

Exhibit C

Rex Computing v. Cerebras Systems - Claim Charts - US 10,355,975

Cerebras CS-1 and CS-2¹

Claim 1

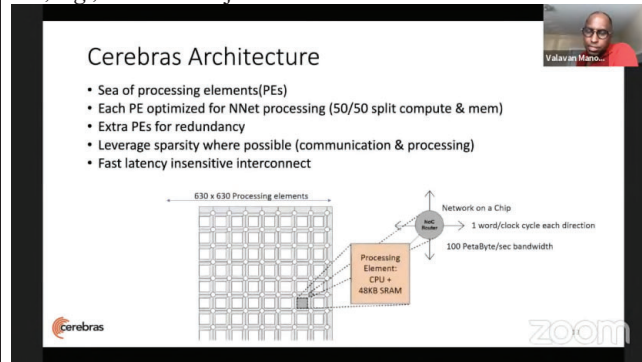
<p>[1.0] A system comprising:</p>	<p>To the extent the preamble is limiting, the Cerebras CS-1 and CS-2 include a system.</p> <p><i>See, e.g.</i>, CS-1 Overview² at 2 (“Cluster-scale deep learning compute in a single system”).</p> <p><i>See, e.g.</i>, CS-1 Overview at 2 (“The CS-1 is a system solution that consists of innovations across three dimensions: a) the Cerebras Wafer Scale Engine (WSE) — the industry’s largest and only trillion-transistor processor, b) the Cerebras System and c) the Cerebras software platform.”).</p> <p><i>See, e.g.</i>, CS-1 Overview at 12 (“The Wafer Scale Engine, the CS-1 system, and the Cerebras software platform together comprise a complete solution to high-performance deep learning compute.”).</p> <p><i>See, e.g.</i>, WP03 at 2 (“The CS-2 is a system solution that consists of innovations across three dimensions: a) the second generation Cerebras Wafer Scale Engine (WSE-2) — the industry’s largest and only multi-trillion-transistor processor, b) the Cerebras System and c) the Cerebras software platform.”)</p> <p><i>See, e.g.</i>, Rocki at 2 (“We describe the system, the processing core architecture, the programming model, and our implementation of BiCGStab in that model</p>
-----------------------------------	--

¹ The Cerebras CS-1 and CS-2 both admittedly utilize the Cerebras Wafer Scale Engine and include the Cerebras Swarm communication fabric (D.I. 10, ¶ 68), which together provide an on-chip network having a set of processor cores and corresponding routers arranged and configured in accordance with the limitations recited in the Asserted Claims as set forth herein. *See, e.g.*, CS-1 Overview at 2-4; WP03 at 2-4. As such, and unless otherwise stated, the documents cited within these claim charts for CS-1 apply equally to CS-2.

² Unless otherwise provided within the citation itself, the full citation to each short form reference throughout the chart can be found at the end of the chart under the heading “References.”

	<p>...</p> <p>The CS-1 wafer is an MIMD, distributed-memory machine with a 2D-mesh interconnection fabric. The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles. Figure 2 illustrates the layout.”).</p>
[1.1] a set of processor cores;	<p>The Cerebras CS-1 and CS-2 include a set of processor cores.</p> <p><i>See, e.g.,</i> Groeneveld EDPS Pres. at 14:55 (“processing elements”).</p> <p><i>See, e.g.,</i> Groeneveld EDPS 2020 at 13.</p> <p style="text-align: center;">Wafer Scale Engine: General Arrangement</p>

See, e.g., Manohararajah Pres. at 50:45



See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 3

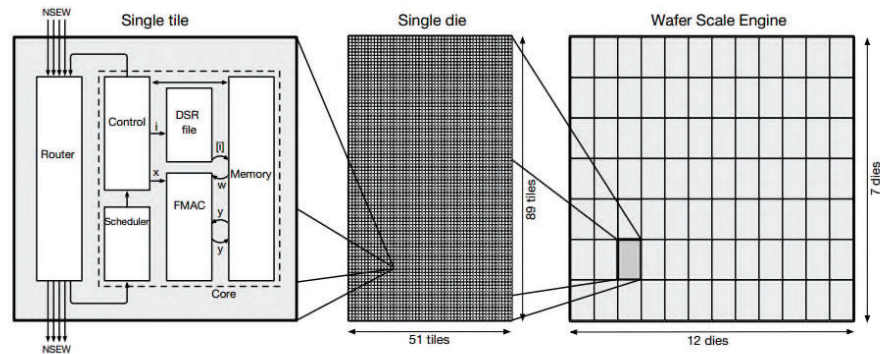


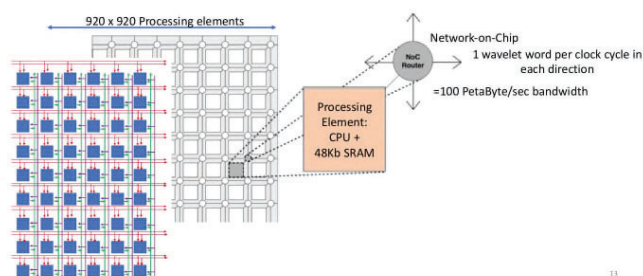
Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).

	<p><i>See, e.g.</i>, CRBR000575 at CRBR000578 [REDACTED]</p> <p><i>See, e.g.</i>, CRBR000575 at CRBR000580:</p> <p>[REDACTED]</p>
<p>[1.2] a set of routers each including a set of input ports and a set of output ports, wherein:</p>	<p>The Cerebras CS-1 and CS-2 include a set of a set of routers each including a set of input ports and a set of output ports.</p> <p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 14:55 (“NoC Router”).</p>

See, e.g., Groeneveld EDPS 2020 at 13.

Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect

The diagram shows a 630 x 630 grid of Processing Elements. A callout box details a single Processing Element, which contains a CPU and 48KB SRAM. This element is connected to a Network on a Chip (NoC) Router. The NoC Router supports 1 word/clock cycle in each direction, providing a bandwidth of 100 PetaByte/sec.

Valavan Manohararajah

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 3

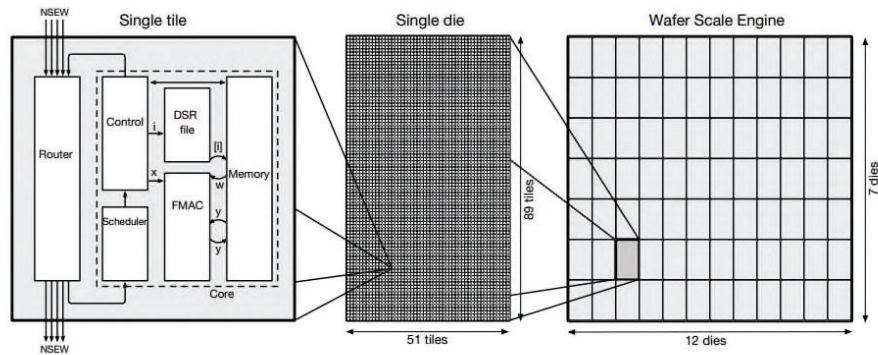


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

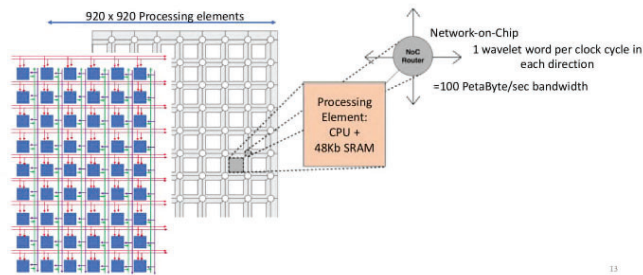
See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or

	<p>usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).</p> <p><i>See, e.g.,</i> US 2020/0005142, ¶ [0499] (“FIG. 5 illustrates selected details of an embodiment of a PE as PE 500 of a deep learning accelerator. PE 500 comprises Router 510 and Compute Element 520. Router 510 selectively and/or conditionally communicates wavelets between other PEs (e.g., logically adjacent and/or physically adjacent PEs) and the instant PE via couplings 511-516. Router 510 selectively and/or conditionally communicates wavelets to the instant PE via Off Ramp 521 and communicates wavelets from the instant PE via On Ramp 522. Compute Element 520 performs computations on data embodied in the wavelets according to instruction address information derivable from the wavelets. The instruction address information is used to identify starting addresses of tasks embodied as instructions stored in memory of the compute element.”).</p> <p><i>See, e.g.,</i> CRBR000274.</p> <p><i>See, e.g.,</i> CRBR000298–300.</p> <p><i>See, e.g.,</i> CRBR000575 at CRBR000580 [REDACTED]</p>
<p>[1.3] each processor core of the set of processor cores corresponds to a different router of the set of routers, wherein</p>	<p>In Cerebras CS-1 and CS-2, each processor core of the set of processor cores corresponds to a different router of the set of routers.</p> <p><i>See, e.g.,</i> Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That’s how every processing element looks like. They’re all identical.”).</p>

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



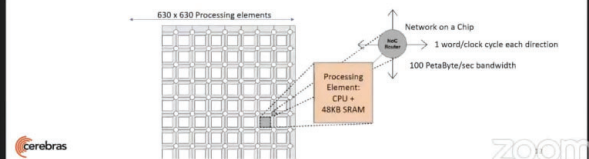
See, e.g., Manohararajah Pres. at 50:45



Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect



Valaraj Manohararajah



See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 3

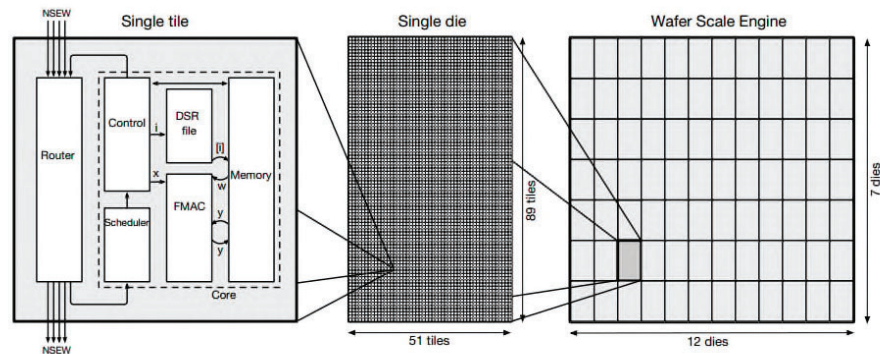


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

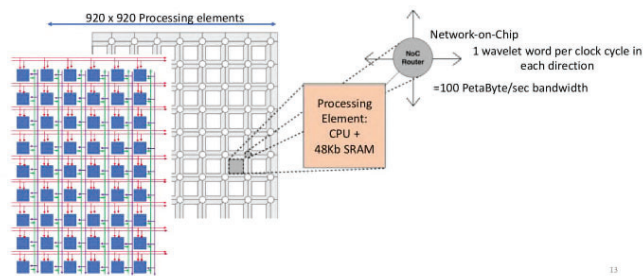
See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE

