Hardware-Software Co-Design for Brain-Computer Interfaces

Ioannis Karageorgos*, Karthik Sriram*, Ja´n Vesely´*, Michael Wu
Marc Powell ,David Borton , Rajit Manohar , Abhishek Bhattacharjee

ISCA 2020

Presented by Cheng Xuan, 10.12.2020
Problem: Current implantable BCIs (as chips) are realized with custom ASICs (ASIC: Application-specific integrated circuit) and therefore treat only certain diseases or perform specific tasks in specific brain regions (one architecture for one task) => low flexibility

Goal: Design a general-purpose architecture for implantable BCIs which realizes multiple tasks by one common architecture with low power consumption to satisfy safety constraint for implantable BCIs

Challenge: Keep power consumption low while the circuit becomes complex

General Idea: Using the principles of hardware-software co-design to preprocess the algorithms before implementing them into hardware

Key Mechanism: Refactor the underlying algorithm of each task into distinct pieces that realize different phases of the algorithm and then implement each piece into distinct hardware block (Processing Element)
  - For each supported task: Configure all necessary processing elements into pipeline to execute it

Result: Realizes an extensible and general-purpose hardware architecture for low-power implantable BCIs
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
Brain-Computer Interfaces (BCI)

• Direct pathway between the brain and an external device

• “From brain to computer” direction: Collects neuronal signal from neurons, digitize and then process

• “From computer to brain” direction: Change behavior of neurons by:
  
  1. Stimulating the neurons directly (rough way)

  2. Translating digital signals to a format which can be understood by neurons

  => Control neurons precisely: bottleneck of BCI
Brain-Computer Interfaces (BCI)

- Technology of BCI has gained more attention
- Used by over 160K patients worldwide
- Neuralink
  - Neurotechnology company
  - Founded by Elon Musk
Applications of BCI

- Treatment of neurological diseases
  E.g., epilepsy, Parkinson’s disease, anxiety

- Support research of brain functions

- Repair of perceptions (Cochlear implant)
Realization of BCI

1. **Headsets or electrodes** placed on the scalp
   - Do not require surgical deployment (safe and beneficial to commercialization)

2. **Implantable BCIs** (as chips embedded in brain tissue)
   - Need surgical deployment but enable BCI to record from and simulate large number of neurons with high signal fidelity, spatial resolution, and in real time
Outline

- Background
- Problems
- Goals
- Key Mechanism
- Implementation
- Evaluation
- Strengths
- Weaknesses
- Discussion
Problem of BCI as headsets (Non-invasive)

BCI as headsets do not satisfy the performance requirements for forward-looking BCI applications because:

1. Collected signals are noisy and low resolution
2. Less ideal for real time processing of signals
Problem of BCI as headsets (Non-invasive)

BCI as headsets do not satisfy the performance requirements for forward-looking BCI applications because:

1. Collected signals are noisy and low resolution
2. Less ideal for real time processing of signals

For this reason, this paper focuses on implantable BCI and aims to improve it.
Basic structure of implantable BCI
Basic structure of implantable BCI

1. **Sensors**: record & stimulate neurons (read & write)
   - # Channels determine **how many** neurons can be processed
1. **Sensors**: record & stimulate neurons (read & write)
   - # Channels determine **how many** neurons can be processed

2. **Analog front-end**: amplify and digitize data from sensors
   - Via ADCs
   - Sample resolution determines quality of digitization
   - Sample frequency determines speed of digitization
1. **Sensors**: record & stimulate neurons (read & write)
   - # Channels determine how many neurons can be processed

2. **Analog front-end**: amplify and digitize data from sensors
   - Via ADCs
   - Sample resolution determines quality of digitization
   - Sample frequency determines speed of digitization

3. **Communication links**: change data with the outer world
   - Via RF link
1. **Sensors**: record & stimulate neurons (read & write)
   - # Channels determine how many neurons can be processed

2. **Analog front-end**: amplify and digitize data from sensors
   - Via ADCs
   - Sample resolution determines quality of digitization
   - Sample frequency determines speed of digitization

3. **Communication links**: change data with the outer world
   - Via RF link

4. **Power sources**:
   - Single-use non-rechargeable batteries
   - Rechargeable batteries by using wireless powering
Problems of implantable BCIS

1. **Safety constraint:** Implantable BCIs must not dissipate more than 15-40mW of power (FDA, FCC, and IEEE guidelines)

2. **Low flexibility:** (Caused by safety constraint)
   - BCIs targeting large numbers of neurons: To keep the circuit simple, they are realized with custom ASICs (treat only certain diseases or perform specific tasks in specific brain regions)

     When we want to realize multiple tasks: Design ASIC for each task, and then:

     1. Combine them into one chip: called monolithic ASIC and exceed power constraint in many cases
     2. Put each ASIC into one chip: \(#\text{tasks} = \#\text{chips} \Rightarrow \text{impractical}\)

