AS 031
BreXting: Brain Texting

Team members：

Arna Ghosh Organization: McGill University

Sivakumar Chidambaram Organization: Polytechnique Montreal

Ajin Jiji Tom Organization: McGill University

Instructor: Prof. Jean-Pierre David Organization: Polytechnique Montreal

I. High-level Project Description

We aim to use Electroencephalography (EEG) as it is one of the most elegant neuroimaging techniques. The major challenge lies in using EEG to decode the character the subject is thinking. To simplify the problem, we try to decode a digit thought by the subject from their EEG recording. We use custom EEG headset interfaced with arduino to record EEG over the occipital region while the subjects are looking at visual stimulus of different digits (0-9). The basic workflow includes: -

1. Discriminating event-related desynchronization (ERD) and synchronization (ERS) that occur during the onset and removal of the stimuli respectively from the resting state to mark the period when the subject thinks of one single digit. The raw data is transmitted from the headset's bluetooth module to the receiver connected to an Arduino.

2. From the EEG data recorded between an ERD and ERS, the stimuli is decoded using deep convolutional networks (CNN). The deep CNN will be implemented on the Terasic DE-10 Nano Board`s Cyclone V FPGA.

3. We use Morlet wavelet components corresponding to frequency ranges of 8-30 Hz (alpha and beta bands) at each time point of the EEG recording. From the time-frequency maps, we aim to decode the digit using a deep feedforward network.

This allows for a proof-of-concept for the method and this can be extended to all characters and then words to revolutionize the way people write texts on their devices. This could also allow people with disabilities to text like others, thus bridging the gap through technology. The method is valid for one particular subject for whom the network is trained. To apply the same to another subject, the training procedure has to be carried out once again.