	<p>499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).</p> <p><i>See, e.g.</i>, US 2020/0005142, ¶ [0499] (“FIG. 5 illustrates selected details of an embodiment of a PE as PE 500 of a deep learning accelerator. PE 500 comprises Router 510 and Compute Element 520. Router 510 selectively and/or conditionally communicates wavelets between other PEs (e.g., logically adjacent and/or physically adjacent PEs) and the instant PE via couplings 511-516. Router 510 selectively and/or conditionally communicates wavelets to the instant PE via Off Ramp 521 and communicates wavelets from the instant PE via On Ramp 522. Compute Element 520 performs computations on data embodied in the wavelets according to instruction address information derivable from the wavelets. The instruction address information is used to identify starting addresses of tasks embodied as instructions stored in memory of the compute element.”).</p> <p><i>See, e.g.</i>, CRBR000298–300.</p> <p><i>See, e.g.</i>, CRBR000575 at CRBR000578 [REDACTED]</p>
[1.4] each processor core and corresponding router form a tile,	<p>In the Cerebras CS-1 and CS-2, each processor core and corresponding router form a tile.</p> <p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor,</p>

your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That's how every processing element looks like. They're all identical.")

See, e.g., Groeneveld EDPS 2020 at 13.

Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect

Valayam Manoj

The diagram shows a 630 x 630 grid of Processing elements. A callout for a single Processing Element indicates it contains a CPU and 48KB SRAM. The network is labeled 'Network on a Chip' with '1 word/clock cycle each direction' and '100 PetaByte/sec bandwidth'. The Cerebras logo is in the bottom left, and a 'zoom' watermark is in the bottom right.

See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 3

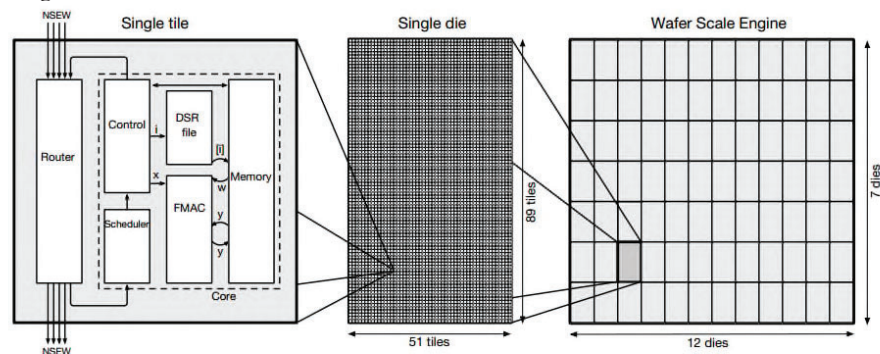


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

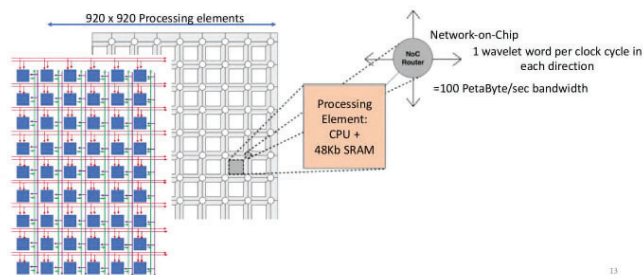
See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are

	<p>illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).</p> <p><i>See, e.g.</i>, CRBR000298-300.</p> <p><i>See, e.g.</i>, CRBR000575 at CRBR000578 [REDACTED] [REDACTED]</p> <p><i>See, e.g.</i>, CRBR000706 at CRBR000715 [REDACTED] [REDACTED]</p> <p><i>See, e.g.</i>, CRBR000995 at CRBR000997 [REDACTED] [REDACTED]</p>
[1.5] each processor core is communicatively coupled with a corresponding router via the router's set of input ports and set of output ports,	<p>In the Cerebras CS-1 and CS-2, each processor core is communicatively coupled with a corresponding router via the router's set of input ports and set of output ports.</p> <p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or</p>

inject it into the network. That's how every processing element looks like. They're all identical.")

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



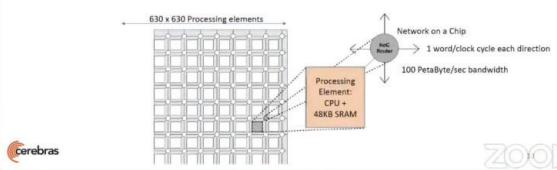
See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect





Valavan Manohararajah



Network on a Chip

1 word/clock cycle each direction

100 PetaByte/sec bandwidth

See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 3

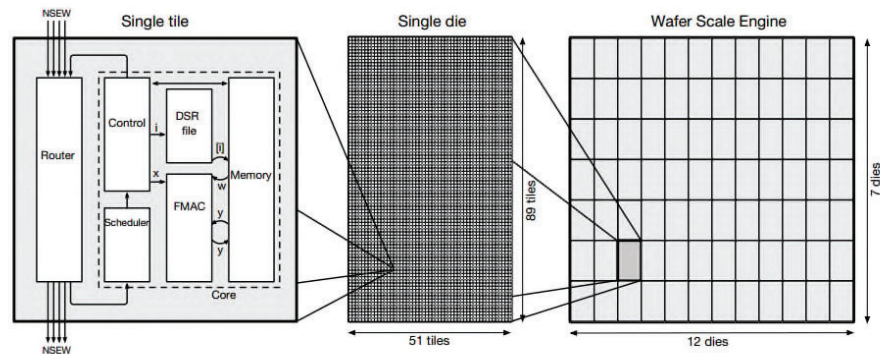


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE

499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).

See, e.g., US 2020/0005142, ¶ [0499] (“FIG. 5 illustrates selected details of an embodiment of a PE as PE 500 of a deep learning accelerator. PE 500 comprises Router 510 and Compute Element 520. Router 510 selectively and/or conditionally communicates wavelets between other PEs (e.g., logically adjacent and/or physically adjacent PEs) and the instant PE via couplings 511-516. Router 510 selectively and/or conditionally communicates wavelets to the instant PE via Off Ramp 521 and communicates wavelets from the instant PE via On Ramp 522. Compute Element 520 performs computations on data embodied in the wavelets according to instruction address information derivable from the wavelets. The instruction address information is used to identify starting addresses of tasks embodied as instructions stored in memory of the compute element.”).

See, e.g., CRBR000306.

See, e.g., CRBR000308.

See, e.g., CRBR000995 at CRBR000997 [REDACTED]

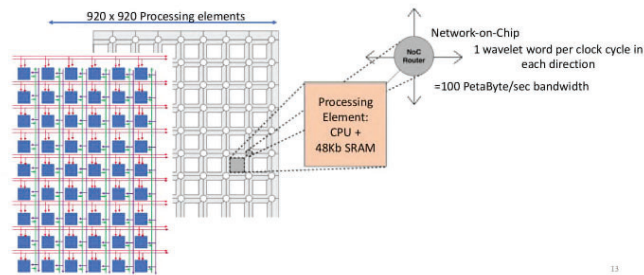
[1.6] each router is communicatively coupled with one or more adjacent routers via the set of input ports and the set of output ports, and wherein

In the Cerebras CS-1 and CS-2, each router is communicatively coupled with one or more adjacent routers via the set of input ports and the set of output ports.

See, e.g., Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That’s how every processing element looks like. They’re all identical.”)

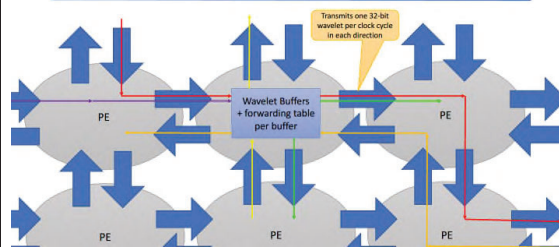
See, e.g., Groeneveld EDPS 2020 at 13.

Wafer Scale Engine: General Arrangement



See, e.g., Groeneveld EDPS 2020 at 14.

Wavelets Flowing on the Network-on-Chip



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect

zoom

See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

See, e.g., Rocki at 2 (“The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles.”).

See, e.g., Rocki at 2 (“The CS-1 wafer is an MIMD, distributed-memory machine with a 2D-mesh interconnection fabric. The repeated element of the architecture is called a tile. The tile contains one processor core, its memory, and the router that it connects to. The routers link to the routers of the four neighboring tiles. Figure 2 illustrates the layout.”).

