Embedded Processing Applications and Multicore

Prof. Mohsenin

CMPE 311

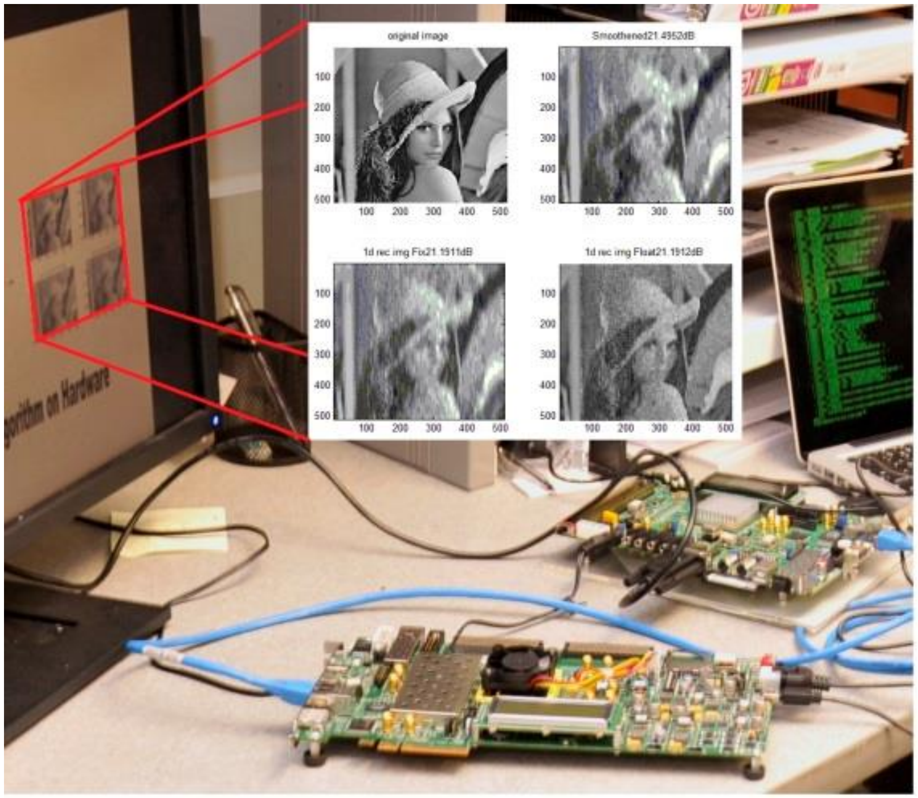
Smart Embedded Processing in Big Data World

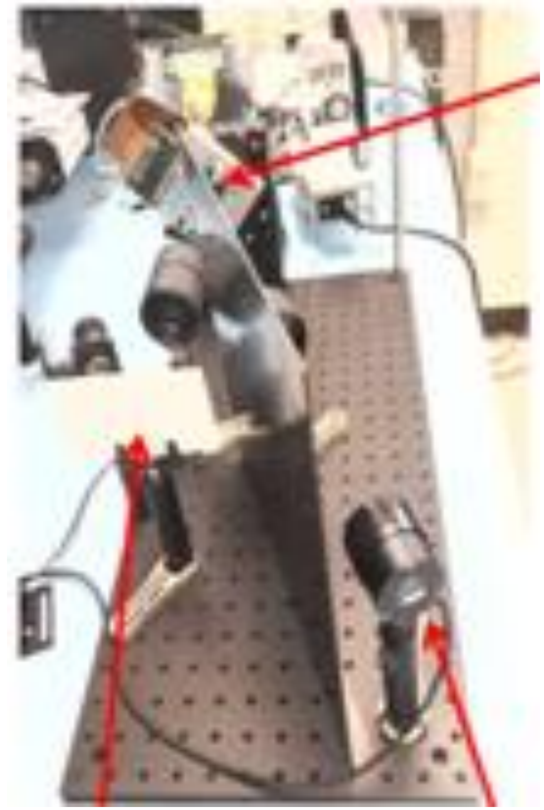
- The vast quantities of real-time data produced by embedded sensors, smartphones and wearable systems present new challenges
  - Data transmission, storage, and analysis
  - Maintaining high throughput processing and low latency communications,
  - Low power consumption.
- Systems are getting smarter and independent
  - Incorporate adaptive and intelligent kernels to overcome the noise and false detection by combining the analysis of multi-modal signals.
- Reconfiguration and programmability are required to generalize hardware for different environments and tasks
  - Reduces design time and overall time to market
- Increasing energy-efficiency (i.e. $\uparrow$GOPS/W, $\downarrow$pJ/op) requires innovations in algorithms, programming models, processor architectures, and circuit design

Embedded Applications

- Requirements:
  - Real time, low power, light weight, high accuracy
- Steps to design an embedded application on a programmable processor
  - Understand the target platform
    - e.g single processor vs multiprocessor
  - Understand the digital signal processing requirement for the application
    - What algorithms
    - How many data channels, how many bits per channel data
  - Break the application into multiple tasks
  - Write a code for each task and verify it using real/simulated data and examine the accuracy
  - Program the processor
    - Single core: all tasks in one core
    - Multi core: parallelize the tasks and program each core for the task