II. Block Diagram

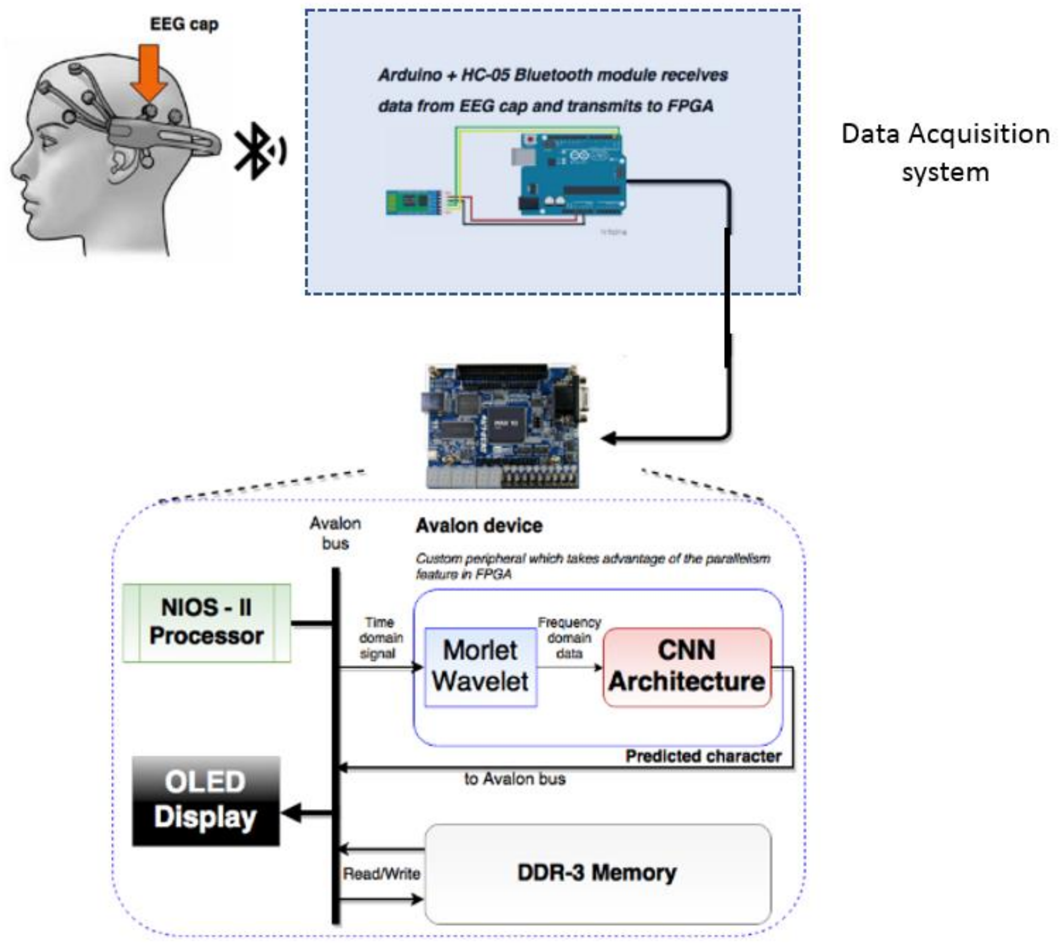


Fig 1: Overall Block Diagram

III. Intel FPGA Virtues in Your Project

The Intel DE-10 Nano board contains various Hardware feature which will aid the development of this project.

NIOS II – softcore processor which controls the overall flow of the system
NIOS II Custom instructions for floating point computations
High speed DDR3 SDRAM to store the data.

The main reasons for using an Embedded system for this project is to make the systems easily movable and real time. We use Intel FPGA because:

1. Cyclone V SoC give an important advantage of Faster clock rates using the

ARM processor and low power and targeted acceleration of the Intel Cyclone V FPGA.

2. The tools provided by Intel such as Quartus, Qsys, Power Estimator allows fast

   development of the product.

3. The IP Designs provided by Intel/Terasic, provide a great starting point to implement the designs.

IV. Functional Description

The major contribution of the design is a basic framework for implementing a Deep learning-based Brain-Computer Interface (BCI) system on FPGA. Present BCIs are mostly used in a research setup or require heavy computing platforms connected for

efficient functioning. This restricts their mobility and hence usability to an average user. In this project we focus on designing a BCI where the subject can type using their EEG signals. Our primary goal was to move the mouse pointer using EEG signals from a subject. Once the subject is able to move the pointer, we plan to relay the output to a visual keyboard where the subject can control the pointer to type using the on-screen keyboard. Since the design is based on FPGA, it is supposed to be mobile and much more usable than current state-of-art machines.

For a proof of concept and to judge the potential of our design, we used one of the datasets from the BNCI Horizon 2020 database, specifically the 4-class motor imagery dataset. We use this dataset because this aligns with our primary goal of mapping the 4 classes to 4 directions of pointer movement (up, down, left and right). Motor imagery is a well-explored paradigm in BCI and therefore it used for the proof of concept. A deep network is trained to classify each segment of EEG data into one of the 4 classes. Deep network allows better classification as compared to the P-300 method and also provides a robust method to classify without prior inspection of EEG data to extract features. Once a deep network is trained, we implement the network using the learnt weights on FPGA and run forward passes of the data. Since the network is trained on multiple trials of the same subject, it is expected to be robust to inter-trial variability and thus perform well on further scans from the same subject.

The combined powers of deep learning and FPGA pave the path for future mobile BCI platforms that would revolutionize the way people go about their daily lives. Our results are promising and indicate the scope of bringing BCIs to mobile platforms. With out current design, we can successfully classify the 4 classes with about 80% class-specific accuracy. Next up, we would be interfacing the output to move mouse pointers and enable typing using an on-screen keyboard. We would also be using the OpenBCI board to collect EEG data from subjects while interacting with this setup and therefore update the network weights to suit that subject. Hence in the upcoming steps, we would design a personalized system for individual subjects that allows them to type using their brain waves, or as like calling it - Brext.

