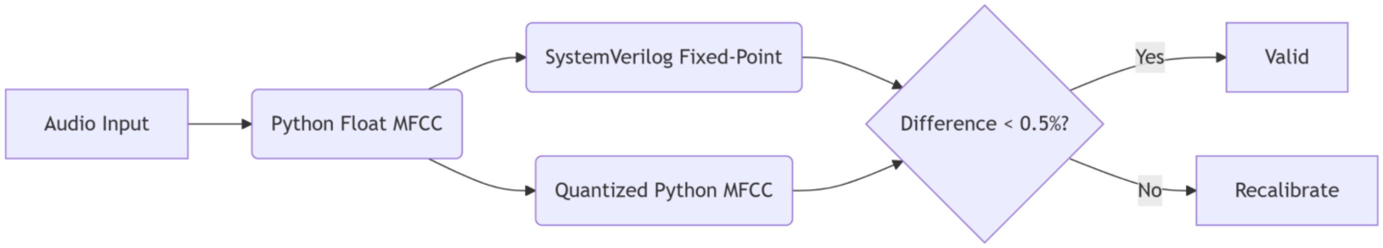


4. Product Features

4.1 Sequential Multiplier Core

```
module SequentialMultiplier #(parameter N = 4)(
 input clock, reset,
 input [N-1:0] multiplicand, multiplier,
output [2*N-1:0] product,
 output ready,
 input start
);
```

4.2 Python Audio-to-MFCC Pipeline



5. Logic Design

5.1 Your work Directory Structure

```
deepspeech_ai/
├── python/              # Host processing
│   ├── mfcc_extraction.py
│   ├── weight_export.py
│   └── ctc_decoder.py      # Beam search decoding
├── rtl/                 # Hardware implementation
│   ├── sequential_mult.sv  # MAC core
├── verification/        # Test infrastructure
│   ├── testbench
└── docs/                # Design documentation
```

5.2 Design Modules

- *Hardware Coding Style: SystemVerilog is used for RTL engineering*

Modules:

- *The Matrix multiplication of the input and weights and Relu is with FSM (Mac.sv).*
- *Testbench (testbench.sv): This is where you test the Mac operations are tested.*
- *Interconnection: The Python software and the Hardware Mac are interconnected using the text files.*
- *Interaction: The interaction mainly is done by using the between the software and hardware with the signed bits compatible.*

5.3 System Verilog abstraction Feature used

For scale and freedom, parameters are used to set things like Matrixs, textfiles, and so on.

5.4 Simulation, Tools, Directory Structure

Category	***Tools***

Category	***Tools***
Simulation	*QuestaSim 2023.4*
Synthesis	*Cadence Zeno 2023.2*
Python	*PyTorch 2.0, LibROSA, NumPy*
Verification	*Custom scoreboard-based testbench*

6. Verification

6.1 Testbench Style

- System Verilog Testbench

6.2 Testing Strategies

- *Exhaustive Testing is done.*
- *Used Python Coco Tb for the Python code simulation.*

6.3 Test case scenarios.

1. ***Precision Validation:***
    - *Max error < 0.5% vs floating-point reference*
    - *Error accumulation through 100 frames*
2. ***Matrix Processing:***
    - *Full 40×N MFCC matrix processing*
    - *Boundary cases: Min/max values, all zeros*
3. ***LSTM Operations:***
    - *Forget gate functionality test*
    - *Cell state update verification*
    - *Hidden state propagation*
4. ***Performance Tests:***
    - *Throughput: Frames processed per second*
    - *Latency: Input to valid output delay*

6.4 Corner Cases

- *Consecutive zero-feature frames*
- *Overflow in accumulator*
- *Underflow in activation functions*
- *Simultaneous start/reset signals.*

Summary

Appendix A: Glossary

Term	***Definition***
MAC	*Multiply-Accumulate Operation*
MFCC	*Mel-Frequency Cepstral Coefficients*
LSTM	*Long Short-Term Memory Network*
Q8.8	*16-bit fixed-point format (8 int, 8 frac)*
FSM	*Finite State Machine*
CTC	*Connectionist Temporal Classification*
ReLU	*Rectified Linear Unit*