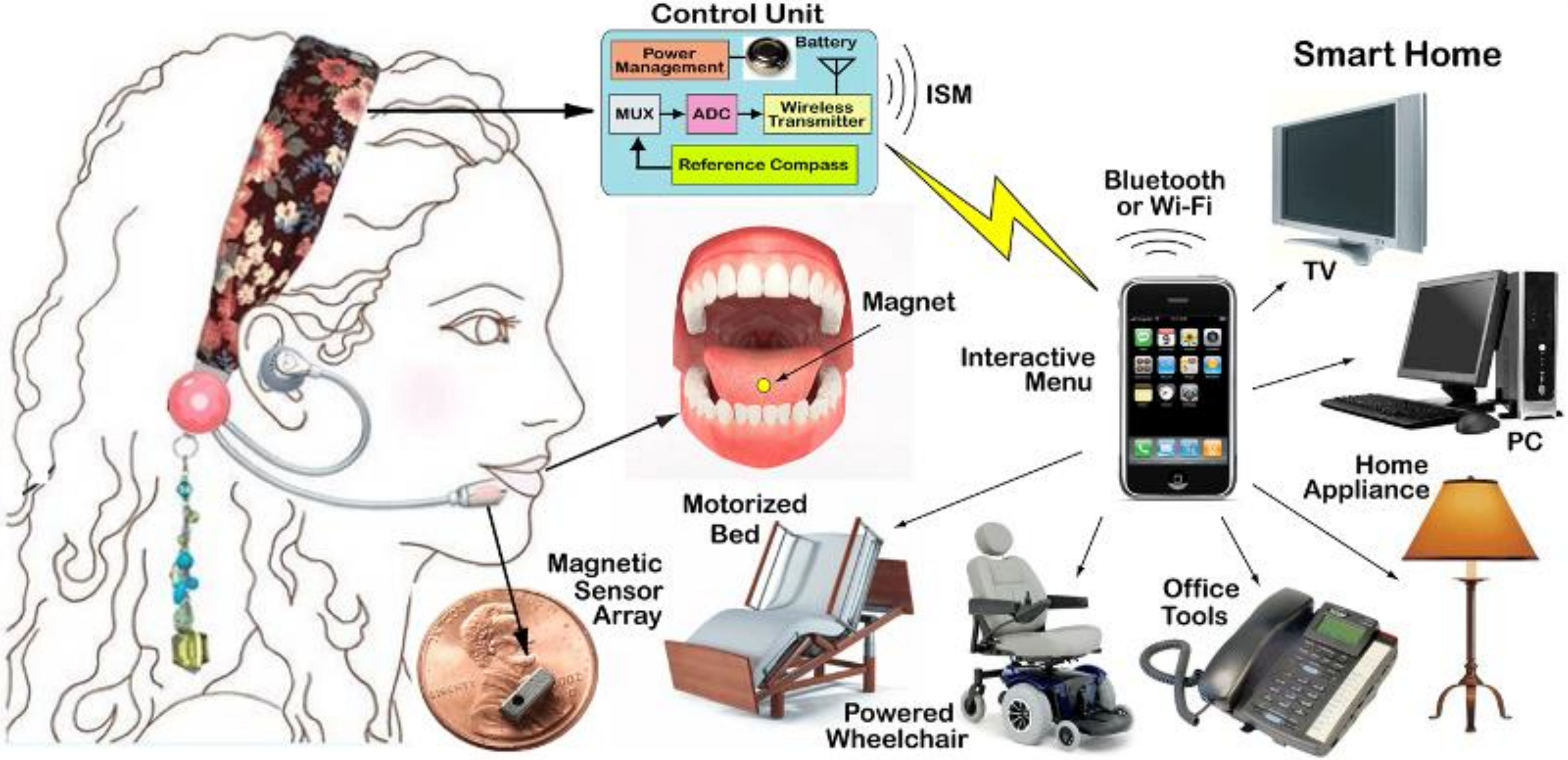
Compressive Sensing for Reduction in Data Transmission





- Single pixel camera setup at NASA Goddard
- Image reconstruction using compressive sensing on Virtex 7 FPGA

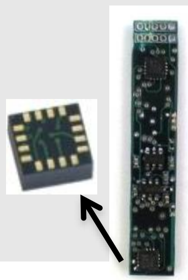
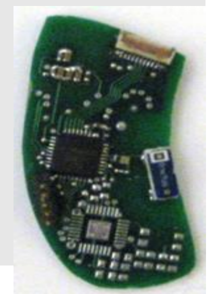
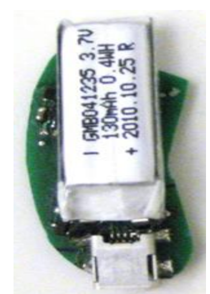
Tongue Drive System (TDS)



- A tongue-operated assistive technology that enables individuals with severe physical impairments to control their environments.
- An array of magnetic sensors detect the magnetic field variations resulted from the movements of a small magnetic tracer attached to the tongue, convert the sensed signals to the user commands in a local processor and wirelessly send the user command to the target device.

www.gtbionics.org Georgia Tech

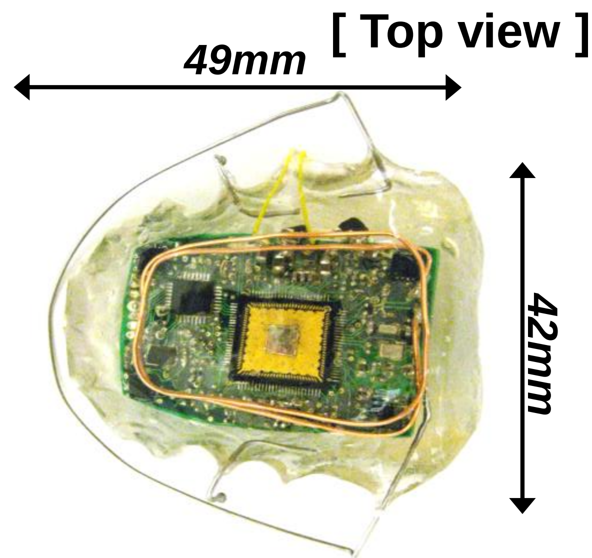
eTDS: Hardware

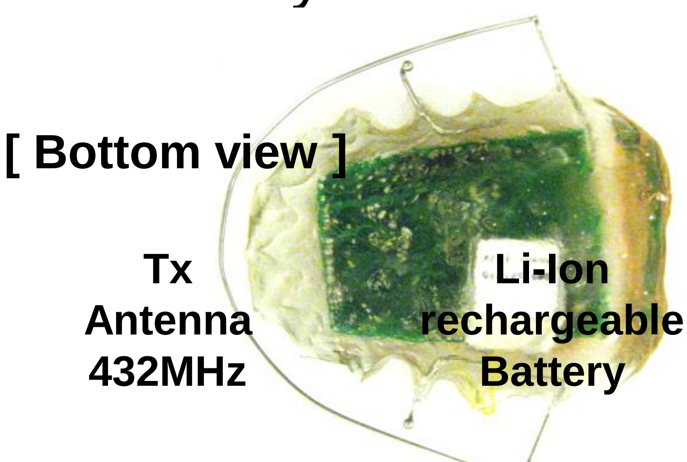
Headset Components	
1. Sensors: Four 3-axial magneto-resistive sensors (two on each pole) 	**3. Control Unit:** MCU: TI CC2510 2.4 GHz RF Transceiver 
2. Magnet: Disk-shaped [4.8mm × 1.5mm] Embedded in a titanium tongue stud 	**4. Battery:** 130 mAh, 3.7 V, plus power management circuit 