V. Performance metrics

The performance parameter for the application includes:

i) Development Time: The time required to develop a design on an embedded system is an important factor. Creating a soft core NIOS II processor and integrating other components such as Timer, Memory using QSYS reduces the development time drastically. Also, in-built IPs such as floating-point hardware/divider aides in the development of the designs.

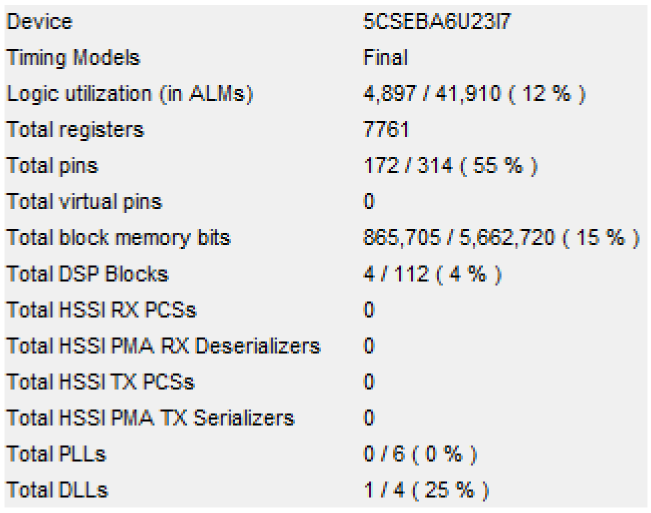
ii) Number of decisions processed in 1 second: An important aspect of using FPGA is to accelerate the forward propagation of Convolutional Neural Networks (CNN).

CNNs involve huge computing overhead of floating point additions, multiplications and division. Hence, it is necessary to have custom hardware which can make these calculations in Realtime. In general FPGA have slower clocks. But in the Cyclone V SoC due the presence of ARM processor allows the designs to run at higher clock frequency. Thus, the throughput increases. Currently, the entire program is run on the NIOS-II softcore processor and can process decisions per minute.

iii) Power consumed: The power consumed by the device is an important factor in embedded systems. It is also necessary to measure the power required by the design. The Early power estimator by Quartus allows us to measure an almost correct estimate of the power consumed by the device.

iv) Area of the hardware on the design: It is important that the design fits in the hardware present.

Current Device Area Summary (Fig 2) :

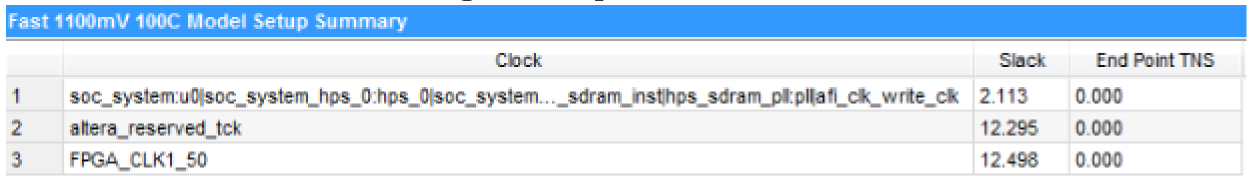


Device	5CSEBA6U23I7
Timing Models	Final
Logic utilization (in ALMs)	4,897 / 41,910 (12 %)
Total registers	7761
Total pins	172 / 314 (55 %)
Total virtual pins	0
Total block memory bits	865,705 / 5,662,720 (15 %)
Total DSP Blocks	4 / 112 (4 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	1 / 4 (25 %)

Timing Summary from TimeQuest Timing Analysis:

Fast Model:

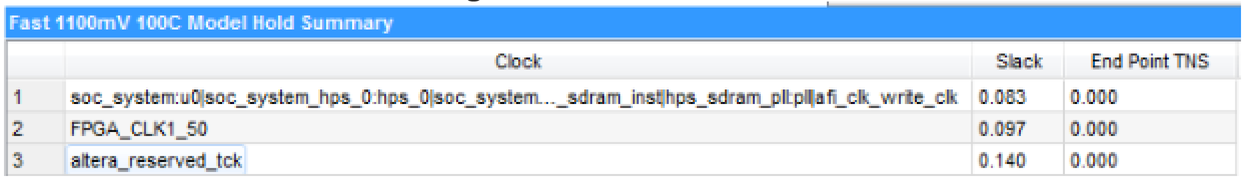
Fig 3a: Setup Time



Fast 1100mV 100C Model Setup Summary

	Clock	Slack	End Point TNS
1	soc_system:u0|soc_system_hps_0:hps_0|soc_system..._sdram_inst|hps_sdram_pll:pll|afi_clk_write_clk	2.113	0.000
2	altera_reserved_tck	12.295	0.000
3	FPGA_CLK1_50	12.498	0.000

Fig 3b: Hold Time

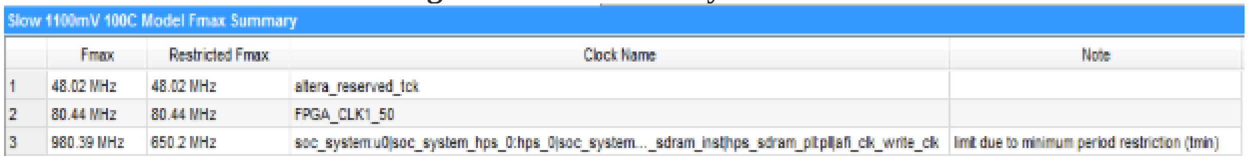


Fast 1100mV 100C Model Hold Summary

	Clock	Slack	End Point TNS
1	soc_system:u0|soc_system_hps_0:hps_0|soc_system..._sdram_inst|hps_sdram_pll:pll|afi_clk_write_clk	0.083	0.000
2	FPGA_CLK1_50	0.097	0.000
3	altera_reserved_tck	0.140	0.000

Slow Model

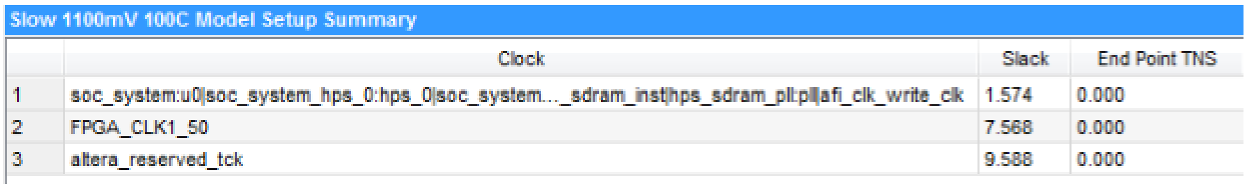
Fig 4a: Fmax Summary



Slow 1100mV 100C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	48.02 MHz	48.02 MHz	altera_reserved_tck	
2	80.44 MHz	80.44 MHz	FPGA_CLK1_50	
3	980.39 MHz	650.2 MHz	soc_system:u0|soc_system_hps_0:hps_0|soc_system..._sdram_inst|hps_sdram_pll:pll|afi_clk_write_clk	limit due to minimum period restriction (tmin)

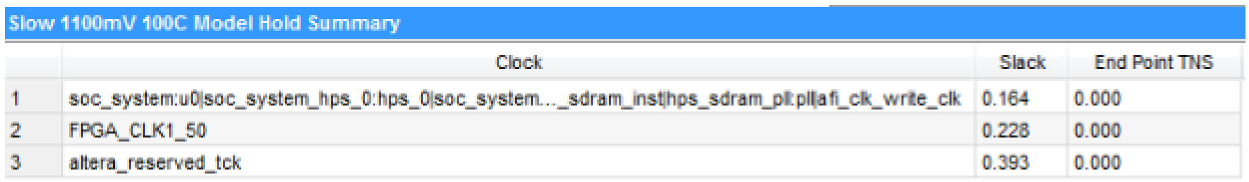
Fig 4b: Setup Time



Slow 1100mV 100C Model Setup Summary

	Clock	Slack	End Point TNS
1	soc_system:u0|soc_system_hps_0:hps_0|soc_system..._sdram_inst|hps_sdram_pll:pll|afi_clk_write_clk	1.574	0.000
2	FPGA_CLK1_50	7.568	0.000
3	altera_reserved_tck	9.588	0.000