See, e.g., Rocki at 3

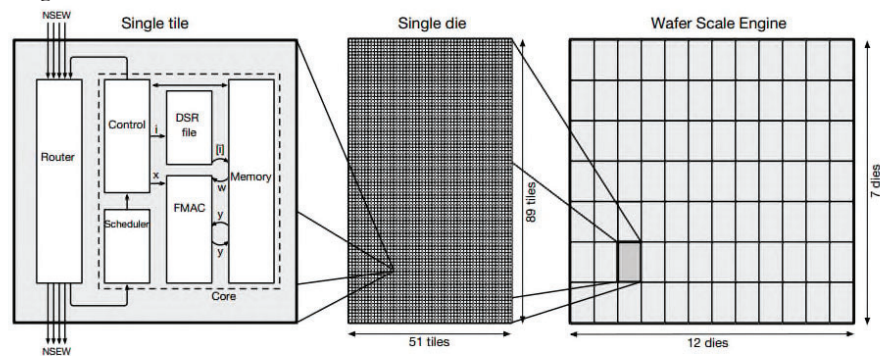


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

See, e.g., Rocki at 7

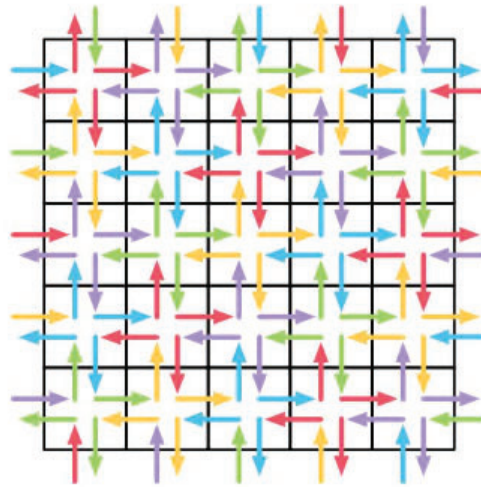


Fig. 5. Tessellation routing pattern for SpMV: a single core pushes its content into adjacent cores' fabric router using a single communication channel. Messages from the four neighbors arrive on four distinct channels and are processed by corresponding threads. This allows us to achieve high fabric utilization due to the fact that we can send the data in 4 directions in a single cycle. WSE allows the fabric to be dynamically reconfigured. Such adaptive topology plays a significant role in offloading routing logic from cores, which can be used primarily for computation.

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE

499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).

See, e.g., US 2020/0005142, ¶ [0499] (“FIG. 5 illustrates selected details of an embodiment of a PE as PE 500 of a deep learning accelerator. PE 500 comprises Router 510 and Compute Element 520. Router 510 selectively and/or conditionally communicates wavelets between other PEs (e.g., logically adjacent and/or physically adjacent PEs) and the instant PE via couplings 511-516. Router 510 selectively and/or conditionally communicates wavelets to the instant PE via Off Ramp 521 and communicates wavelets from the instant PE via On Ramp 522. Compute Element 520 performs computations on data embodied in the wavelets according to instruction address information derivable from the wavelets. The instruction address information is used to identify starting addresses of tasks embodied as instructions stored in memory of the compute element.”).

See, e.g., CRBR000253–254.

See, e.g., CRBR000575 at CRBR000580



	<p><i>See, e.g.</i>, CRBR000706 at CRBR000720 [REDACTED]</p> <p><i>See, e.g.</i>, CRBR000995 at CRBR000997 [REDACTED]</p>
[1.7] each processor core is communicatively coupled with the other processor cores via the set of routers,	<p>In the Cerebras CS-1 and CS-2, each processor core is communicatively coupled with the other processor cores via the set of routers.</p> <p><i>See, e.g.</i>, Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel, on every cycle. Even with scalar granularity, communication is efficient”).</p> <p><i>See, e.g.</i>, Rocki at 6</p>

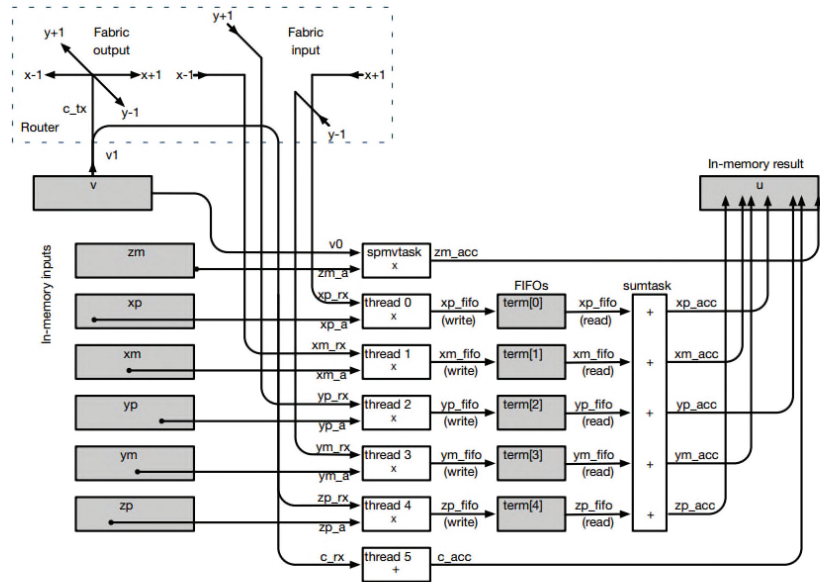


Fig. 4. Implementation of SpMV ($u = Av$). Shaded regions represent memory objects, annotated arrows are tensor descriptors, and white boxes are tasks that perform computations. The diagram uses the names of objects in the code of Listing 1.

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets,

backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).

See, e.g., CRBR000575 at CRBR000580:



See, e.g., CRBR000706 at CRBR000718

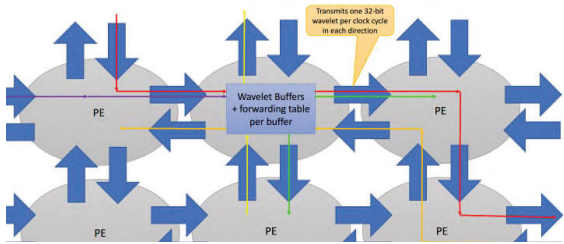


See, e.g., CRBR000706 at CRBR000720



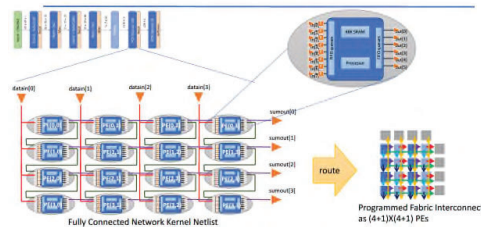
See, e.g., CRBR000995 at CRBR000997



<p>[1.8] each router is operable to receive one or more data packets from the one or more adjacent routers or the processor core corresponding to the router, based on a physical destination address of a data packet, each router is operable to send one or more data packets to the one or more adjacent routers or the processor core corresponding to the router, wherein</p>	<p>In the Cerebras CS-1 and CS-2, each router is operable to receive one or more data packets from the one or more adjacent routers or the processor core corresponding to the router, based on a physical destination address of a data packet, each router is operable to send one or more data packets to the one or more adjacent routers or the processor core corresponding to the router.</p> <p><i>See</i> claim elements [1.5], [1.6], and [1.7], <i>supra</i>.</p> <p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 15:36 (“data packet or a ‘wavelet’”).</p> <p><i>See, e.g.</i>, Groeneveld EDPS 2020 at 14.</p> <p style="text-align: center;">Wavelets Flowing on the Network-on-Chip</p> 
---	--

See, e.g., Groeneveld EDPS 2020 at 15.

Netlist and Routing of Processor Elements



See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).

	<p><i>See, e.g.</i>, US 2020/0005142, ¶ [0499] (“FIG. 5 illustrates selected details of an embodiment of a PE as PE 500 of a deep learning accelerator. PE 500 comprises Router 510 and Compute Element 520. Router 510 selectively and/or conditionally communicates wavelets between other PEs (e.g., logically adjacent and/or physically adjacent PEs) and the instant PE via couplings 511-516. Router 510 selectively and/or conditionally communicates wavelets to the instant PE via Off Ramp 521 and communicates wavelets from the instant PE via On Ramp 522. Compute Element 520 performs computations on data embodied in the wavelets according to instruction address information derivable from the wavelets. The instruction address information is used to identify starting addresses of tasks embodied as instructions stored in memory of the compute element.”).</p> <p><i>See, e.g.</i>, US 2020/0005142, ¶ [0502] (“FIG. 6 illustrates selected details of an embodiment a router of a PE, as Router 600. Consider that there are a plurality of PEs, each comprising a respective router and a respective CE. Router 600 is an instance of one of the respective routers. Router 600 routes wavelets, in accordance with color information of the wavelets and routing configuration information, to the CE of the PE that the instant router is comprised in, as well as others of the routers. The routed wavelets are variously received by the instant router and/or generated by the CE of the PE that the instant router is comprised in. The routing enables communication between the PEs. Stall information is communicated to prevent overflowing of wavelet storage resources in Router 600.”).</p> <p><i>See, e.g.</i>, US 2020/0005142, ¶ [0508] (“FIG. 7 illustrates selected details of an embodiment of processing associated with a router of a processing element, as Wavelet Ingress 710, Stall Info 720, and Wavelet Egress 730. Conceptually, the router accepts as many wavelets as possible from ingress ports, queuing as necessary and as queue space is available, and routes as many wavelets as possible to egress ports per unit time (e.g., clock cycle). Wavelet Ingress 710 comprises actions 711-713 corresponding to wavelet ingress from (logically and/or physically) adjacent PEs and/or an instant PE, for each respective queue. Stall Info 720 comprises actions 721-723 correspond to providing stall information, for each respective queue. Wavelet Egress 730 comprises actions 731-734 that correspond to wavelet egress to (logically and/or physically) adjacent PEs and/or the instant PE, for each respective queue. In some circumstances, in accordance with color information of a wavelet and routing configuration information, Send Wavelet 734 sends a wavelet from a single queue entry to a single destination (e.g., unicast). In some circumstances, in accordance with color information of a wavelet and routing configuration</p>
--	---

information, Send Wavelet 734 sends a wavelet from a single queue entry to a plurality of destinations (e.g., multicast). In various embodiments and/or usage scenarios, any one or more of all or any portions of actions of 710, 720, and/or 730 correspond to actions performed by and/or related to all or any portions of any one or more elements of Router 600 of FIG. 6.”).






See, e.g., CRBR000279.

See, e.g., CRBR000289.

