A 40×40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control

Luke R. Everson, Sachin S. Sapatnekar, Chris H. Kim
University of Minnesota, Minneapolis, MN
Outline

• Background: Path Planning Algorithms
• Time-Based A* ASIC
  – Time-Based Primer
  – Top Level Design
  – Vertex Details
  – Edge Details
• 65nm Test Chip Results
• Applications
• Conclusion
Graph Definition

- Graph: set of “objects” and their “connections”
- Formal definition:
  - $G = (V, E)$, $V = \{v_1, v_2, ..., v_n\}$, $E = \{e_1, e_2, ..., e_m\}$
  - $V$: set of vertices (nodes), $E$: set of edges (links, arcs)
  - Directed graph: $e_k = (v_i, v_j)$
  - Undirected graph: $e_k = \{v_i, v_j\}$
  - Weighted graph: $w: E \rightarrow \mathbb{R}$, $w(e_k)$ is the “weight” of $e_k$. 
Why Graphs?

- **Network Analysis**
- **Routing/Path Planning**
- **Self-Driving Cars**

[Wikipedia.org]
[yuchisia.github.io]
[techrepublic.com]
Graph Algorithms - Breadth First Search

- Explores search area equally in all directions
- Adds new neighbors to queue and visits in order added
- Simple to implement in software
Graph Algorithms- Breadth First Search

• Explore entire graph from Source (S)
  • Visited = {S}
  • Queue = {A,D}
Graph Algorithms- Breadth First Search

• Explore entire graph from Source (S)
  • Visited = \{S,A\}
  • Queue = \{D,B\}
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
  - Visited = \{S,A,D\}
  - Queue = \{B,E\}
Graph Algorithms - Breadth First Search

- Explore entire graph from Source (S)
  - Visited = \{S,A,D,B\}
  - Queue = \{E,C\}
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = \{S,A,D,B,E\}
- Queue = \{C,T\}
Graph Algorithms - Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S, A, D, B, E, C}
- Queue = {T}
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = \{S,A,D,B,E,C,T\}
- Queue = {}
Graph Algorithms- Dijkstra’s

- Similar to BFS
- Uniform cost search when edges have weights
- Search “cheapest” neighbor first
- Keeps track of distance FROM start

BFS

Dijkstra’s

cost=1

cost=5

[Patel, RBG]
Graph Algorithms - Dijkstra’s

- Find shortest path from Source (S) to Target (T)
  - Visited={S₀}
  - Cost=0
  - PriorityQueue={A₁, D₁}

Store Parent Node for traceback
Graph Algorithms- Dijkstra’s

• Find shortest path from Source (S) to Target (T)
  • Visited={S₀,A₁,S₁}
  • Cost=1
  • PriorityQueue={D₁,B₂,A₂}

![Graph Diagram]
Graph Algorithms- Dijkstra’s

• Find shortest path from Source (S) to Target (T)
  • Visited={S₀,Aₛ₁,Dₛ₁}
  • Cost=1
  • PriorityQueue={Bₐ₂,Eₐ₄}
Graph Algorithms- Dijkstra’s

- Find shortest path from Source (S) to Target (T)
- Visited\(\{S_0, A_{S1}, D_{S1}, B_{A2}\}\)
- Cost=2
- PriorityQueue\(\{C_{B3}, E_{D4}\}\)
Graph Algorithms- Dijkstra’s

- Find shortest path from Source (S) to Target (T)
- Visited={S₀,A₁,D₁,B₂,C₃,T₄}
- Cost=3
- PriorityQueue={T₄,E₄}

Diagram:

- S₀ → A₁ → B₂ → C₃ → T₄
- S₀ → D₁ → E₄
- C₃ → T₄
- A₁ → B₂
- B₂ → C₃
- C₃ → T₄
- D₁ → E₄
- E₄ → T₄

Weights:
- S₀ to A₁: 1
- A₁ to B₂: 1
- B₂ to C₃: 1
- C₃ to T₄: 1
- S₀ to D₁: 1
- D₁ to E₄: 3
- E₄ to T₄: 1
Graph Algorithms- Dijkstra’s

• Find shortest path from Source (S) to Target (T)
• Visited={S₀, A₁, D₁, B₂, C₃, T₄}
• Cost=3
• PriorityQueue={E₄₄}
Graph Algorithms- A*

• A* guides the search towards the destination

• Cost(n) = F(n) + H(n)
  – F(n) = actual distance from source
  – H(n) = heuristic that predicts the distance to target
  – H(n) = 0 for Dijkstra’s

• Provides optimal path if H(n) ≤ Actual Cost
Graph Algorithms - A*

- Let $H(n) = \text{Manhattan Distance}$
- $H(n) = \abs{x_i - x_T} + \abs{y_i - y_T}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>T</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph Algorithms - A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- $\text{Visited} = \{S_0\}$
- Cost/F(n) = 0
- $\text{PriorityQueue} = \{A_{S_1}\}$
Graph Algorithms - A*