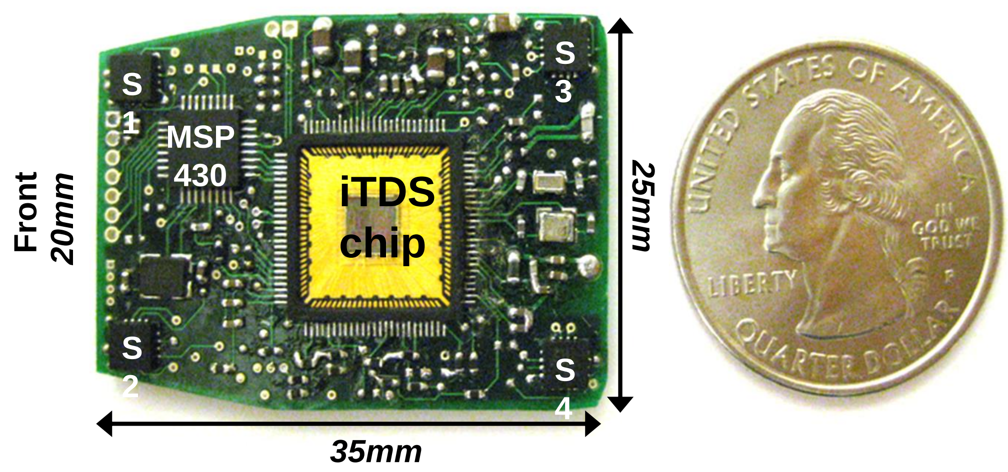
- eTDS has been clinically tested with NIH support at the top rehab institutes, such as Shepherd Center in Atlanta and Rehabilitation Institute of Chicago.

Current iTDS Prototype

- Transmits all the raw data to a computer to process
- High transmission volume cause high power consumption
  - Sends 20bits for each sensor at 50 Hz
  - There are 12 sensors => total is 12 Kbits/sec
- Size limitation restricts us to a 50mAh battery and consequently a shorter battery life







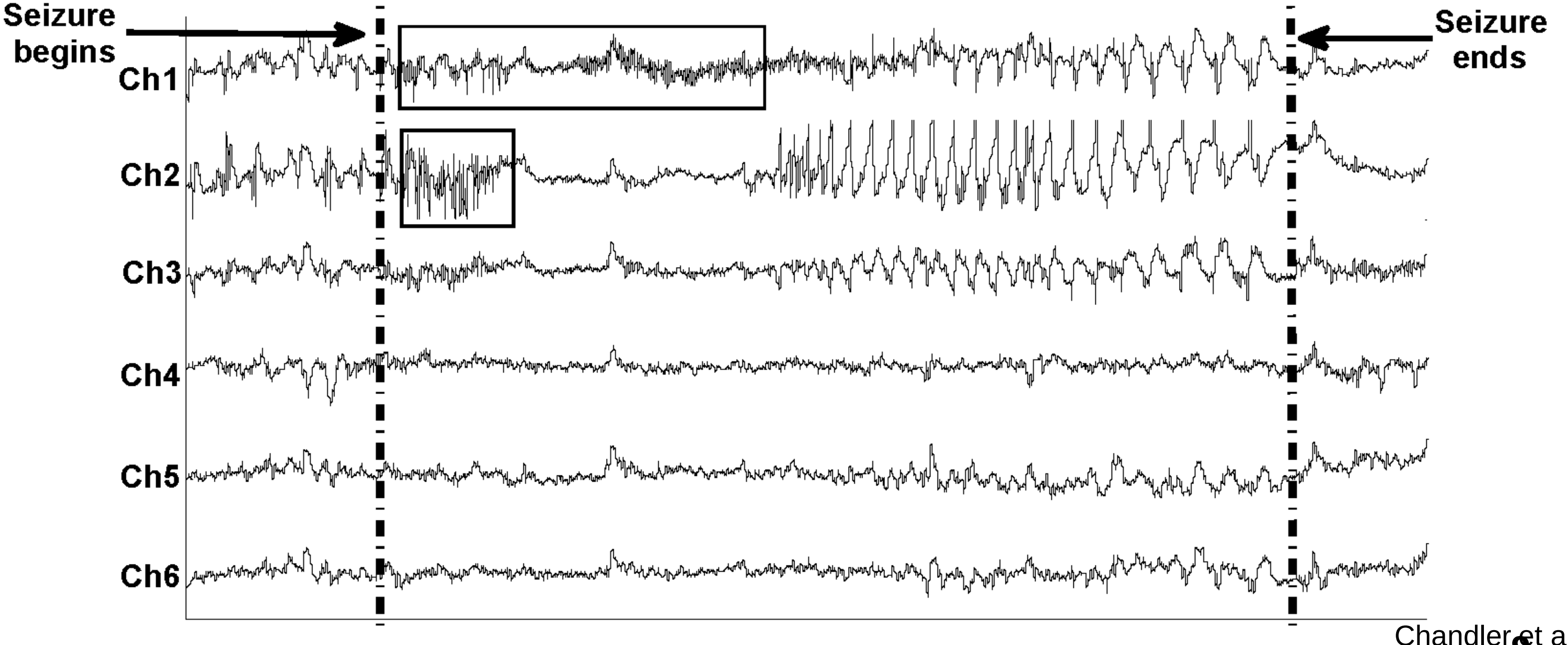
Wearable Seizure Detection



- Epilepsy is the 4th most common neurological disorder, 1 in 26 people may develop epilepsy in their lifetime.
- About 25% of epilepsy patients have intractable seizures which may occur with an unpredictable pattern, including during sleep when there may be less surveillance by family.
  - Places these patients at greatest risk from the potential morbidity and mortality of severe or sustained seizures.
- Current ambulatory seizure monitoring devices are infeasible for long-term and continuous use due to:
  - Large false positive/negative signals, noise due to patient activity, bulky equipment, high power consumption, and the inability of patients to carry on with their daily lives.