See, e.g., CRBR000575 at CRBR000578

[REDACTED]

See, e.g., CRBR000575 at CRBR000585:

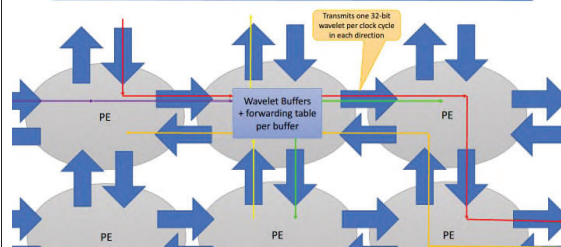
	<p data-bbox="581 573 1377 1136"></p> <p data-bbox="683 1129 748 1150">U – West</p> <p data-bbox="500 1182 1430 1241"><i>See, e.g.</i>, CRBR000706 at CRBR000720  </p> <p data-bbox="500 1266 1451 1497"><i>See, e.g.</i>, CRBR000706 at CRBR000758–59  </p>
--	--

[illegible]

	<p>[REDACTED]</p> <p><i>See, e.g.,</i> CRBR001587 at CRBR001587–90.</p> <p><i>See, e.g.,</i> CRBR001663 [REDACTED]</p> <p><i>To the extent that Cerebras contends this limitation is not literally met, the CS-1 and CS-2 also meet the limitation under the doctrine of equivalents. Each router in CS-1 and CS-2 is operable to perform the substantially the same function of sending one or more data packets to the one or more adjacent routers or the processor core corresponding to the router, in substantially the same way that is based on the physical destination address of the data packet, to achieve the same result of the limitation. Moreover, each router in CS-1 and CS-2 is not substantially different from the claimed limitation.</i></p>
[1.9] each router is operable to retain a data packet in the event of a traffic condition, and	<p>In the Cerebras CS-1 and CS-2, each router is operable to retain a data packet in the event of a traffic condition.</p> <p><i>See, e.g.,</i> Groeneveld EDPS Pres. at 15:36 (“Wavelet Buffers + forwarding table per buffer”).</p>

See, e.g., Groeneveld EDPS 2020 at 14.

Wavelets Flowing on the Network-on-Chip



See, e.g., Manohararajah Pres. at 52:42 (“[Q:] You sometimes overlay two traffic flows on the same links but you do that by scheduling them clock cycle by clock cycle as part of your compilation? [A:] Correct, uh no that’s done but that’s done by the hardware. That is not something that the software deals with. [Q:] So it’s still buffering and it still figures out when it can send it, the software just figures out it’s not going to over allocate it? [A:] Exactly, yeah.”).

See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel on every cycle.”).

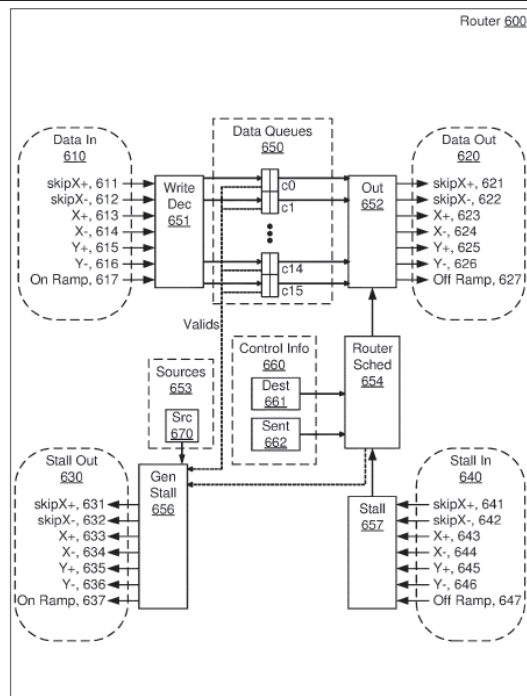
See, e.g., Rocki at 3 (“The router has hardware queues for its connection to the core and for each of a set of virtual channels, avoiding deadlock. Communication between potentially distant processors occurs along predetermined routes.”).

See, e.g., US 2020/0005142, ¶ [0502] (“FIG. 6 illustrates selected details of an embodiment a router of a PE, as Router 600. Consider that there are a plurality of PEs, each comprising a respective router and a respective CE. Router 600 is an instance of one of the respective routers. Router 600 routes wavelets, in accordance with color information of the wavelets and routing configuration information, to the CE of the PE that the instant router is comprised in, as well as others of the routers. The routed wavelets are variously received by the instant router and/or

generated by the CE of the PE that the instant router is comprised in. The routing enables communication between the PEs. Stall information is communicated to prevent overflowing of wavelet storage resources in Router 600.”).

See, e.g., US 2020/0005142, ¶ [0508] (“FIG. 7 illustrates selected details of an embodiment of processing associated with a router of a processing element, as Wavelet Ingress 710, Stall Info 720, and Wavelet Egress 730. Conceptually, the router accepts as many wavelets as possible from ingress ports, queuing as necessary and as queue space is available, and routes as many wavelets as possible to egress ports per unit time (e.g., clock cycle). Wavelet Ingress 710 comprises actions 711-713 corresponding to wavelet ingress from (logically and/or physically) adjacent PEs and/or an instant PE, for each respective queue. Stall Info 720 comprises actions 721-723 correspond to providing stall information, for each respective queue. Wavelet Egress 730 comprises actions 731-734 that correspond to wavelet egress to (logically and/or physically) adjacent PEs and/or the instant PE, for each respective queue. In some circumstances, in accordance with color information of a wavelet and routing configuration information, Send Wavelet 734 sends a wavelet from a single queue entry to a single destination (e.g., unicast). In some circumstances, in accordance with color information of a wavelet and routing configuration information, Send Wavelet 734 sends a wavelet from a single queue entry to a plurality of destinations (e.g., multicast). In various embodiments and/or usage scenarios, any one or more of all or any portions of actions of 710, 720, and/or 730 correspond to actions performed by and/or related to all or any portions of any one or more elements of Router 600 of FIG. 6.”).

See, e.g., US 2021/0256362, Fig. 6.



See, e.g., CRBR000706 at CRBR000723

See, e.g., CRBR000706 at CRBR000723

	<p><i>See, e.g.</i>, CRBR001477 at CRBR00148 [REDACTED]</p> <p><i>See, e.g.</i>, CRBR000575 at CRBR000589 [REDACTED]</p>
[1.10] each router implements a static priority routing policy; and	<p>In Cerebras CS-1 and CS-2, each router implements a static priority routing policy.</p> <p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 15:36 (“Programming these forwarding tables allows us to kind of program any possible routing forwarding in this, on our PEs.”).</p> <p><i>See, e.g.</i>, Manohararajah Pres. at 51:52 (“It’s statically configured. It’s a virtual channel. So there’s some number of virtual channels. Certain virtual channels can talk to certain other virtual channels. So there are some restrictions as to as to how the connectivity is achieved and so that’s one of the constraints within the router, so I have to adhere to that constraint”).</p> <p><i>See, e.g.</i>, Rocki at 3 (“Communication between potentially distant processors occurs along predetermined routes.”).</p> <p><i>See, e.g.</i>, US 10,762,418 at 36:9-21 (“There is a routing pattern associated with each color and implemented by the routers. The routing pattern of each pattern is programmable and in some embodiments is statically configured, e.g., based at least in part on determinations made by Placement Server(s) SW 210 and/or Neuron to PE Mapping SW 212 of FIG. 2. Once configured, e.g., under control of software (such as Connection Server(s) SW 220 of FIG. 2), each color is a fixed routing pattern. All data that flows within a color always flows in accordance with the fixed routing pattern. There are no dynamic routing decisions. The fixed routing matches neural network communication patterns where neuron connections are statically specified. The fixed routing enables relatively lower cost hardware implementation.”).</p>

See, e.g., US 2021/0256362, ¶ [0688] (“There is a routing pattern associated with each color and implemented by the routers . The routing pattern of each pattern is programmable and in some embodiments is statically configured , e.g. , based at least in part on determinations made by Placement Server(s) SW 210 and /or Neuron to PE Mapping SW 212 of FIG . 2. Once configured, e.g. , under control of software (such as Connection Server(s) SW 220 of FIG . 2) , each color is a fixed routing pattern. All data that flows within a color always flows in accordance with the fixed routing pattern . There are no dynamic routing decisions . The fixed routing matches neural network communication patterns where neuron connections are statically specified. The fixed routing enables relatively lower cost hardware implementation.”).

See, e.g., CRBR000706 at CRBR000721 [REDACTED]

See, e.g., CRBR000706 at CRBR00072 [REDACTED]


See, e.g., CRBR000706 at CRBR000758 [REDACTED]

See, e.g., CRBR000995 at CRBR000997 [REDACTED]

	<p>See, e.g., CRBR001508 [REDACTED]</p> <p>See, e.g., CRBR000706 at CRBR000759 [REDACTED]</p> <p>See, e.g., CRBR001578 [REDACTED]</p> <p><i>To the extent that Cerebras contends this limitation is not literally met, the CS-1 and CS-2 also meet the limitation under the doctrine of equivalents. Each router in CS-1 and CS-2 is capable of performing substantially the same function of implementing a static routing policy, in substantially the same way of the claimed static routing policy, to achieve the same result of sending and receiving data packets to and from adjacent routers. Moreover, each router in CS-1 and CS-2 is not substantially different from the claimed limitation.</i></p>
[1.11] an optimization module configured to: determine optimal function assignment configurations for groups of tiles, and	<p>The Cerebras CS-1 and CS-2 include an optimization module configured to determine optimal function (e.g., “kernel”) assignment configurations for groups of tiles.</p> <p>See e.g., Software Co-design at 8 [REDACTED]</p>

See e.g., Software Co-design at 8-11



	<div data-bbox="500 573 1146 1142" data-label="Image"></div> <p data-bbox="500 1150 1463 1207"><i>See, e.g.</i>, CS-1 Overview at 4 (“Cerebras’ software can configure all the cores and routers on the WSE to support a unique, optimized communication pattern for each particular neural network.”).</p> <p data-bbox="500 1234 1463 1434"><i>See e.g.</i>, CS-1 Overview at 8 (“To translate a deep learning network into an optimized executable, CGC . . . allocates compute and memory to each kernel in the graph and maps every kernel onto a physical region of the computational array of cores. . . . kernel placement is formulated as a multi-constraint problem on 1) memory capacity and bandwidth, 2) computation requirements, and 3) computation costs. The placement engine then takes into account both algorithmic efficiency and compute core utilization to generate a result that maximizes locality, minimizes routing distances, and avoids hotspots.”)</p>
--	---