   - The few programmable BCIs: Solves flexibility problem to a certain extent, but process only a limited number of neurons to meet the low-power requirements \(\Rightarrow \text{impractical by real application}\)
Problems of implantable BCIS

1. **Safety constraint:** Implantable BCIs must not dissipate more than 15-40mW of power (FDA, FCC, and IEEE guidelines)

2. **Low flexibility:** (Caused by safety constraint)

   Flexibility and Performance: One of them as victim to satisfy power constraint

   1. Combine them into one chip: called monolithic ASIC and exceed power constraint in many cases

   2. Put each ASIC into one chip: #tasks = #chips => impractical

     - The few programmable BCIs: Solves flexibility problem to a certain extent, but process only a limited number of neurons to meet the low-power requirements => impractical by real application
Outline

• Background
• Problems
• **Goals**
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
Goals

Design a general-purpose architecture for implantable BCIs

- Realize multiple tasks by one common architecture
- Also target large number of neurons with high sampling frequency and resolution
- No need to design ASIC individually for each task

While:

Meeting power constraint of 15-40mW (adequately low-power)

- For safe and chronic implantation in the brain
Hardware Architecture for Low-power BCIs (HALO)
## HALO & BCIS based on ASIC: Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tasks Supported</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spike Detection</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

## Technical Capabilities

<table>
<thead>
<tr>
<th></th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
<th>Programmable</th>
<th>Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td><strong>Read Channels</strong></td>
<td>4</td>
<td>8</td>
<td>256</td>
<td>4</td>
<td>24</td>
<td>3072</td>
<td>32</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stimulation Channels</strong></td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sample Frequency (Hz)</strong></td>
<td>250</td>
<td>250</td>
<td>5K</td>
<td>200</td>
<td>7.2K</td>
<td>18.6K</td>
<td>256</td>
<td>30K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sample Resolution (bits)</strong></td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Safety (&lt;15mW)</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### HALO & BCIS based on ASIC: Comparison

<table>
<thead>
<tr>
<th>Tasks Supported</th>
<th>Medtronic</th>
<th>Neuropace</th>
<th>Aziz</th>
<th>Chen</th>
<th>Kassiri</th>
<th>Neuralink</th>
<th>NURIP</th>
<th>HALO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Technical Capabilities

<table>
<thead>
<tr>
<th>Programmable</th>
<th>✓</th>
<th>Limited</th>
<th>x</th>
<th>Limited</th>
<th>✓</th>
<th>x</th>
<th>Limited</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Channels</td>
<td>4</td>
<td>8</td>
<td>256</td>
<td>4</td>
<td>24</td>
<td>3072</td>
<td>32</td>
<td>96</td>
</tr>
<tr>
<td>Stimulation Channels</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Sample Frequency (Hz)</td>
<td>250</td>
<td>250</td>
<td>5K</td>
<td>200</td>
<td>7.2K</td>
<td>18.6K</td>
<td>256</td>
<td>30K</td>
</tr>
<tr>
<td>Sample Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Safety (&lt;15mW)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- HALO can realize **all** listed tasks
- Configurable and flexible
HALO & BCIS based on ASIC: Comparison

<table>
<thead>
<tr>
<th>Tasks Supported</th>
<th>Medtronic</th>
<th>Neuropace</th>
<th>Aziz</th>
<th>Chen</th>
<th>Kassiri</th>
<th>Neuralink</th>
<th>NURIP</th>
<th>HALO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

Technical Capabilities

<table>
<thead>
<tr>
<th>Programmable</th>
<th>Limited</th>
<th>Read Channels</th>
<th>Stimulation Channels</th>
<th>Sample Frequency (Hz)</th>
<th>Sample Resolution (bits)</th>
<th>Safety (&lt;15mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>Limited</td>
<td>4</td>
<td>8</td>
<td>256</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>✓</td>
<td>Limited</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>7.2K</td>
<td>10</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>3072</td>
<td>0</td>
<td>0</td>
<td>18.6K</td>
<td>10</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>256</td>
<td>16</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>30K</td>
<td>16</td>
</tr>
</tbody>
</table>

- HALO can realize all listed tasks
- Configurable and flexible
- HALO has high sample frequency & resolution
## HALO & BCIS based on ASIC: Comparison

<table>
<thead>
<tr>
<th>Tasks Supported</th>
<th>Medtronic</th>
<th>Neuropace</th>
<th>Aziz</th>
<th>Chen</th>
<th>Kassiri</th>
<th>Neuralink</th>
<th>NURIP</th>
<th>HALO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Technical Capabilities

<table>
<thead>
<tr>
<th>Programmable</th>
<th>Medtronic</th>
<th>Neuropace</th>
<th>Aziz</th>
<th>Chen</th>
<th>Kassiri</th>
<th>Neuralink</th>
<th>NURIP</th>
<th>HALO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Read Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stimulation Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample Resolution (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Safety (&lt;15mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
</tr>
</tbody>
</table>