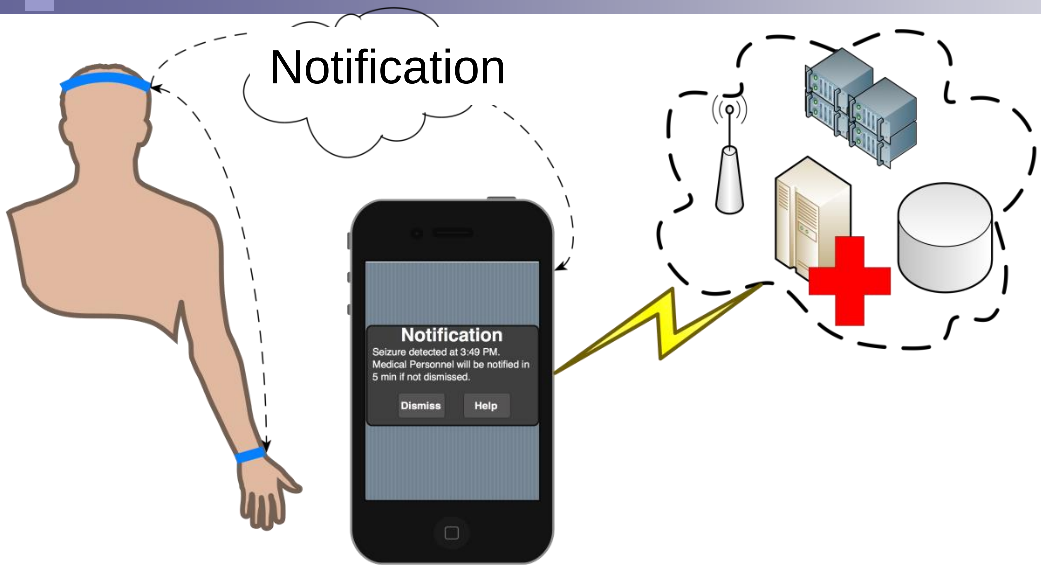
Seizure Detection Problem

- Electrical signals can be detected by EEG signals before or just at the start of clinical symptoms
  - The ability to detect can be used to warn the patient or alert caregiver
- Seizure patterns are unique to each patient and seizure and non-seizure EEG signals from the same patient can share similar characteristics
- Complex algorithms and multichannel detection is necessary for better detection