See, e.g., Manohararajah Pres. at 17:09

Model Parallel, Within a Neural Network Layer

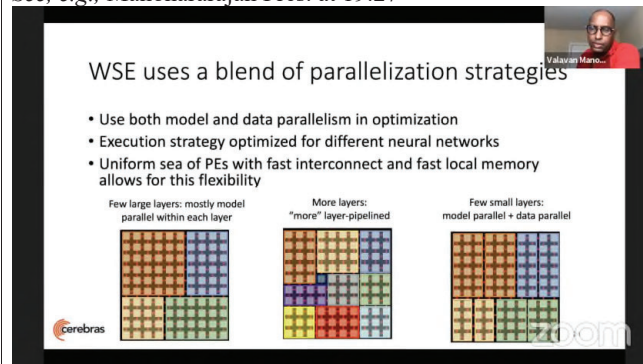
- Distribute execution of a single layer across multiple processing elements (PEs)
- Compiler chooses an optimal number of PEs and optimal shape for every layer
- Compute-heavy layers get larger PEs allocations

See, e.g., Manohararajah Pres. at 17:48

Model Parallel, Layer Pipelined

- Distribute execution of multiple layers across fabric section
- Compiler places layers on the fabric to optimize compute and communication
- Pipelined execution of model

See, e.g., Manohararajah Pres. at 19:27



See, e.g., Manohararajah Pres. at 19:41 (“The distribution of how each layer gets mapped into the PEs and how many PEs are allocated for each type of compute is all determined automatically by the compiler.”).

See, e.g., WP03 at 4 (“Computation inside the WSE-2 happens in 850,000 AI-optimized Sparse Linear Algebra Compute (SLAC) cores. The SLAC cores are designed for the sparse linear algebra primitives that underpin all neural network computation. This programmability ensures cores can run all neural network algorithms in the constantly changing machine learning field.”).

See, e.g., WP03 at 8 (“During this compilation process, kernel placement is formulated as a multi-constraint problem on 1) memory capacity and bandwidth, 2) computation requirements, and 3) communication costs.

The placement engine then takes into account both algorithmic efficiency and compute core utilization to generate a result that maximizes locality, minimizes routing distances, and avoids hotspots. The final result is a CS-2 executable, customized to the unique needs of each neural network, so that all 850,000 SLAC cores and 40 Gigabytes of on-chip SRAM can be used at maximum utilization towards accelerating the deep learning application”).

	<p><i>See, e.g.,</i> US 2020/0005142, ¶ [0487] (“FIG. 3 illustrates selected details of an embodiment of processing associated with training a neural network and performing inference using the trained neural network, using a deep learning accelerator, as Neural Network Training/Inference 300. As illustrated, neurons of the neural network are placed, e.g., allocated and/or associated with specific PE resources in action 310.”).</p> <p><i>See, e.g.,</i> US 2020/0005142, ¶ [0784] (“Conceptually, Deep Learning Accelerator 400 (FIG. 4) is a programmable compute fabric (see, e.g., FIGS. 5-8 and section “Processing Element: Compute Element and Router”). For example, the compute element of each PE 499 element is enabled to execute sequences of instructions of tasks (such as conceptually corresponding to all or any portions of executions of instructions of Task SW on PEs 260 of FIG. 2), and the router element of router element of each PE 499 is configurable to route wavelets between the PEs. The programmable compute fabric enables mapping of workloads onto the compute fabric in various manners.”).</p> <p><i>See, e.g.,</i> CRBR001477 at CRBR001494 [REDACTED]</p>
<p>[1.12] assign two or more functions, which communicate at least unilaterally more frequently with one another than with other functions, to groups of adjacent tiles based on an optimal function</p>	<p>The Cerebras CS-1 and CS-2 include an optimization module configured to assign two or more functions, which communicate at least unilaterally more frequently with one another than with other functions, to groups of adjacent tiles based on an optimal function assignment configuration determination.</p> <p><i>See</i> claim element [1.11], <i>supra</i>.</p>

assignment configuration determination, wherein	<p><i>See, e.g.</i>, Vassilieva Pres. at 31:28. (“We can maximize their communication between those fractions or those sections of the fabric which should talk to each other for this specific model.”)</p> <p><i>See, e.g.</i>, Vassilieva Pres. at 38:26 (“We configure the network fabric, so we route it to maximize the communication between those sections of the fabric which should talk to each other for this specific model.”).</p> <p><i>See, e.g.</i>, WP03 at 4 (“Computation inside the WSE-2 happens in 850,000 AI-optimized Sparse Linear Algebra Compute (SLAC) cores. The SLAC cores are designed for the sparse linear algebra primitives that underpin all neural network computation. This programmability ensures cores can run all neural network algorithms in the constantly changing machine learning field.”).</p> <p><i>See, e.g.</i>, WP03 at 8 (“During this compilation process, kernel placement is formulated as a multi-constraint problem on 1) memory capacity and bandwidth, 2) computation requirements, and 3) communication costs. The placement engine then takes into account both algorithmic efficiency and compute core utilization to generate a result that maximizes locality, minimizes routing distances, and avoids hotspots. The final result is a CS-2 executable, customized to the unique needs of each neural network, so that all 850,000 SLAC cores and 40 Gigabytes of on-chip SRAM can be used at maximum utilization towards accelerating the deep learning application”).</p> <p><i>See, e.g.</i>, CRBR001477 at CRBR001494 [REDACTED]</p>
[1.13] the two or more functions are assigned to groups	The Cerebras CS-1 and CS-2 include an optimization module configured to assign two or more functions . . . wherein the two or more functions are assigned to groups of tiles communicatively

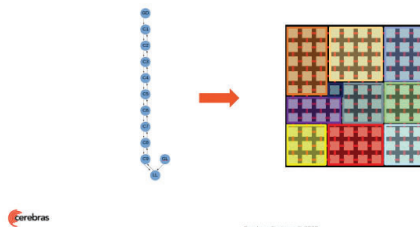
of tiles communicatively coupled in square configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the square configurations and are assigned to groups of tiles communicatively coupled in linear configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the linear configurations.

coupled in square configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the square configurations and are assigned to groups of tiles communicatively coupled in linear configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the linear configurations.

See, e.g., Vassilieva Pres. at 34:47 (“Place Kernels & Route On-Chip Network”).

See, e.g., Vassilieva Neocortex at 37

Place Kernels & Route On-Chip Network



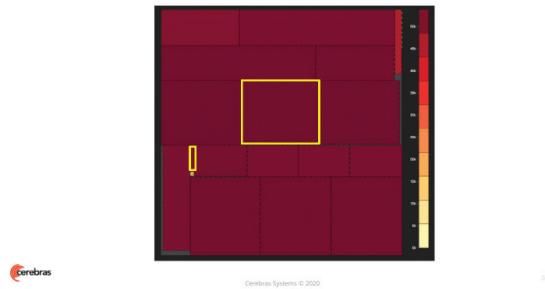
Cerebras

Cerebras Systems © 2020

37

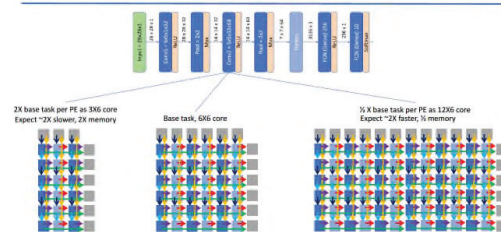
See, e.g., Vassilieva Pres. at 36:53.

See, e.g., Vassilieva Neocortex at 38



See, e.g., Groeneveld EDPS 2020 at 16

Area vs Performance of a Kernel

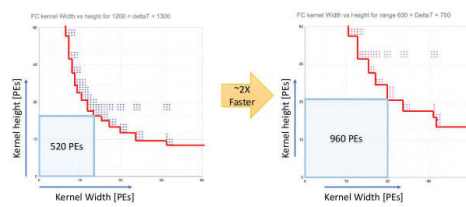


See, e.g., Groeneveld EDPS Pres. at 18:14 (“Typically a kernel will be implemented as a square or rectangular array of processors on our wafer scale engine.”).

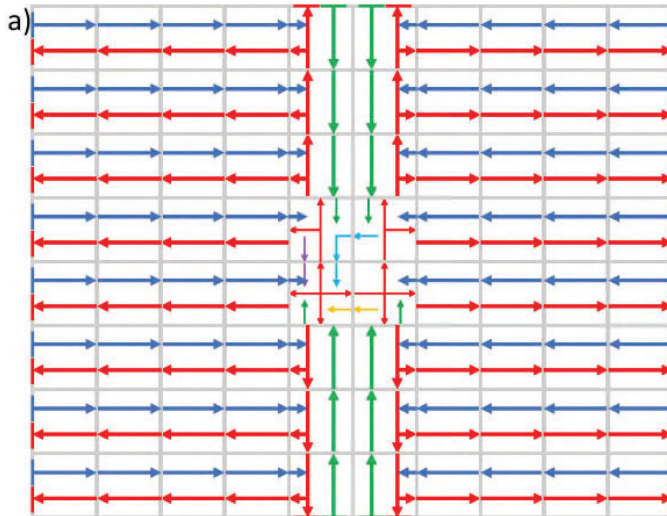
See, e.g., Groeneveld EDPS Pres. at 19:26 (“I can either make one that’s almost square . . . I can make all kinds of different shapes.”).

Groeneveld EDPS 2020 at 17.