Fig 4c: Hold Time



Slow 1100mV 100C Model Hold Summary

	Clock	Slack	End Point TNS
1	soc_system:u0|soc_system_hps_0:hps_0|soc_system..._sdram_inst|hps_sdram_pll:pll|afi_clk_write_clk	0.164	0.000
2	FPGA_CLK1_50	0.228	0.000
3	altera_reserved_tck	0.393	0.000

Design Method

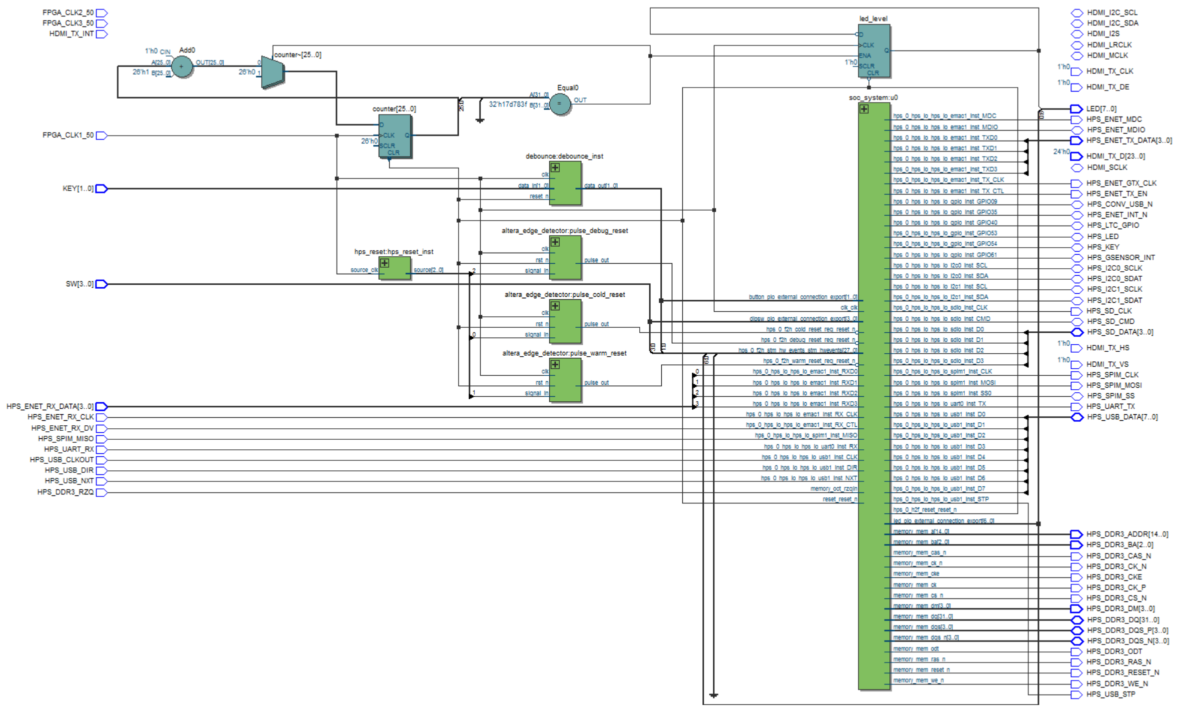


Fig 5: Top Module

Currently, the design architecture was trained on a GPU. The weights and parameters were obtained. These weights were converted into C arrays. The forward propagation of the convolutional neural network was implemented from scratch.