- HALO can realize **all** listed tasks
  - Configurable and flexible
- HALO has **high** sample frequency & resolution
- HALO meets safety constraint
  - **Safe** for chronic use
  - **While** achieve the outstanding performance
HALO & BCIS based on ASIC: Comparison

- **HALO can realize all listed tasks**
  - Configurable and flexible

- **HALO has high sample frequency & resolution**

- **HALO meets safety constraint**
  - Safe for chronic use
  - While achieve the outstanding performance
HALO & BCIS based on ASIC: Comparison

- **HALO** can realize all listed tasks
  - Configurable and flexible

- **HALO** has **high** sample frequency & resolution

- **HALO** meets safety constraint
  - **Safe** for chronic use
  - **While** achieve the outstanding performance

<table>
<thead>
<tr>
<th>Tasks Supported</th>
<th>Medtronic</th>
<th>Neuropace</th>
<th>Aziz</th>
<th>Chen</th>
<th>Kassiri</th>
<th>Neuralink</th>
<th>NURIP</th>
<th>HALO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technical Capabilities</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable</td>
<td>✓</td>
<td>Limited</td>
<td>x</td>
<td>Limited</td>
<td>✓</td>
<td>x</td>
<td>Limited</td>
<td>✓</td>
</tr>
<tr>
<td>Read Channels</td>
<td>4</td>
<td>8</td>
<td>256</td>
<td>4</td>
<td>24</td>
<td>3072</td>
<td>32</td>
<td>96</td>
</tr>
<tr>
<td>Stimulation Channels</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Sample Frequency (Hz)</td>
<td>250</td>
<td>250</td>
<td>5K</td>
<td>200</td>
<td>7.2K</td>
<td>18.6K</td>
<td>256</td>
<td>30K</td>
</tr>
<tr>
<td>Sample Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Safety (&lt;15mW)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
HALO & BCIS based on ASIC: Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spike Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Compression</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Seizure Prediction</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Movement Intent</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Encryption</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

Technical Capabilities

<table>
<thead>
<tr>
<th>Programmable</th>
<th>✓</th>
<th>Limited</th>
<th>x</th>
<th>Limited</th>
<th>✓</th>
<th>x</th>
<th>Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Channels</td>
<td>4</td>
<td>8</td>
<td>256</td>
<td>4</td>
<td>24</td>
<td>3072</td>
<td>32</td>
</tr>
<tr>
<td>Stimulation Channels</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>Sample Frequency (Hz)</td>
<td>250</td>
<td>250</td>
<td>5K</td>
<td>200</td>
<td>7.2K</td>
<td>18.6K</td>
<td>256</td>
</tr>
<tr>
<td>Sample Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Safety (&lt;15mW)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

- HALO can realize all listed tasks
  - Configurable and flexible
- HALO has high sample frequency & resolution
- HALO meets safety constraint
  - Safe for chronic use
  - While achieve the outstanding performance

HALO is comprehensive and outperforms existing BCIs
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
Key Mechanism

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)

- Determine a list of tasks that we want to support by HALO
Key Mechanism

- Determine a list of tasks that we want to support by HALO
- For each task: Refactor the underlying algorithm of it into distinct pieces that realize different phases of the algorithm
Key Mechanism

- Determine a list of tasks that we want to support by HALO
- For each task: Refactor the underlying algorithm of it into distinct pieces that realize different phases of the algorithm

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)
Key Mechanism

- Determine a list of tasks that we want to support by HALO
- For each task: Refactor the underlying algorithm of it into distinct pieces that realize different phases of the algorithm
- Identify shared pieces among algorithms
Key Mechanism

- Determine a list of tasks that we want to support by HALO
- For each task: Refactor the underlying algorithm of it into distinct pieces that realize different phases of the algorithm
- Identify shared pieces among algorithms
- Implement these pieces into distinct hardware blocks (PEs)

Seizure prediction, Movement intent, Spike detection

Encryption, Compression (LZ4, LZMA, DWT)

One phase of algo

Software Level

Hardware Level

One processing element
Key Mechanism

- Determine a list of tasks that we want to support by HALO
- For each task: Refactor the underlying algorithm of it into distinct pieces that realize different phases of the algorithm
- Identify shared pieces among algorithms
- Implement these pieces into distinct hardware blocks (PEs)
- Arrange all PEs together in a suitable way
- Now for each (supported) task: controller chooses all required PEs and configure them into pipeline to execute it
General Idea

While executing LZMA:

Routing Switch

- Compression
- Movement Intent
- Seizure Prediction
- Spike Detection
- Encryption
General Idea

While executing LZMA:

Routing Switch
General Idea

While executing LZMA:

Routing Switch
General Idea

While executing LZMA:

LZ

MA → RC

AES

Routing Switch

(Optional)
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
Basic structure of HALO

Frequently used BCI tasks pipelines:

- Compression (LZ4): ADC → MUX → INT → LZ → LIC → AES
- Compression (LZMA): ADC → MUX → INT → LZ → MA → RC → AES
- Compression (DWT): ADC → MUX → INT → DWT → TOK → MA → RC → AES
- Movement Intent: ADC → MUX → INT → FFT → THR → GATE → AES
- Encryption: ADC → MUX → GATE → AES
- Spike Detection (NEO): ADC → MUX → INT → NEO → THR → GATE → AES
- Spike Detection (DWT): ADC → MUX → DWT → THR → GATE → AES
- Seizure Prediction: ADC → MUX → INT → (FFT, XCOR, BBF) → SVM → THR → GATE → AES
Basic structure of HALO

For each BCI task, Controller assembles all required PEs for this task into pipelines to execute the task.
Basic structure of HALO

For each BCI task: **Controller assembles** all required PEs for this task into **pipelines** to execute the task

Each **single PE** operates at:
- a frequency **catered to its specific computational needs** => reduces power consumption (while ASIC ran all logic at **same** frequency)
- private memory => **cannot share** large amounts of data ( "Locality Refactoring" by PE decomposition)
- adapter to communicate over the interconnect

=> PEs communicate with each other via lower-power circuit-switched network built on an asynchronous communication fabric
For each BCI task: Controller assembles all required PEs for this task into pipelines to execute the task.

Each single PE operates at:
- a frequency catered to its specific computational needs => reduces power consumption (while ASIC ran all logic at same frequency)
- private memory => cannot share large amounts of data ("Locality Refactoring" by PE decomposition)
- adapter to communicate over the interconnect

=> PEs communicate with each other via lower-power circuit-switched network built on an asynchronous communication fabric

RISC-V Micro-controller: Assembles PEs into pipelines for each task and interrupts PEs by power overshoot
PE decomposition

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)
PE decomposition

- PE decomposition: Process that *refactoring* underlying algorithm of tasks into distinct pieces and then implement pieces into distinct *hardware blocks* (Processing Elements)

- Complexity depends on *how clearly separated* the algorithmic phases are
LZMA: One algorithm to realize **data compression**

=> reduces radio transmission and useful for high-bandwidth brain interaction

---

**Algorithm 1** LZMA pseudocode

```python
1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:         Probmatch = count(table_match, best_match) / count_total(table_match);
6:         r1 = range_encode(Probmatch);
7:         output.push_back(r1);
8:         increment_counter(table_match, best_match);
9:     end while
10:    return output;
11: end function
```
Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:    output = list(lzma_header);
3:    while data = input.get() do
4:        best_match = find_best_match(data);
5:        \[ \text{Prob}_\text{match} = \text{count}(\text{table}_\text{match}, \text{best}_\text{match}) / \text{count}_\text{total}(\text{table}_\text{match}) \]
6:        \[ r1 = \text{range}_\text{encode}(\text{Prob}_\text{match}) \]
7:        output.push_back(r1);
8:        increment_counter(\text{table}_\text{match}, \text{best}_\text{match});
9:    end while
10:  return output;
11: end function

Initial Version: implement Algorithm line by line into hardware blocks
Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best match = find_best_match(data);
5:         $ Prob_{match} = \text{count}(table_{match}, best_{match}) $
6:         /$ \text{count_total}(table_{match}) $;
7:         $ r1 = \text{range_encode}(Prob_{match}) $;
8:         output.push_back(r1);
9:         increment_counter(table_{match}, best_{match});
10:     end while
11:     return output;
12: end function

Line 5&6: use Markov (MA) chains to calculate the probability of the current input value based on observed history (frequency table)
Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:  output = list(lzma_header);
3:  while data = input.get() do
4:    best_match = find_best_match(data);
5:    Prob_match = count(table_match, best_match)
6:    /count_total(table_match);
7:    r1 = range_encode(Prob_match);
8:    output.push_back(r1);
9:    increment_counter(table_match, best_match);
10: end while
11: return output;
12: end function

Line 7&8: try to pick more efficient encoding of the input signal based on the calculated probability in the last step
Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:         Prob_match = count(table_match, best_match)
6:             /count_total(table_match);
7:         r1 = range_encode(Prob_match);
8:         output.push_back(r1);
9:     increment_counter(table_match, best_match);
10:     end while
11:     return output;
12: end function

Line 9: update frequency table
PE decomposition: Example

Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:    output = list(lzma_header);
3:    while data = input.get() do
4:        best_match = find_best_match(data);
5:        Prob_match = count(table_match, best_match)
6:            /count_total(table_match);
7:        r1 = range_encode(Prob_match);
8:        output.push_back(r1);
9:        increment_counter(table_match, best_match);
10:    end while
11:    return output;
12: end function

- **Duplication** of hardware component with similar / same functionality

- Line 5,6 & 9 share the same table, but they are separated into two PEs

Since each PE has **private memory** => unnecessary data movement
PE decomposition: Example

Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:         Prob_match = count(table_match, best_match)
6:          /count_total(table_match);
7:         r1 = range_encode(Prob_match);
8:         output.push_back(r1);
9:         increment_counter(table_match, best_match);
10:    end while
11: return output;
12: end function

• Bring together phases that operate on the same data structures
• Separate the PEs => operate independently with minimal data movement
Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:    output = list(lzma_header);
3:    while data = input.get() do
4:        best_match = find_best_match(data);
5:        Prob_match = count(table_match, best_match)
6:        /count_total(table_match);
7:        r1 = range_encode(Prob_match);
8:        output.push_back(r1);
9:        increment_counter(table_match, best_match);
10:    end while
11:    return output;
12: end function

Green part: operations related to frequency table
**Algorithm 1** LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:         Prob\_match = count(table\_match, best\_match)
6:             /count\_total(table\_match);
7:         r1 = range\_encode(Prob\_match);
8:         output.push\_back(r1);
9:         increment\_counter(table\_match, best\_match);
10:     end while
11: return output;
12: end function

**Blue part:** operations related to encoder state
Key Mechanism: Recap

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)
Key Mechanism: Recap

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)

1. Choose LZMA task and its underlying algorithm to process at first
Seizure prediction, Movement intent, Spike detection

Encryption, Compression (LZ4, LZMA, DWT)

Key Mechanism: Recap

1. Choose LZMA task and its underlying algorithm to process at first
2. & 3. Implement line 5, 6, 9 into processing element “MA”
4. Implement line 9 into processing element “RC”

Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:             ProbMatch = count(table_match, best_match);
6:             Count_total(table_match);
7:             I = range_encode(ProbMatch);
8:             output.push_back(I);
9:             increment_counter(table_match, best_match);
10:        end while
11:     return output;
12: end function
Key Mechanism: Recap

Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)

1. Choose LZMA task and its underlying algorithm to process at first
2. & 3. Implement line 5, 6, 9 into processing element “MA”
4. Implement line 9 into processing element “RC”
5. & 6. The resulting processing elements corresponds to square blocks MA and RC in the figure
Seizure prediction, Movement intent, Spike detection

Encryption, Compression (LZ4, LZMA, DWT)

Key Mechanism: Recap

Algorithm 1 LZMA pseudocode

1: function LZMA_COMPRESS_BLOCK(input)
2:     output = list(lzma_header);
3:     while data = input.get() do
4:         best_match = find_best_match(data);
5:         Prob_match = count(table_match, best_match)
6:         count_total(table_match);
7:         r3 = range_encode(Prob_match);
8:         output.push_back(r3);
9:         count_counter(table_match, best_match);
10:     end while
11:     return output;
12: end function

① Choose LZMA task and its underlying algorithm to process at first
② & ③ Implement line 5,6,9 into processing element “MA”
④ Implement line 9 into processing element “RC”
⑤ & ⑥ The resulting processing elements corresponds to square blocks MA and RC in the figure
⑦ Arrange the resulting PEs together and get the final architecture
Seizure prediction, Movement intent, Spike detection
Encryption, Compression (LZ4, LZMA, DWT)

While executing LZMA:

① Choose LZMA task and its underlying algorithm to process at first
② & ③ Implement line 5, 6, 9 into processing element “MA”
④ Implement line 9 into processing element “RC”
⑤ & ⑥ The resulting processing elements corresponds to square blocks MA and RC in the figure
⑦ Arrange the resulting PEs together and get the final architecture
⑧ When executing LZMA: configure the corresponding pipeline
Optimization of Processing elements

- Optimization in software level
- Does not change output
Optimization of Processing elements

- Optimization in software level
- Does not change output

Example:

Algorithm 2 XCOR naïve implementation

1: function XCOR(input, output)
2:     // channel[][] stores input in appropriate channel location
3:     channel[channel_num][sample_num] = input
4:     // Calculate correlation
5:     if channel.filled() then
6:         for each $i,j \in \text{channels}$ do
7:             data$_i$ = 0
8:             for each data $\in \text{channel}[i]$ do
9:                 data$_i+$ = data
10:         end for
11:         data$_j$ = 0
12:         for $k \in [LAG, SIZE]$ do
13:             data$_j+$ = channel[j][k]
14:         end for
15:         avg$_i$ = data$_i$/SIZE
16:         avg$_j$ = data$_j$/SIZE
17:         output.push_back(avg$_i$, avg$_j$)
18:     end for
19:     return output;
20: end if
21: end function
Optimization of Processing elements

- Optimization in software level
- Does not change output

Example:

```
Algorithm 2 XCOR naive implementation

1: function XCOR(input, output)
2:   // channel[][] stores input in appropriate channel location
3:   channel[channel_num][sample_num] = input
4:   // Calculate correlation
5:   if channel.filled() then
6:     for each i, j ∈ channels do
7:       data_i = 0
8:       for each data ∈ channel[i] do
9:         data_i+ = data
10:      end for
11:     data_j = 0
12:     for k ∈ [LAG, SIZE] do
13:       data_j+ = channel[j][k]
14:     end for
15:     avg_i = data_i/SIZE
16:     avg_j = data_j/SIZE
17:     output.push_back(avg_i, avg_j)
18:   end for
19:   return output;
20: end if
21: end function
```