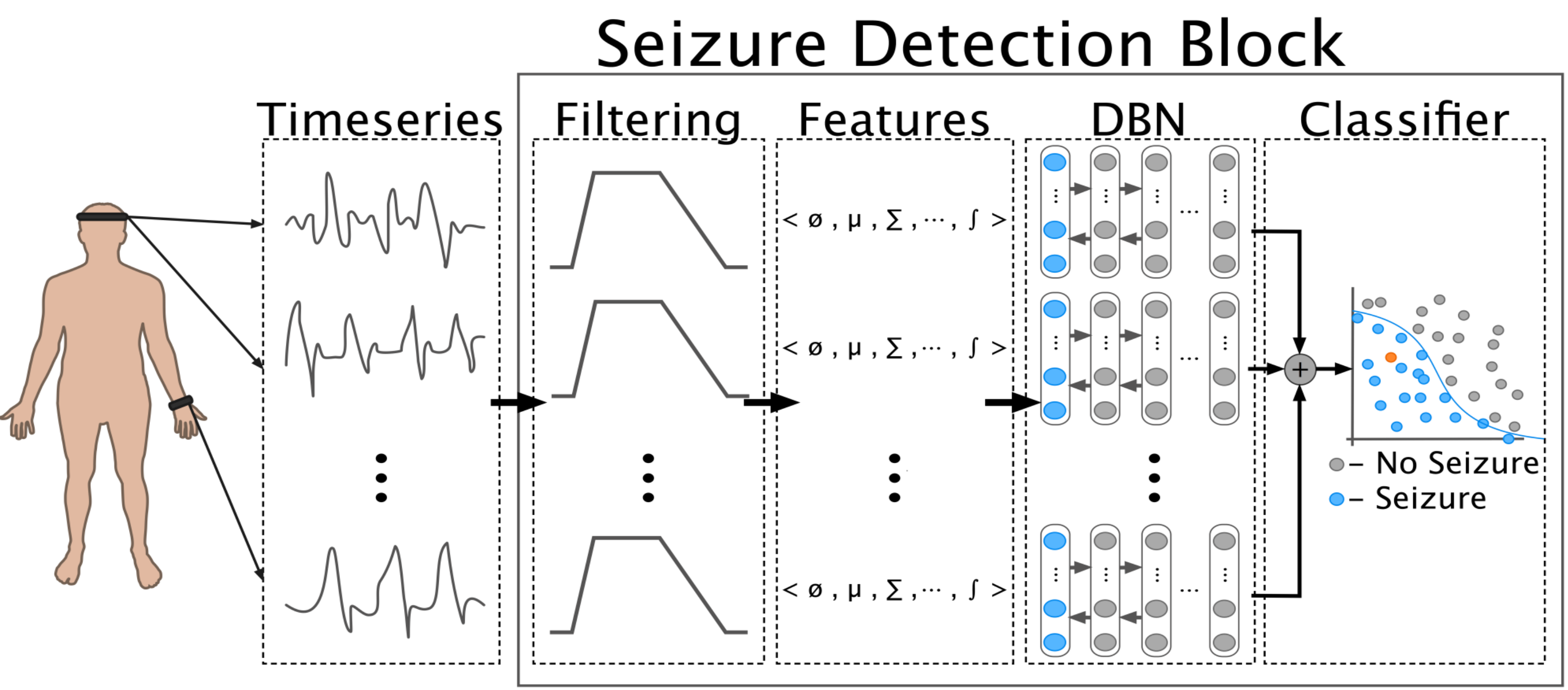


Chandler et al
BioCass 2011

A wearable solution for Multi-physiological signal processing

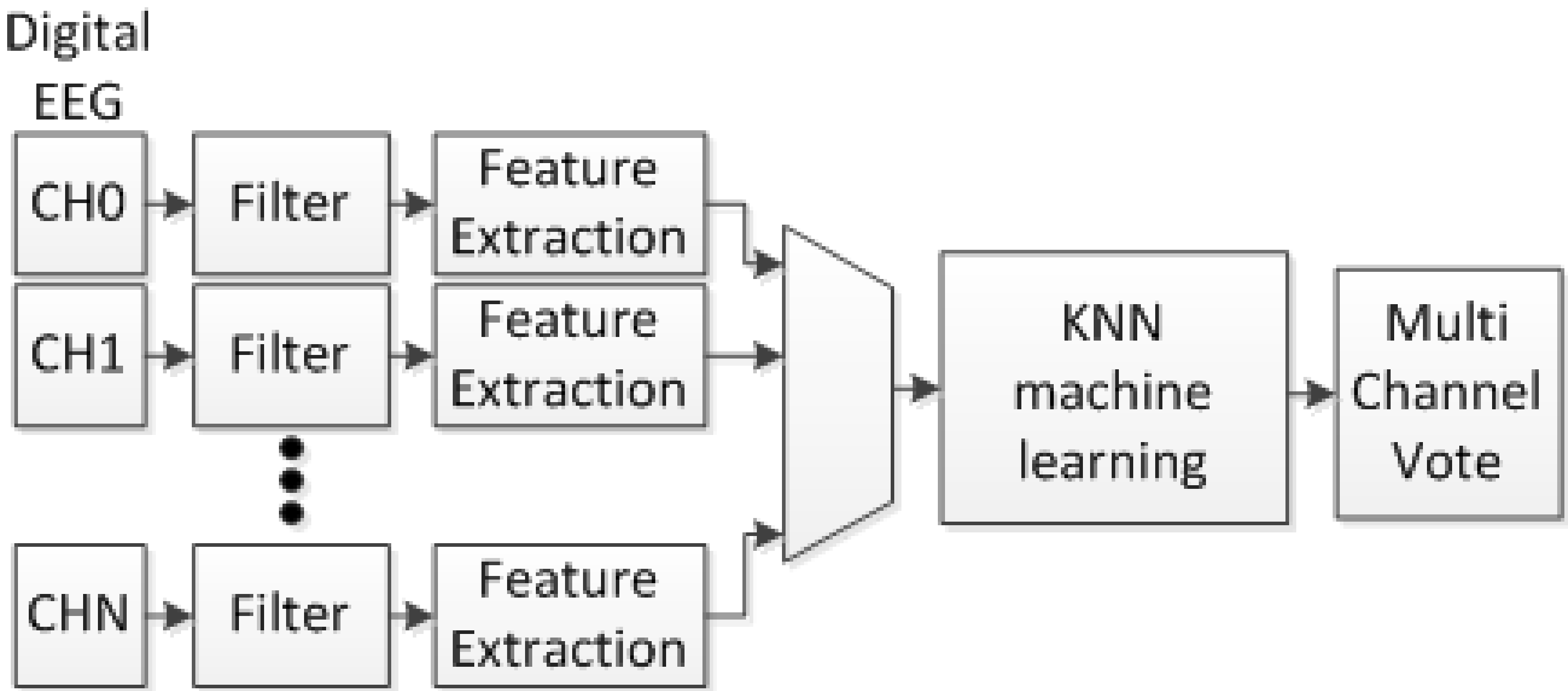


- Headband sensors
  EEG data, EOG, gyroscope data, and accelerometer
- Wristband sensors
  heart rate, blood flow, and blood oxygenation through pulse oximeter.



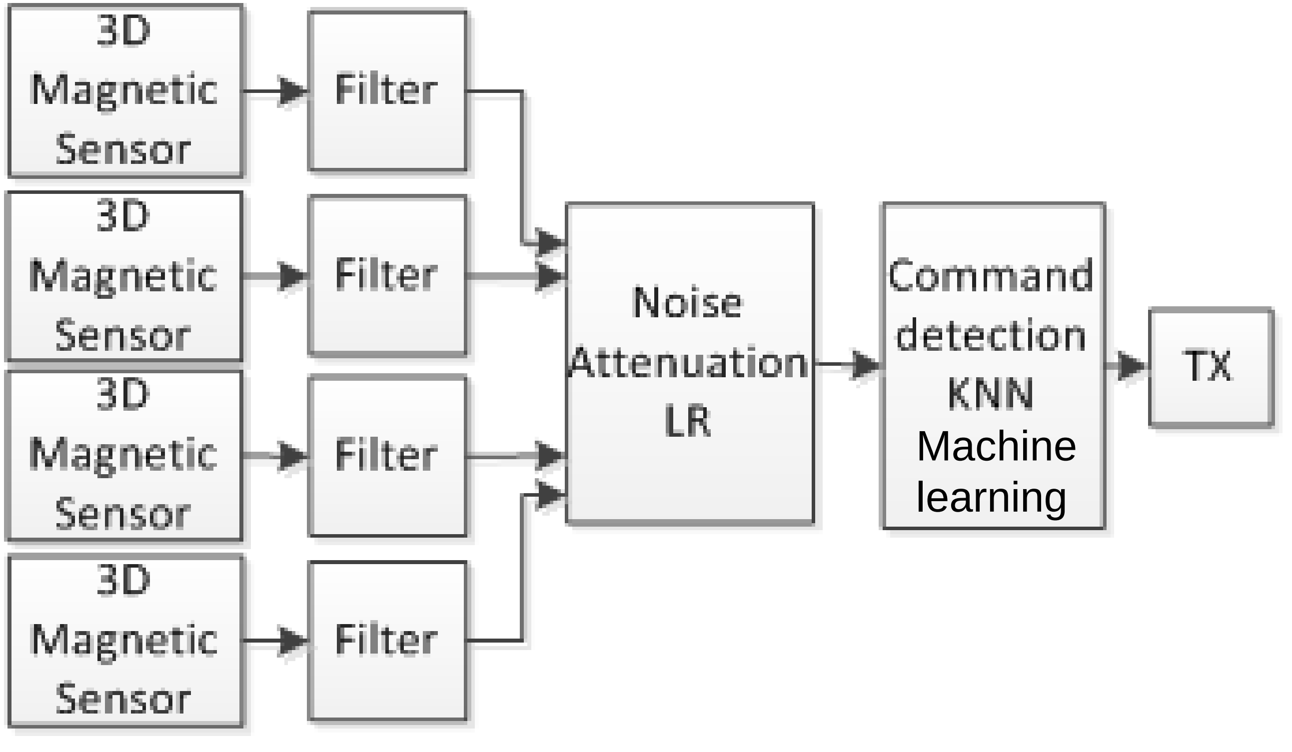
Breaking the application into multiple tasks