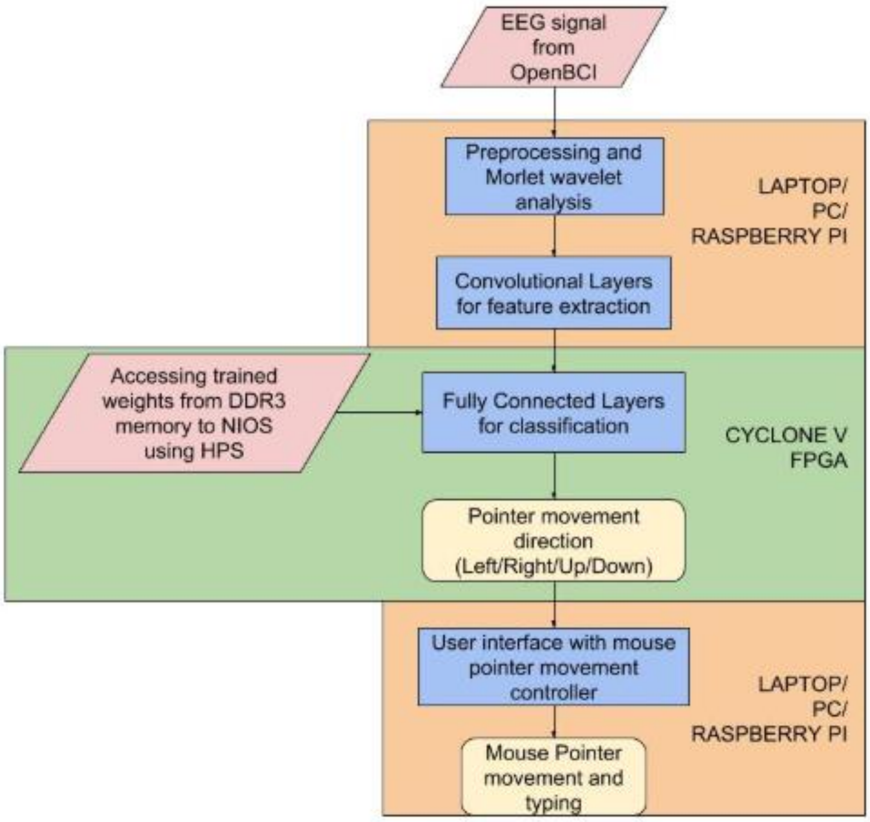

EEG signal from OpenBCI
Preprocessing and Morlet wavelet analysis
LAPTOP/ PC/ RASPBERRY PI
Convolutional Layers for feature extraction
Accessing trained weights from DDR3 memory to NIOS using HPS
Fully Connected Layers for classification
CYCLONE V FPGA
Pointer movement direction (Left/Right/Up/Down)
User interface with mouse pointer movement controller
LAPTOP/ PC/ RASPBERRY PI
Mouse Pointer movement and typing

Fig 6: Flow Chart Illustrating the Input to Output information flow

VI. Conclusion

Current Deliverables:

1) Implementation of Fully Connected, Batch Normalization and Activation layers on FPGA using NIOS II softcore processor. Current design has 862 floating point parameters in the FPGA.

2) DDR3 Memory access using HPS.

3) Acceleration (14.3% reduction in time) using the In-built floating-point hardware module.

4) Recognizing left, right, up, down movement direction from EEG signal to move the mouse pointer which selects the letters on on-screen keyboard.

Future Deliverables:

1) The current system is not real time. The data is first preprocessed then loaded into the FPGA manually. Automating this process would be the next step.

2) Replacing laptop/PC with Arduino/RaspberryPi to create a standalone system.

VII. References

1) GHRD project – NIOS DD3 Interface:
   http://www.terasic.com.tw/cgibin/page/archive.plLanguage=English&CategoryNo=205&No=1046&PartNo=4

2) NIOS II Reference:
   https://www.altera.com/en_US/pdfs/literature/tt/tt_my_first_nios_sw.pdf

3) NIOS Profiling:
   https://www.altera.com/content/dam/alterawww/global/en_US/pdfs/literature/an/an391.pdf

4) NIOS Custom Instruction:
   https://www.altera.com/en_US/pdfs/literature/ug/ug_nios2_custom_instruction.pdf