Fully Connected Kernel:
Actual **Shape Curves** for Different Throughputs



See, e.g., Rocki at 7



	<p><i>See, e.g.,</i> Rocki at 8 (“Figure 6. The reduction is performed in parallel along fabric rows, then along two central columns. When the reduction starts, each core sends its value toward the center of its row. On each successive cycle the two central cores of that row receive a datum and accumulate it into a local sum. We use two cores in the center, each receiving input from one direction at the rate of one datum per cycle. Similarly, the partial sums are reduced along two columns towards the central four cores that finally reduce their content to a single core”).</p> <p><i>See, e.g.,</i> US 2020/0005142, ¶ [0495] (“In some embodiments and/or usage scenarios, an array of identical instances of a same ASIC is formed on a wafer, and each of the same ASICs comprises a plurality of identical instances of a same PE (e.g., PE 499), forming a wafer (e.g., Wafer 412) usable in wafer-scale integration techniques. In some embodiments and/or usage scenarios, a peripheral portion of the PEs are coupled to I/O FPGAs 420. Example ASICs are illustrated as ASIC 410, comprising a column-organized section of PEs (replicated, e.g., in a one-dimensional fashion to form a wafer), and ASIC 411, comprising a square-organized section or a rectangular-organized section of PEs (replicated, e.g., in a two-dimensional fashion to form a wafer). Other organizations of ASICs on a wafer are contemplated.”).</p> <p><i>See, e.g.,</i> CRBR001477 at CRBR00149 [REDACTED]</p>
--	--

Claim 2

[2.0] The system of claim 1, wherein:	<i>See</i> Claim 1.
[2.1] a traffic condition occurs when packets of more than one input port of a particular router are to be sent via the same	<p>In the Cerebras CS-1 and CS-2, a traffic condition occurs when packets of more than one input port of a particular router are to be sent via the same output port of the particular router.</p> <p><i>See</i> claim element [1.10], <i>supra</i>.</p>

output port of the particular router; and	<p><i>See, e.g.</i>, US 10,762,418 at 37:37-38:14 (“When a color is back pressured, data queued at each hop within the fabric is stalled. Conceptually, the queued data is an extension to a queue at the destination since it is drained into the destination once the backpressure is released. For example, the backpressure signal from a particular PE and corresponding to a particular color is only asserted when a data queue of the router of the particular PE and corresponding to the particular color is at a predetermined threshold (e.g., full or nearly full). Therefore, with respect to the particular color, data flows until reaching a stalled PE, such that the data queue effectively operates as a portion of a distributed in-fabric queue.</p> <p>The fixed routing pattern provides for multicast replication within each router. Multicast enables high fan-out communication patterns, such as within some neural network workloads. To perform multicast, each router node is statically configured with multiple outputs per multicast color. The router replicates an incoming wavelet corresponding to the multicast color to all outputs specified by the static configuration before processing the next wavelet of the multicast color. In some circumstances there are a plurality of multicast colors, each statically configured with a respective set of multiple outputs.</p> <p>The router provides for multiple input sources per color and processes a single active input source at a time. Coordination of the input sources is performed, for example, by software at a higher-level (e.g. flow control dependency, explicit messaging between PEs, or other suitable mechanisms) so that only a single input source is active at a time. Implementing a single active input source enables, in some embodiments and/or usage scenarios, relatively lower-cost hardware since the router has a single buffer per color instead of a buffer per input source.</p> <p>Since there is only a single active input source at a time, there is not any congestion within a color. However, in some circumstances, congestion occurs between colors since the colors share a single physical channel. The router responds to the congestion by scheduling between ready colors onto a single shared output channel.</p> <p>Deadlock on the fabric is possible since the fabric is blocking (e.g., the fabric and the routers have no hardware deadlock avoidance mechanisms). Deadlock is avoided by software configuring the fixed routing patterns to be free of dependent loops, thus avoiding circular dependencies and deadlock.”).</p>
---	--

<p>[2.2] the static priority routing policy assigns priority levels to each input port of the particular router, causing data packets from the higher priority input port to be sent before data packets from the lower priority input ports in the event of the traffic condition.</p>	<p>In the Cerebras CS-1 and CS-2, the static priority routing policy assigns priority levels to each input port of the particular router, causing data packets from the higher priority input port to be sent before data packets from the lower priority input ports in the event of the traffic condition.</p> <p><i>See</i> claim element [1.10], <i>supra</i>.</p> <p><i>See, e.g.,</i> Manohararajah Pres. at 51:52 (“It’s statically configured. It’s a virtual channel. So there’s some number of virtual channels. Certain virtual channels can talk to certain other virtual channels. So there are some restrictions as to as to how the connectivity is achieved and so that’s one of the constraints within the router, so I have to adhere to that constraint”).</p> <p><i>See, e.g.,</i> Manohararajah Pres. at 52:42 (“[Q:] You sometimes overlay two traffic flows on the same links but you do that by scheduling them clock cycle by clock cycle as part of your compilation? [A:] Correct, uh no that’s done but that’s done by the hardware. That is not something that the software deals with. [Q:] So it’s still buffering and it still figures out when it can send it, the software just figures out it’s not going to over allocate it? [A:] Exactly, yeah.”).</p> <p><i>See, e.g.,</i> US 10,762,418 at 37:37-38:14 (“When a color is back pressured, data queued at each hop within the fabric is stalled. Conceptually, the queued data is an extension to a queue at the destination since it is drained into the destination once the backpressure is released. For example, the backpressure signal from a particular PE and corresponding to a particular color is only asserted when a data queue of the router of the particular PE and corresponding to the particular color is at a predetermined threshold (e.g., full or nearly full). Therefore, with respect to the particular color, data flows until reaching a stalled PE, such that the data queue effectively operates as a portion of a distributed in-fabric queue.</p> <p>The fixed routing pattern provides for multicast replication within each router. Multicast enables high fan-out communication patterns, such as within some neural network workloads. To perform multicast, each router node is statically configured with multiple outputs per multicast color. The router replicates an incoming wavelet corresponding to the multicast color to all outputs specified by the static configuration before processing the next wavelet of the multicast color. In some circumstances there are a plurality of multicast colors, each statically configured with a respective set of multiple outputs.</p>
---	---

The router provides for multiple input sources per color and processes a single active input source at a time. Coordination of the input sources is performed, for example, by software at a higher-level (e.g. flow control dependency, explicit messaging between PEs, or other suitable mechanisms) so that only a single input source is active at a time. Implementing a single active input source enables, in some embodiments and/or usage scenarios, relatively lower-cost hardware since the router has a single buffer per color instead of a buffer per input source.

Since there is only a single active input source at a time, there is not any congestion within a color. However, in some circumstances, congestion occurs between colors since the colors share a single physical channel. The router responds to the congestion by scheduling between ready colors onto a single shared output channel.

Deadlock on the fabric is possible since the fabric is blocking (e.g., the fabric and the routers have no hardware deadlock avoidance mechanisms). Deadlock is avoided by software configuring the fixed routing patterns to be free of dependent loops, thus avoiding circular dependencies and deadlock.”).

See, e.g., US2021/0256362, [0715] (“The priority scheduling policy comprises the scheduler choosing from among a first set of predetermined colors (e.g., colors 0-7) with higher priority than from among a second set of predetermined colors (e.g., colors 8-15)”).

See, e.g., CRBR000706 at CRBR000726 [REDACTED]

See, e.g., CRBR000706 at CRBR000758 [REDACTED]

To the extent that Cerebras contends this limitation is not literally met, the CS-1 and CS-2 also meet the limitation under the doctrine of equivalents. Each router in CS-1 and CS-2 is capable of

	<i>performing substantially the same function of implementing a static routing policy, in substantially the same way of the claimed static routing policy that assigns priority levels to each input port of the particular router, to achieve the same result of causing data packets of higher priority to be sent before lower priority data packets. Moreover, the static routing policies in CS-1 and CS-2 are not substantially different from the claimed limitation.</i>
--	--

Claim 3

[3.0] The system of claim 1, wherein	See Claim 1.
[3.1] a traffic condition occurs when a particular router is ready to send a data packet to another router adjacent to the particular router that is not ready to receive the data packet.	<p>In the Cerebras CS-1 and CS-2, a traffic condition occurs when a particular router is ready to send a data packet to another router adjacent to the particular router that is not ready to receive the data packet.</p> <p>See claim element [1.9], <i>supra</i>.</p> <p>See, e.g., Manohararajah Pres. at 52:42 (“[Q:] You sometimes overlay two traffic flows on the same links but you do that by scheduling them clock cycle by clock cycle as part of your compilation? [A:] Correct, uh no that’s done but that’s done by the hardware. That is not something that the software deals with. [Q:] So it’s still buffering and it still figures out when it can send it, the software just figures out it’s not going to over allocate it? [A:] Exactly, yeah.”).</p> <p>See, e.g., US 10,762,418 at 37:37-38:14 (“When a color is back pressured, data queued at each hop within the fabric is stalled. Conceptually, the queued data is an extension to a queue at the destination since it is drained into the destination once the backpressure is released. For example, the backpressure signal from a particular PE and corresponding to a particular color is only asserted when a data queue of the router of the particular PE and corresponding to the particular color is at a predetermined threshold (e.g., full or nearly full). Therefore, with respect to the particular color, data flows until reaching a stalled PE, such that the data queue effectively operates as a portion of a distributed in-fabric queue.</p>

	<p>The fixed routing pattern provides for multicast replication within each router. Multicast enables high fan-out communication patterns, such as within some neural network workloads. To perform multicast, each router node is statically configured with multiple outputs per multicast color. The router replicates an incoming wavelet corresponding to the multicast color to all outputs specified by the static configuration before processing the next wavelet of the multicast color. In some circumstances there are a plurality of multicast colors, each statically configured with a respective set of multiple outputs.</p> <p>The router provides for multiple input sources per color and processes a single active input source at a time. Coordination of the input sources is performed, for example, by software at a higher-level (e.g. flow control dependency, explicit messaging between PEs, or other suitable mechanisms) so that only a single input source is active at a time. Implementing a single active input source enables, in some embodiments and/or usage scenarios, relatively lower-cost hardware since the router has a single buffer per color instead of a buffer per input source.</p> <p>Since there is only a single active input source at a time, there is not any congestion within a color. However, in some circumstances, congestion occurs between colors since the colors share a single physical channel. The router responds to the congestion by scheduling between ready colors onto a single shared output channel.</p> <p>Deadlock on the fabric is possible since the fabric is blocking (e.g., the fabric and the routers have no hardware deadlock avoidance mechanisms). Deadlock is avoided by software configuring the fixed routing patterns to be free of dependent loops, thus avoiding circular dependencies and deadlock.”).</p> <p><i>See, e.g.</i>, US 10,762,418 at 38:36-62 (“In addition to data, Router 510 selectively and/or conditionally communicates (e.g. transmits and receives) backpressure information between the other PEs and PE 500 via couplings 511-516. Router 510 selectively and/or conditionally transmits backpressure information to PE 500 via On Ramp 522. Router 510 receives backpressure information from PE 500 via Off Ramp 521. The backpressure information provided to the other PEs, as well as the backpressure information provided to PE 500, is used by the other PEs and PE 500 to stall transmitting data (e.g. wavelets) that would otherwise be lost due to insufficient queue space to store the data in Router 510. The backpressure information</p>
--	--