- Process data in blocks instead of samples
- Wait for all inputs in the block to arrive
Optimization of Processing elements

- Optimization in software level
- Does not change output

Example:

```plaintext
Algorithm 2 XCOR naive implementation
1: function XCOR(input, output)
2:    // channel[][] stores input in appropriate channel location
3:    channel[channel_num][sample_num] = input
4:    // Calculate correlation
5:    if channel.filled() then
6:        for each i, j in channels do
7:            data_i = 0
8:                for each data ∈ channel[i] do
9:                    data_i += data
10:               end for
11:               data_j = 0
12:               for k ∈ [LAG, SIZE] do
13:                   data_j += channel[j][k]
14:               end for
15:               avg_i = data_i / SIZE
16:               avg_j = data_j / SIZE
17:               output.push_back(avg_i, avg_j)
18:           end for
19:        return output;
20:    end if
21: end function
```

- Process data in blocks instead of samples
- Wait for all inputs in the block to arrive
- When all inputs arrive, computation occurs in a burst
- Requires large buffers to sink the bursts, or high PE frequency
Optimization of Processing elements

- Optimization in software level
- Does not change output

Example:

```
Algorithm 3 XCOR spatial programming refactoring
1: function XCOR(input, output)
2:   // channel[][] stores input in appropriate channel location
3:   channel[channel_num][sample_num] = input
4:   // data[] stores sums of input received so far
5:   data[count]++ = input
6:   // data_lag[] stores sums of input till LAG
7:   if count_2 == LAG then
8:      data_lag[count] = data[count]
9:   end if
10:  // Finish correlation computation
11:  if channel.filled() then
12:      for each i, j ∈ channels do
13:         avg_i = data[i]/SIZE
14:         avg_j = (data[j] - data_lag[j])/SIZE
15:         output.push_back(avg_i, avg_j)
16:      end for
17:   end if
18:   return output
19: end function
```

Optimization: avoid the bursty computation
Optimization of Processing elements

- Optimization in software level
- Does not change output

---

**Algorithm 3** XCOR spatial programming refactoring

```
1: function XCOR(input, output)
2:    // channel[] stores input in appropriate channel location
3:    channel[channel_num][sample_num] = input
4:    // data[] stores sums of input received so far
5:    data[count]++ = input
6:    // data_lag[] stores sums of input till LAG
7:    if count_2 == LAG then
8:        data_lag[count] = data[count]
9:    end if
10:   // Finish correlation computation
11:   if channel.filled() then
12:       for each i, j in channels do
13:           avg_i = data[i]/SIZE
14:           avg_j = (data[j] - data_lag[j])/SIZE
15:           output.push_back(avg_i, avg_j)
16:       end for
17:   return output
18: end function
```

---

**Example:**

- Complete part of computation while reading inputs

**Optimization:** avoid the bursty computation
Optimization of Processing elements

- Optimization in software level
- Does not change output

Example:

```
Algorithm 3 XCOR spatial programming refactoring
1: function XCOR(input, output)
2:   // channel[][] stores input in appropriate channel location
3:   channel[channel_num][sample_num] = input
4:   // data[] stores sums of input received so far
5:   data[count]++ = input
6:   // data_lag[] stores sums of input till LAG
7:   if count_2 == LAG then
8:     data_lag[count] = data[count]
9:   end if
10:  // Finish correlation computation
11:  if channel_filled() then
12:     for each i, j in channels do
13:       avg_i = data[i]/SIZE
14:       avg_j = (data[j] - data_lag[j])/SIZE
15:       output.push_back(avg_i, avg_j)
16:     end for
17:     return output
18:   end if
19: end function
```

Optimization: avoid the bursty computation

- Complete part of computation while reading inputs
- Amount of computation needed in the final step is reduced
- Power savings of 2.2× over the original algorithm
Optimization of Processing elements

- Adapting the precision => reduce power consumption significantly while causing slight error

Example:

- Unnecessary high resolution by some of the signal processing algorithms (32 bit integers)

- Replace floating point arithmetic with fixed point arithmetic (e.g. in BBF PE)

- Results in only < 0.1% increase in relative error and an order of magnitude reduction in power
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
HALO versus RISC-V and monolithic ASICs

HALO can satisfy the constraint (red line) for all tasks
# Power Analysis of HALO