- Seizure detection



Breaking the application into multiple tasks

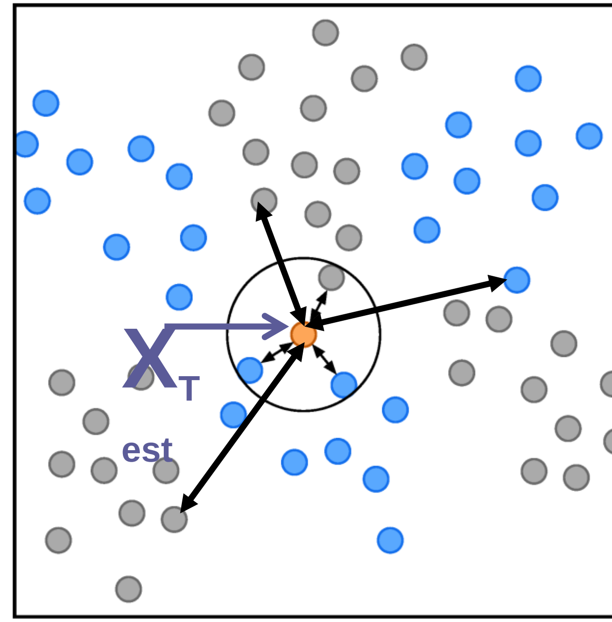
- Tongue drive system



Breaking down tasks further to multiple parallel smaller tasks

- Example K-nearest Neighborhood (KNN) Machine Learning

- Finds K- nearest neighbors to the test input and decides based on the majority vote of the neighbors.
- utilizes Euclidean distance



$$d_1 = \sqrt{(x_{Test-f1} - x_{Train-f1_1})^2 + (x_{Test-f2} - x_{Train-f2_1})^2 + \cdots + (x_{Test-fm} - x_{Train-fm_1})^2}$$

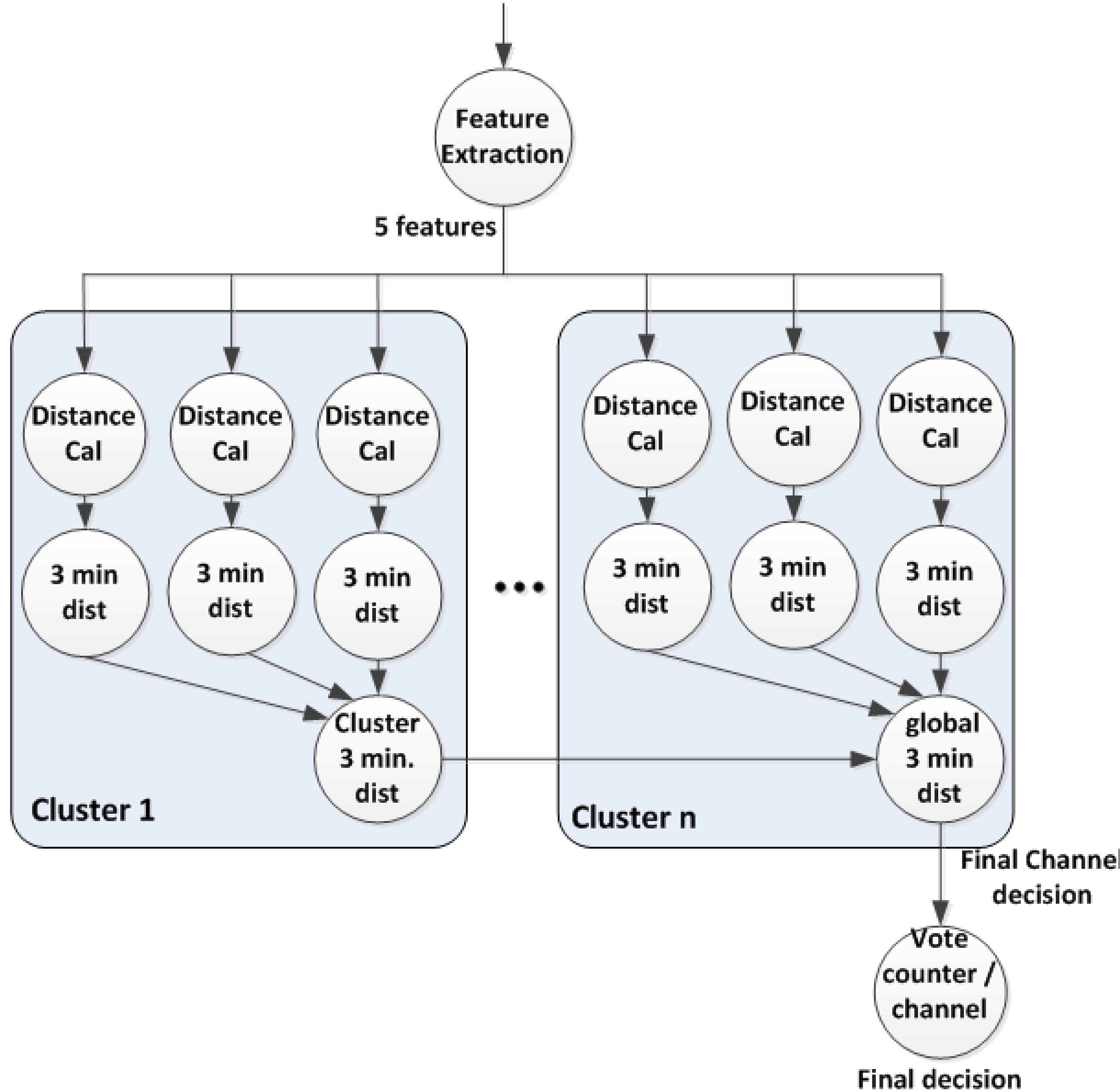
$$d_2 = \sqrt{(x_{Test-f1} - x_{Train-f1_2})^2 + (x_{Test-f2} - x_{Train-f2_2})^2 + \cdots + (x_{Test-fm} - x_{Train-fm_2})^2}$$