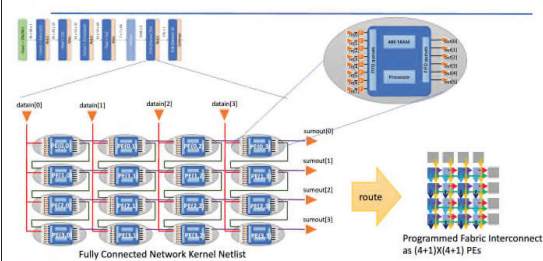
	<p>received from the other PEs and PE 500 is used respectively by Router 510 to prevent transmitting data (e.g. wavelets) that would otherwise be lost due respectively to insufficient queue space in the routers of the other PEs and insufficient space in input queues of Compute Element 520.”).</p> <p><i>See, e.g.</i>, US 10,762,418 at 67:26-32 (“In some embodiments and/or usage scenarios, wavelets are received by the router, queued, and routed to router output ports without any specific determination that a wavelet is for a local CE. Instead, wavelets destined for the local CE are routed to the off ramp and are then written into the picker queue. Wavelets not destined for the local CE are routed to other-than the off ramp router outputs.”).</p> <p><i>See, e.g.</i>, CRBR001477 at CRBR001489 [REDACTED]</p>
--	--

Claim 4

[4.0] The system of claim 1, wherein	<i>See</i> Claim 1.
[4.1] each router utilizes a set of FIFO memory elements, and wherein:	<p>In the Cerebras CS-1 and CS-2, each router utilizes a set of FIFO memory elements (<i>e.g.</i>, “FIFO queues”).</p> <p><i>See</i> claim element [1.9], <i>supra</i>.</p>

See, e.g., Groeneveld EDPS 2020 at 15.

Netlist and Routing of Processor Elements



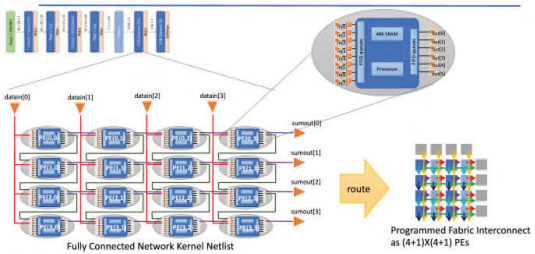
See, e.g., Rocki at 7 (“The Cerebras hardware FIFOs have a distinctive feature. They are able to activate tasks, in this case sumtask, whenever they aren’t empty. This they do. The hardware task scheduler runs sumtask (while the threads continue to fill the FIFOs). It executes five vector add instructions (as shown in the white box) that each has one FIFO input. Each add pulls as much data as it can from its input FIFO, finishing when empty. Their destination tensor descriptors track their progress, so that they add once to each of the result tensor. So sumtask is invoked repeatedly, pulling from the FIFOs, allowing the upstream threads to continue to push data into them.”).

See, e.g., CRBR000575 at CRBR000589

[4.2] each FIFO memory element of a particular router is operable to store a data packet for subsequent sending to a different router of one or more routers adjacent to the particular

In the Cerebras CS-1 and CS-2, each FIFO memory element of a particular router is operable to store a data packet for subsequent sending to a different router of one or more routers adjacent to the particular router or the processor core corresponding to the router.

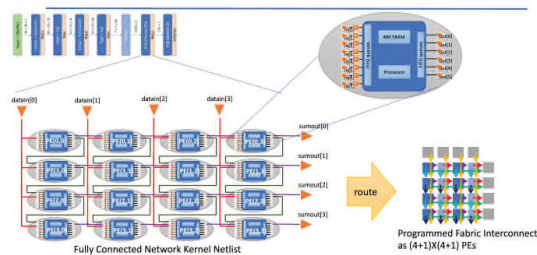
See claim element [1.9], *supra*.

<p>router or the processor core corresponding to the router; and</p>	<p><i>See, e.g.</i>, Groeneveld EDPS Pres. at 17:16 (“Each of those processing elements then runs a little program to perform the actions and the actions would be: read a piece of data from the input that comes into the PE; multiply that by the data that comes from the memory; put that output in the memory and, after a while, send it back out on its merry way to the neighboring processing elements.”).</p> <p><i>See, e.g.</i>, Groeneveld EDPS 2020 at 15.</p> <p>Netlist and Routing of Processor Elements</p>  <p><i>See, e.g.</i>, Rocki at 7 (“The Cerebras hardware FIFOs have a distinctive feature. They are able to activate tasks, in this case sumtask, whenever they aren’t empty. This they do. The hardware task scheduler runs sumtask (while the threads continue to fill the FIFOs). It executes five vector add instructions (as shown in the white box) that each has one FIFO input. Each add pulls as much data as it can from its input FIFO, finishing when empty. Their destination tensor descriptors track their progress, so that they add once to each of the result tensor. So sumtask is invoked repeatedly, pulling from the FIFOs, allowing the upstream threads to continue to push data into them.”).</p>
<p>[4.3] each FIFO memory element is operable to retain a data packet in the event of a traffic condition.</p>	<p>In the Cerebras CS-1 and CS-2, each FIFO memory element is operable to retain a data packet in the event of a traffic condition.</p> <p><i>See</i> claim element [1.9], <i>supra</i>.</p>

See, e.g., Groeneveld EDPS Pres. at 17:16 (“Each of those processing elements then runs a little program to perform the actions and the actions would be: read a piece of data from the input that comes into the PE; multiply that by the data that comes from the memory; put that output in the memory and, after a while, send it back out on its merry way to the neighboring processing elements.”).

See, e.g., Groeneveld EDPS 2020 at 15.

Netlist and Routing of Processor Elements



See, e.g., Rocki at 3 (“The router has hardware queues for its connection to the core and for each of a set of virtual channels, avoiding deadlock.”).

See, e.g., Rocki at 7 (“The Cerebras hardware FIFOs have a distinctive feature. They are able to activate tasks, in this case sumtask, whenever they aren’t empty. This they do. The hardware task scheduler runs sumtask (while the threads continue to fill the FIFOs). It executes five vector add instructions (as shown in the white box) that each has one FIFO input. Each add pulls as much data as it can from its input FIFO, finishing when empty. Their destination tensor descriptors track their progress, so that they add once to each of the result tensor. So sumtask is invoked repeatedly, pulling from the FIFOs, allowing the upstream threads to continue to push data into them.”).

Claim 5

[5.0] The system of claim 4, wherein	<i>See</i> Claim 4.
[5.1] each FIFO memory element is a single stage memory element.	In the Cerebras CS-1 and CS-2, each FIFO memory element is a single stage memory element. <i>See, e.g.,</i> claim elements [4.1], [4.2], and [4.3], <i>supra</i> .

Claim 7

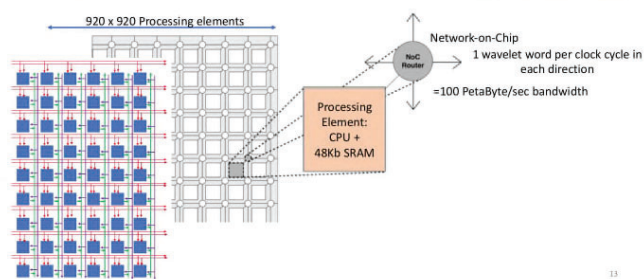
[7.0] The system of claim 1, wherein	<i>See</i> Claim 1.
[7.1] each router of the set of routers does not include SRAM (Static Random-Access Memory) and does not utilize SRAM for routing processes.	In the Cerebras CS-1 and CS-2, each router of the set of routers does not include SRAM (Static Random-Access Memory) and does not utilize SRAM for routing processes. <i>See, e.g.,</i> CRBR000706 at CRBR000725 [REDACTED].

Claim 8

[8.0] The system of claim 1, wherein	<i>See</i> Claim 1.
[8.1] the set of routers are arranged in a grid configuration.	In the Cerebras CS-1 and CS-2, the set of routers are arranged in a grid configuration. <i>See</i> claim element [1.2], <i>supra</i> . <i>See, e.g.,</i> Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That’s how every processing element looks like. They’re all identical.”)

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect



Valavan Manohararajah

The diagram shows the Cerebras Architecture, featuring a grid of 630 x 630 Processing elements. A callout box labeled 'Processing Element: CPU + 48Kb SRAM' points to a single element. To the right, a 'Network on a Chip' is shown with a 'NoC Router' at its center. Arrows indicate communication between the router and the grid. Text next to the router specifies: '1 word/clock cycle each direction' and '100 PetaByte/sec bandwidth'. The Cerebras logo is in the bottom left, and a 'zoom' watermark is in the bottom right.

See, e.g., Verdoolaege at 2 (“The target architecture is an MPPA (Massively Parallel Processor Array), consisting of a 2-dimensional grid of PEs [processing elements] that communicate with their nearest neighbors in the four cardinal directions.”).