## Tasks

<table>
<thead>
<tr>
<th>PE</th>
<th>Freq (MHz)</th>
<th>Logic (mW)</th>
<th>Mem (mW)</th>
<th>Total (mW)</th>
<th>Area (KGE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Leak</td>
<td>Dyn</td>
<td>Leak</td>
<td>Dyn</td>
</tr>
<tr>
<td>LZ</td>
<td>129</td>
<td>0.055</td>
<td>1.455</td>
<td>0.095</td>
<td>1.466</td>
</tr>
<tr>
<td>LIC</td>
<td>22.5</td>
<td>0.057</td>
<td>0.267</td>
<td>0.066</td>
<td>0.046</td>
</tr>
<tr>
<td>MA</td>
<td>92</td>
<td>0.127</td>
<td>2.148</td>
<td>0.067</td>
<td>0.997</td>
</tr>
<tr>
<td>RC</td>
<td>90</td>
<td>0.029</td>
<td>0.763</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DWT</td>
<td>3</td>
<td>0.004</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NEO</td>
<td>3</td>
<td>0.012</td>
<td>0.003</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FFT</td>
<td>15.7</td>
<td>0.057</td>
<td>0.509</td>
<td>0.085</td>
<td>0.356</td>
</tr>
<tr>
<td>XCOR</td>
<td>85</td>
<td>0.07</td>
<td>4.182</td>
<td>0.307</td>
<td>0.053</td>
</tr>
<tr>
<td>BBF</td>
<td>6</td>
<td>0.066</td>
<td>0.034</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SVM</td>
<td>3</td>
<td>0.018</td>
<td>0.018</td>
<td>0.081</td>
<td>0.033</td>
</tr>
<tr>
<td>THR</td>
<td>16</td>
<td>0.002</td>
<td>0.011</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GATE</td>
<td>5</td>
<td>0.003</td>
<td>0.006</td>
<td>0.067</td>
<td>0.054</td>
</tr>
<tr>
<td>AES</td>
<td>5</td>
<td>0.053</td>
<td>0.059</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Compressions and Seizure Prediction

- **Compr (LZ4)**: 0.112, 1.722, 0.101, 1.512, 3.447, 80
- **Compr (LZMA)**: 0.211, 4.366, 0.122, 2.463, 7.162, 133
- **Compr (DWTMA)**: 0.16, 2.913, 0.0123, 0.33, 3.415, 80
- **Seizure Prediction**: 0.216, 4.760, 0.54, 0.496, 6.012, 111
- **Spike Det (NEO)**: 0.017, 0.02, 0.067, 0.054, 0.158, 24
- **Spike Det (DWT)**: 0.009, 0.019, 0.067, 0.054, 0.149, 20
- **Movement Intent**: 0.062, 0.526, 0.152, 0.41, 1.15, 40
- **Encrypt (Raw)**: 0.053, 0.059, 0, 0, 0.112, 34
- **RISC-V Control**: 0.341, 0.137, 0.248, 1.080, 1.800, 70

**Power Consume of each pipeline under 15 mW**

**Compression and seizure prediction: consume the most power**
Power Analysis of HALO

<table>
<thead>
<tr>
<th>PE</th>
<th>Freq (MHz)</th>
<th>Logic (mW)</th>
<th>Mem (mW)</th>
<th>Total (mW)</th>
<th>Area (KGE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Leak</td>
<td>Dyn</td>
<td>Leak</td>
<td>Dyn</td>
</tr>
<tr>
<td>LZ</td>
<td>129</td>
<td>0.055</td>
<td>1.453</td>
<td>0.095</td>
<td>1.466</td>
</tr>
<tr>
<td>LIC</td>
<td>22.5</td>
<td>0.057</td>
<td>0.267</td>
<td>0.006</td>
<td>0.046</td>
</tr>
<tr>
<td>MA</td>
<td>92</td>
<td>0.127</td>
<td>2.148</td>
<td>0.067</td>
<td>0.997</td>
</tr>
<tr>
<td>RC</td>
<td>90</td>
<td>0.029</td>
<td>0.763</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DWT</td>
<td>3</td>
<td>0.004</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NEO</td>
<td>3</td>
<td>0.012</td>
<td>0.003</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FFT</td>
<td>15.7</td>
<td>0.057</td>
<td>0.509</td>
<td>0.085</td>
<td>0.356</td>
</tr>
<tr>
<td>XCOR</td>
<td>85</td>
<td>0.07</td>
<td>4.182</td>
<td>0.307</td>
<td>0.053</td>
</tr>
<tr>
<td>BBF</td>
<td>6</td>
<td>0.066</td>
<td>0.034</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SVM</td>
<td>3</td>
<td>0.018</td>
<td>0.018</td>
<td>0.081</td>
<td>0.033</td>
</tr>
<tr>
<td>THR</td>
<td>16</td>
<td>0.002</td>
<td>0.011</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GATE</td>
<td>5</td>
<td>0.003</td>
<td>0.006</td>
<td>0.067</td>
<td>0.054</td>
</tr>
<tr>
<td>AES</td>
<td>5</td>
<td>0.053</td>
<td>0.059</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Compr (LZ4)</th>
<th>Compr (LZMA)</th>
<th>Compr (DWTMA)</th>
<th>Seizure Prediction</th>
<th>Spike Det (NEO)</th>
<th>Spike Det (DWT)</th>
<th>Movement Intent</th>
<th>Encrypt (Raw)</th>
<th>RISC-V Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.112</td>
<td>0.211</td>
<td>0.16</td>
<td>0.216</td>
<td>0.017</td>
<td>0.009</td>
<td>0.062</td>
<td>0.053</td>
<td>0.341</td>
</tr>
<tr>
<td></td>
<td>1.722</td>
<td>4.366</td>
<td>2.913</td>
<td>4.760</td>
<td>0.02</td>
<td>0.019</td>
<td>0.526</td>
<td>0.059</td>
<td>0.137</td>
</tr>
<tr>
<td></td>
<td>0.101</td>
<td>0.122</td>
<td>0.0123</td>
<td>0.54</td>
<td>0.067</td>
<td>0.067</td>
<td>0.152</td>
<td>0.059</td>
<td>0.248</td>
</tr>
<tr>
<td></td>
<td>1.512</td>
<td>2.463</td>
<td>0.33</td>
<td>0.496</td>
<td>0.054</td>
<td>0.054</td>
<td>0.41</td>
<td>0</td>
<td>1.080</td>
</tr>
<tr>
<td></td>
<td>3.447</td>
<td>7.162</td>
<td>3.415</td>
<td>6.012</td>
<td>0.158</td>
<td>0.149</td>
<td>1.15</td>
<td>0.112</td>
<td>1.800</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>133</td>
<td>80</td>
<td>111</td>
<td>24</td>
<td>20</td>
<td>40</td>
<td>34</td>
<td>70</td>
</tr>
</tbody>
</table>