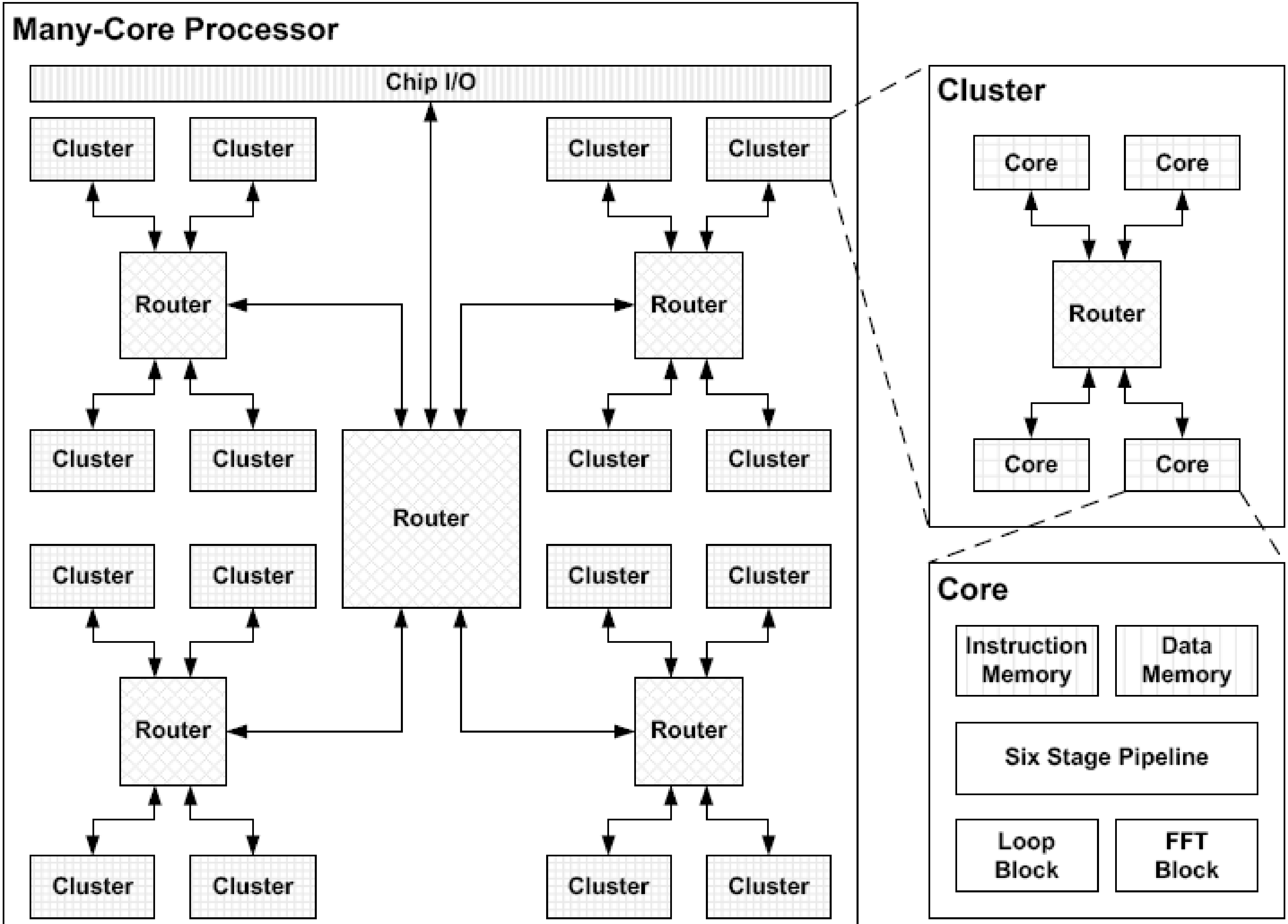
$$\vdots$$

$$d_n = \sqrt{(x_{Test-f1} - x_{Train-f1_n})^2 + (x_{Test-f2} - x_{Train-f2_n})^2 + \cdots + (x_{Test-fm} - x_{Train-fm_n})^2}$$

Breaking down tasks further to multiple parallel smaller tasks

- Example K-nearest Neighborhood (KNN) mapping





Many-Core Processor
Chip I/O
Cluster
Cluster
Cluster
Cluster
Router
Router
Cluster
Cluster
Router
Cluster
Cluster
Cluster
Cluster
Cluster
Cluster
Router
Router
Cluster
Cluster
Cluster
Cluster
Cluster
Core
Core
Router
Core
Core
Core
Instruction Memory
Data Memory
Six Stage Pipeline
Loop Block
FFT Block