- Find path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- $\text{Visited} = \{S_0\}$
- Cost/F(n) = 0
- $\text{PriorityQueue} = \{A_{S_1}^5\}$

Cost $C(A) = F(A) + H(A) = 1 + 4$
Graph Algorithms - A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- $\text{Visited} = \{S_0\}$
- $\text{Cost/F}(n) = 0$
- $\text{PriorityQueue} = \{A_{S_1}^5, D_{S_1}^5\}$
Graph Algorithms - A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) =$ Manhattan Distance
- $\text{Visited} = \{S_0, A_{S1}^5\}$
- Cost/F(n) = 1
- $\text{PriorityQueue} = \{B_{A2}^5, D_{S1}^5\}$
Graph Algorithms - A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- $\text{Visited} = \{S_0, A_{S1}^5, D_{S1}^5\}$
- $\text{Cost/F(n)} = 1$
- $\text{PriorityQueue} = \{B_{A2}^5, E_{D4}^6\}$
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited=$\{S_0,A_{S1}^5,D_{S1}^5,B_{A2}^5\}$
- Cost/F(n)=2
- PriorityQueue=$\{C_{B3}^5,E_{D4}^6\}$
Graph Algorithms - A*

• Find shortest path from Source (S) to Target (T)
• Let $H(n) = \text{Manhattan Distance}$
• Visited=$\{S_0, A_{S1}^5, D_{S1}^5, B_{A2}^5, C_{B3}^5\}$
• Cost/F(n)=3
• PriorityQueue=$\{T_{c4}^4, E_{D4}^6\}$
Graph Algorithms - A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = $ Manhattan Distance
- $Visited = \{S_0, A_{S_1}^5, D_{S_1}^5, B_{A_2}^5, C_{B_3}^5, T_{C_4}^4\}$
- Cost/ $F(n) = 3$
- PriorityQueue = $\{E_{D_4}^6\}$
Algorithm Review

**BFS**
- Search entire graph
- No edge weights

**Dijkstra’s**
- Search best first
- Finds optimal path

**A***
- Search best heuristic first
- Dijkstra’s if no heuristic
- Optimal path if $H(n) \leq$ actual

---

© 2019 IEEE International Solid-State Circuits Conference

2.5: A 40x40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control
Graph Structure - This Work

- 4-neighbor grid
- Directed edges
- Each direction binary on/off control
- 4 bit digital control edge weight
- Programmable at runtime

On/Off

\[ w_{fc} \]

\[ w_{cf} = 4b \]
Outline

• Background: Path Planning Algorithms

• Time-Based A* ASIC
  – Time-Based Primer
  – Top Level Design
  – Vertex Details
  – Edge Details

• 65nm Test Chip Results
• Applications
• Conclusion
Advantages of digital arithmetic:
- Binary representation
- Less “buy-in” required
- Existing IP for rapid SoC development
- No calibration

Advantages of time-based circuits:
- Compact area
- Low power consumption
- High precision tunability
40×40 A* ASIC

- Time $\propto$ Distance
- Each vertex stores first input
- Directly readout shortest path
- Multiple start points

X-Dir Gradient

Y-Dir Gradient

40x40 4-Neighbor Array

Vertex Cell

Edge Cell

X, Y Gradient Bias

12b SRAM

Detect/Decode

Latch

4b SRAM

4b SRAM
2.5: A 40×40 ×××× Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control

© 2019 IEEE International Solid-State Circuits Conference

40×40 A* ASIC- Vertex

Pre-Input State

Post-Input State

© 2019 IEEE International Solid-State Circuits Conference

2.5: A 40×40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control
40×40 A* ASIC - Vertex
40×40 A* ASIC - Edge
40×40 A* ASIC - Gradient
Outline

• Background: Path Planning Algorithms
• Time-Based A* ASIC
• 65nm Test Chip Results
• Applications
• Conclusion
Test Setup

Custom Test PCB
- COTS DIO Card
- Chip Socket
- Supply Filters

Test Environment
- DIO
- Power & Bias Supply
- Test PCB
Edge Delay Linearity

![Graph showing edge delay linearity](image)