Compression and seizure prediction: consume the most power

Power Consume of each pipeline under 15 mW

In general: higher operating frequency => higher dynamic power
Optimization of PE: Impact

For XCOR (left diagram):

- Before optimization: over 15 mW
- After: Spatial reprogramming saves 50% power

For LZMA (right diagram):

- Spatial reprogramming saves 1.5x power
- Locality refactoring: reduces power further to 11.2 mW
Conclusion

HARDWARE-SOFTWARE CO-DESIGN concept:

- Provides idea of refactoring the underlying algorithm of each task and implement each piece into a PE
- Provides idea of optimizing each PE in software level (spatial programming by XCOR PE)

HALO meets the safety constraint by realizing each task (under 15mW)

- Run each PE at minimum clock frequency catered to its need while ASICs run all logic at same frequency=> saves power
- Optimize each PE separately

HALO realizes many tasks (general-purpose architecture) , not a specific one like in ASIC

- For each task, controller configures required PEs into pipeline to execute it

HALO is extensible

- While realizing new BCI tasks: refactor algorithm, design PE for each piece, add new PEs into existing HALO architecture
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• **Strengths**
• Weaknesses
• Discussion
Strengths

Design:

• Great improvement with respect to flexibility => wider BCI adoption
• Low power consumption => safe for chronic use
• Extensible (benefit from its modularity) => can support new tasks by adding PEs into current HALO directly

Paper:

• Can understand the paper without prior background knowledge about BCI
• Highlight the advantage of HALO against BCIs as ASIC through comparison
• Basic structure and concept of HALO well explained
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
Weaknesses

Paper:

• Problems /constraint that HALO meets: not mentioned in the paper => discussion point

• Lack of performance evaluation with respect to processing speed (time to execute a task)
  - Tables focus on evaluation of power consumption and achieved flexibility
Outline

• Background
• Problems
• Goals
• Key Mechanism
• Implementation
• Evaluation
• Strengths
• Weaknesses
• Discussion
About HALO:

In the paper, author only claims: “HALO meets a different set of constraints”, but doesn’t explain it explicitly.

Constraints that HALO meets / Problems of HALO/ Places that still need to be improved?
Discussion

About HALO:

In the paper, author only claims: “HALO meets a different set of constraints”, but doesn’t explain it explicitly.

Constraints that HALO meets / Problems of HALO/ Places that still need to be improved?

- **Power constraint** is still a great limitation by designing since HALO belongs to implantable BCI (Key problem of implantable BCI)
  

- Need **finer grained** design than monolithic ASICs

- Does not solve the bottleneck problem of application of BCI in “From computer to brain” direction
About HALO:

In the paper, author only claims: “HALO meets a different set of constraints”, but doesn’t explain it explicitly.

**Constraints that HALO meets / Problems of HALO/ Places that still need to be improved?**

- **Power constraint** is still a great limitation by designing since HALO belongs to implantable BCI (Key problem of implantable BCI)
  

- Need **finer grained** design than monolithic ASICs

- Does not solve the bottleneck problem of application of BCI in “From computer to brain” direction

About BCI in general:

**Future prospects of brain-computer interface?**

- Fields in which BCI can be appilcated?

- As headsets vs. implantable (as chips embedded on brain): tradeoff?

- Problems/Difficulties that BCI will meet?
Thank you for Listening and Participating