Processor Pipeline

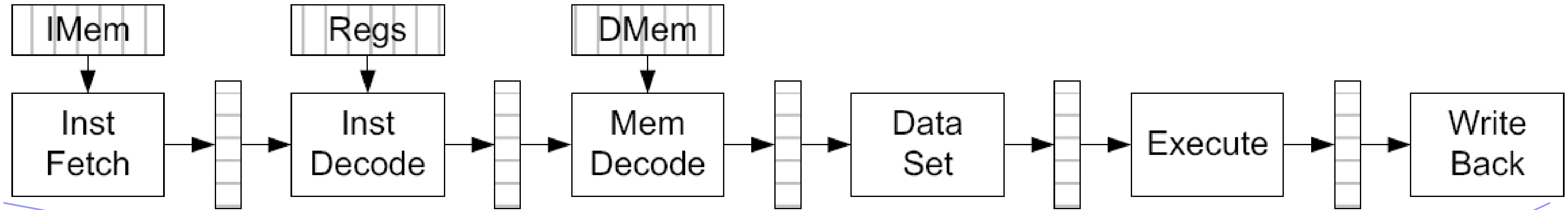
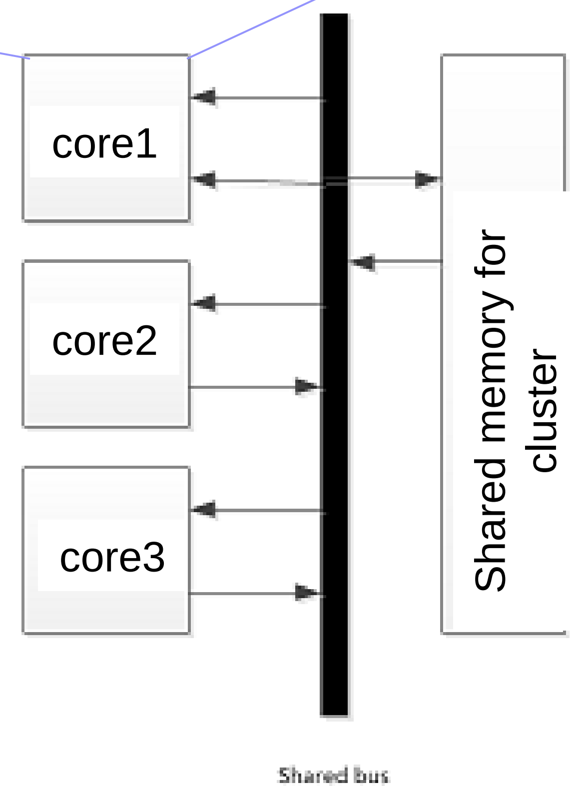
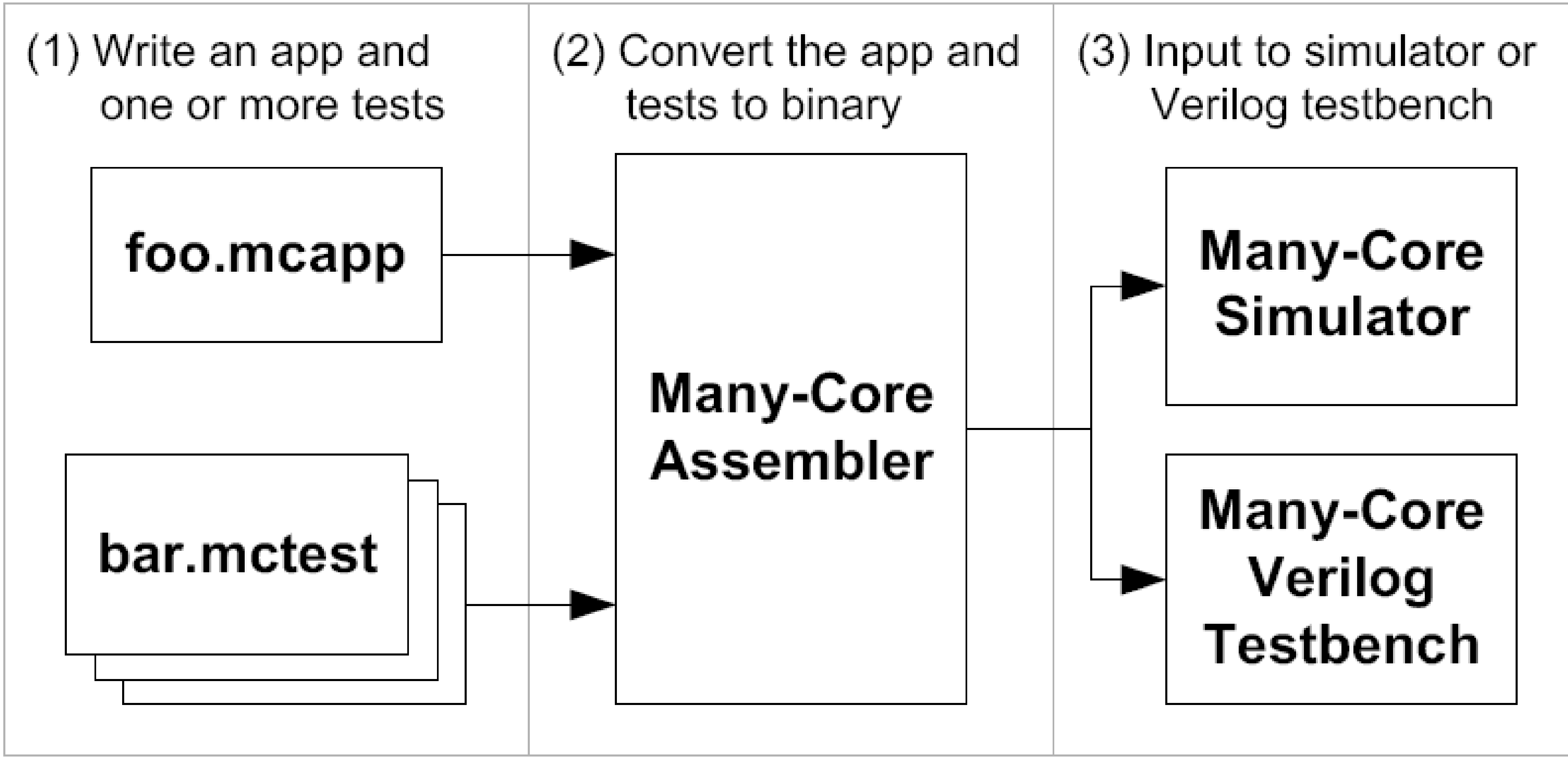


FIG. 1.2: Block diagram of the six stage pipeline



Mapping and Simulating the application on manycore



(a) Application directory structure

```
foo.mcapp/
  asm/
    inst/
      core0.asm
      core1.asm
      ...
      core63.asm
  bin/
    inst/
      core0.bin
      core1.bin
      ...
      core63.bin
  config.json
```

Example asm code for Core0

```
MUL R4 R3 R3 // power 2 to calcul
ADD R5 R5 R4 // adding up all dis
INC R1 R1 0
BG   26   R5     R9     // sorting
MOV  R11     R10     0
MOV  R10     R9     0
MOV  R9      R5     0
JMP  32   0      0
```

Challenges for multicore programming

- Parallelizing the task in a very efficient way to reduce data
- Communication between cores
  - Through shared router, bus
- Data Storage and coherence