- **Delay, [ns]**
  - 0.75 V
  - 0.80 V
  - 0.90 V

- **65nmLP, 1.2V, 25°C**

- **INL, [TSTEP]**
  - -2
  - -1
  - 0
  - 1
  - 2

- **DNL, [TSTEP]**
  - -2
  - -1
  - 0
  - 1
  - 2

- **Delay, [ns]**
  - 1.6
  - 1.8
  - 2.0
  - 2.2
  - 2.4

---

© 2019 IEEE International Solid-State Circuits Conference
## Comparison Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>ASIC</td>
<td>Xilinx Virtex</td>
<td>ARM Cortex-M0</td>
<td>Intel Xeon E5630</td>
<td>NVIDIA Tesla K20c</td>
</tr>
<tr>
<td>Technology</td>
<td>65nm</td>
<td>20nm</td>
<td>40nm</td>
<td>32nm</td>
<td>28nm</td>
</tr>
<tr>
<td>Voltage</td>
<td>1.2V</td>
<td>-</td>
<td>1.1V</td>
<td>0.7-1.35V</td>
<td>-</td>
</tr>
<tr>
<td>Peak Power</td>
<td>26.4mW</td>
<td>24.22W</td>
<td>127μW</td>
<td>20W/core</td>
<td>225W</td>
</tr>
<tr>
<td>Throughput [MTEPS]</td>
<td>559</td>
<td>731</td>
<td>5.34x10^{-4}</td>
<td>0.83</td>
<td>9.0</td>
</tr>
<tr>
<td>Energy per Node</td>
<td>0.328pJ^*</td>
<td>33nJ</td>
<td>89.1nJ</td>
<td>24.1µJ</td>
<td>25µJ</td>
</tr>
<tr>
<td>Normalized Energy</td>
<td>1x</td>
<td>10^5</td>
<td>2.7x10^5</td>
<td>1.19x10^6</td>
<td>2.3x10^7</td>
</tr>
</tbody>
</table>

*55% from SRAM Program (does not include cache access energy)
Energy/Node=Unit Delay*Unit Power
MTEPS = Million Traversed Edges Per Second

Outline

• Background: Path Planning Algorithms
• Time-Based A* ASIC
• 65nm Test Chip Results
• Applications
• Conclusion
Voronoi Diagrams

• Segmentation of plane such that distance to seeds is maximized

• Applications:
  – kNN classification
  – Biological structures (bone and cells)
  – Computational fluid dynamics meshes
  – Autonomous vehicles

[Wikipedia]
Collision Avoidance

Binary Encoding

Readout Colormap Key

- 1101 = 13
- 1011 = 11
- 1110 = 14
- 0111 = 7
- 1001 = 9
Collision Avoidance
Collision Avoidance

![Diagram of collision avoidance](image)

**Readout Colormap Key**

<table>
<thead>
<tr>
<th>Binary Encoding</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
</tbody>
</table>

© 2019 IEEE International Solid-State Circuits Conference
Path Planning - Dijkstra’s

All diagonal paths are equal
Path Planning - Dijkstra’s

Paths above blockage dominate paths below
Path Planning – Gradient A*

Gradient improves dominant path
Path Planning

Finding shortest path w/o gradient

0.9V ----------- 0.9V

Finding shortest path w/ gradient

0.7V ----------- 0.9V
Path Planning

Finding shortest path w/o gradient

Finding shortest path w/ gradient

0.9V

0.7V

0.9V

Olive indicates FASTER Path
Multicore Evaluation (1/5)

Global Map
Multicore Evaluation (2/5)

Global Map

2.5: A 40×40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control
Multicore Evaluation (3/5)

Global Map

© 2019 IEEE International Solid-State Circuits Conference
Multicore Evaluation (4/5)
Multicore Evaluation (5/5)
Optics Experiment
Optics Experiment
Outline

- Background: Path Planning Algorithms
- Time-Based A* ASIC
- 65nm Test Chip Results
- Applications
- Conclusion
Die Photo and Chip Summary

<table>
<thead>
<tr>
<th>Applications</th>
<th>A* shortest path, obstacle avoidance, scientific computation (optics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>65nm LP CMOS</td>
</tr>
<tr>
<td>Architecture</td>
<td>Time-based</td>
</tr>
<tr>
<td># of Vertices</td>
<td>1600</td>
</tr>
<tr>
<td>Unit Area</td>
<td>249μm²</td>
</tr>
<tr>
<td># of Edges</td>
<td>6400</td>
</tr>
<tr>
<td>Edge Resolution</td>
<td>4b + Analog Gradient</td>
</tr>
<tr>
<td>Voltage</td>
<td>1.2V</td>
</tr>
<tr>
<td>Peak Power</td>
<td>26.4mW</td>
</tr>
<tr>
<td>Delay per Node</td>
<td>1.79ns @ [V_B=9V, V_DD=1.2V]</td>
</tr>
<tr>
<td>Power per Node</td>
<td>183.1μW</td>
</tr>
<tr>
<td>Energy per Node</td>
<td>0.238pJ</td>
</tr>
</tbody>
</table>
Conclusion

• 40×40 A* ASIC in 65nmLP
• A* heuristic implemented with analog bias gradient
• Vertex cells asynchronously evaluate and lockout
• Diverse range of applications
• Scalable to multicore for larger maps
• Orders of magnitude improvement in energy efficiency

This research was supported in part by the National Science Foundation under award number CCF-1763761.
2.5: A 40 × × × × 40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control