	<p><i>See, e.g.</i>, US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).</p> <p><i>See, e.g.</i>, CRBR000575 at CRBR00058 [REDACTED]</p>
--	--

Claim 9

[9.0] The system of claim 8, wherein:	<i>See</i> Claim 8.
[9.1] each processor core includes a physically addressable memory module, and the memory module of adjacent cores include	In the Cerebras CS-1 and CS-2, each processor core includes a physically addressable memory module, and the memory module of adjacent cores include sequential physical addressing with one another; and the routers determine to which adjacent router to send a data packet based on a physical destination address of the data packet and the sequential physical addressing.

sequential physical addressing with one another; and the routers determine to which adjacent router to send a data packet based on a physical destination address of the data packet and the sequential physical addressing.

See, e.g., CRBR000575 at CRBR000589

See, e.g., CRBR001477 at CRBR00148

See, e.g., CRBR001477 at CRBR001491

To the extent that Cerebras contends this limitation is not literally met, the CS-1 and CS-2 also meet the limitation under the doctrine of equivalents. Each router in CS-1 and CS-2 is operable to perform the substantially the same function of determining to which adjacent router to send a data packet, the router corresponding to a processor core including a physically addressable memory module including sequential physical addressing with adjacent processing cores, in substantially the same way that is based on the physical destination address of the data packet, to achieve the same result of the limitation. Moreover, each router in CS-1 and CS-2 is not substantially different from the claimed limitation.

Claim 10

[10.1] The system of claim 1, wherein	<i>See</i> Claim 1.
[10.1] the static priority routing policy assigns static priority to designated input ports of a router.	<p>In the Cerebras CS-1 and CS-2, the static priority routing policy assigns static priority to designated input ports of a router.</p> <p><i>See</i> claim element [1.10], <i>supra</i>.</p> <p><i>See, e.g.</i>, US 2021/0256362, ¶ [0688] (“There is a routing pattern associated with each color and implemented by the routers . The routing pattern of each pattern is programmable and in some embodiments is statically configured , e.g. , based at least in part on determinations made by Placement Server(s) SW 210 and /or Neuron to PE Mapping SW 212 of FIG . 2. Once configured, e.g. , under control of software (such as Connection Server(s) SW 220 of FIG . 2) , each color is a fixed routing pattern. All data that flows within a color always flows in accordance with the fixed routing pattern . There are no dynamic routing decisions . The fixed routing matches neural network communication patterns where neuron connections are statically specified. The fixed routing enables relatively lower cost hardware implementation.”).</p>

See, e.g., CRBR000575 at CRBR000582



See, e.g., CRBR000706 at CRBR000726



To the extent that Cerebras contends this limitation is not literally met, the CS-1 and CS-2 also meet the limitation under the doctrine of equivalents. Each router in CS-1 and CS-2 is capable of performing substantially the same function of implementing a static routing policy, in substantially the same way of the claimed static routing policy that assigns priority levels to each

input port of the particular router, to achieve the same result of sending and receiving data packets to and from adjacent routers. Moreover, the static routing policies in CS-1 and CS-2 are not substantially different from the claimed limitation.

Claim 11

[11.0] The system of claim 1, wherein

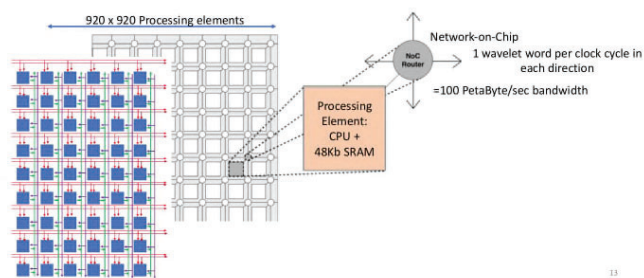
See Claim 1.

[11.1] each router is operable to simultaneously receive data packets from one or more input ports and simultaneously send data packets on one or more output ports.

In the Cerebras CS-1 and CS-2, each router is operable to simultaneously receive data packets from one or more input ports and simultaneously send data packets on one or more output ports.

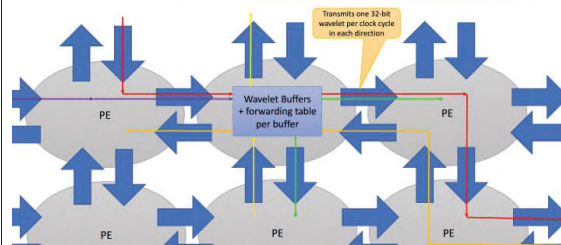
See, e.g., Groeneveld EDPS 2020 at 13.

Wafer Scale Engine: General Arrangement



See, e.g., Groeneveld EDPS 2020 at 14.

Wavelets Flowing on the Network-on-Chip



See, e.g., Rocki at 3 (“The core connects to a local router that has five bidirectional links, one to each of its four nearest neighbors and one to its own core. The router can move data into and out of these five links, in parallel, on every cycle. Even with scalar granularity, communication is efficient.”).

See, e.g., US 10,762,418 at 37:349-59 (“The fixed routing pattern provides for multicast replication within each router. Multicast enables high fan-out communication patterns, such as within some neural network workloads. To perform multicast, each router node is statically configured with multiple outputs per multicast color. The router replicates an incoming wavelet corresponding to the multicast color to all outputs specified by the static configuration before processing the next wavelet of the multicast color. In some circumstances there are a plurality of multicast colors, each statically configured with a respective set of multiple outputs.”).

See, e.g., US 2020/0005142, ¶ [0508] (“FIG. 7 illustrates selected details of an embodiment of processing associated with a router of a processing element, as Wavelet Ingress 710, Stall Info 720, and Wavelet Egress 730. Conceptually, the router accepts as many wavelets as possible from ingress ports, queuing as necessary and as queue space is available, and routes as many wavelets as possible to egress ports per unit time (e.g., clock cycle). Wavelet Ingress 710 comprises actions 711-713 corresponding to wavelet ingress from (logically and/or physically) adjacent PEs and/or an instant PE, for each respective queue. Stall Info 720 comprises actions 721-723 correspond to

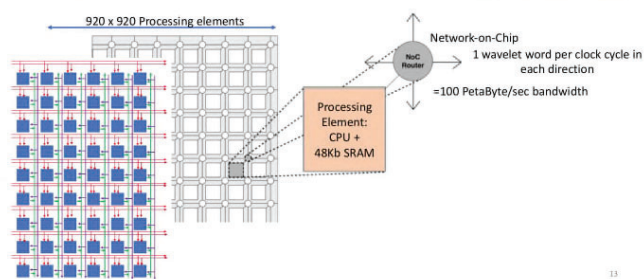
	<p>providing stall information, for each respective queue. Wavelet Egress 730 comprises actions 731-734 that correspond to wavelet egress to (logically and/or physically) adjacent PEs and/or the instant PE, for each respective queue. In some circumstances, in accordance with color information of a wavelet and routing configuration information, Send Wavelet 734 sends a wavelet from a single queue entry to a single destination (e.g., unicast). In some circumstances, in accordance with color information of a wavelet and routing configuration information, Send Wavelet 734 sends a wavelet from a single queue entry to a plurality of destinations (e.g., multicast). In various embodiments and/or usage scenarios, any one or more of all or any portions of actions of 710, 720, and/or 730 correspond to actions performed by and/or related to all or any portions of any one or more elements of Router 600 of FIG. 6.”).</p>
--	--

Claim 12

[12.0] The system of claim 1, wherein	See Claim 1.
[12.1] the circuit design defining each router is identical and the circuit design defining each processor core is identical.	<p>In the Cerebras CS-1 and CS-2, the circuit design defining each router is identical and the circuit design defining each processor core is identical.</p> <p>See, e.g., Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That’s how every processing element looks like. They’re all identical.”)</p>

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect



Valavan Manohararajah

The diagram shows a 630 x 630 grid of processing elements. A callout box labeled 'Processing Element: CPU + 48Kb SRAM' points to one of the elements. To the right, a 'Network on a Chip' router is shown with four ports. Text next to it states: '1 word/clock cycle each direction' and '100 PetaByte/sec bandwidth'. The Cerebras logo is in the bottom left, and the Zoom logo is in the bottom right.

See, e.g., US 2020/0005142, ¶ [0494] (“FIG. 4 illustrates selected details of an embodiment of a deep learning accelerator as Deep Learning Accelerator 400. Each of PE 499 elements has couplings to other of PE 499 elements. Two of the PE elements (PE 497 and PE 498) are

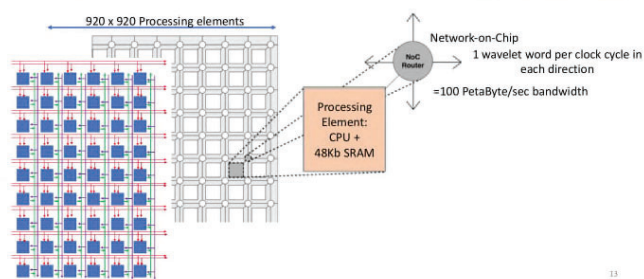
	<p>illustrated with unique identifiers, and are otherwise respectively identical to a instances of PE 499. PE 497 is illustrated with identifiers for each of four couplings (North coupling 430, East coupling 431 with PE 498, and South coupling 432) to others of the PEs and one of the I/O FPGAs (West coupling 433), but is otherwise identical to others of the PE elements illustrated. In some embodiments and/or usage scenarios, the couplings are logical and/or physical. In various embodiments and/or usage scenarios, the couplings are usable to communicate wavelets, backpressure information, or both. In various embodiments and/or usage scenarios, all or any portions of the physical couplings are to physically adjacent PEs. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid. In some embodiments and/or usage scenarios, the PEs are physically implemented in a 2D grid of aligned rectangles, and physically adjacent PEs correspond to PEs sharing a horizontal boundary (North/South PEs with respect to each other) and PEs sharing a vertical boundary (East/West PEs with respect to each other).”).</p> <p><i>See, e.g.,</i> CRBR000251.</p>
--	---

Claim 13

<p>[13.0] A network-on-chip microprocessor chip, comprising:</p>	<p>The Cerebras CS-1 and CS-2 include a network on-chip microprocessor chip (<i>e.g.</i>, “network on-chip”).</p> <p><i>See, e.g.,</i> CS-1 Overview at 4 (“The Cerebras Swarm communication fabric creates a massive on-chip network”).</p> <p><i>See, e.g.,</i> Groeneveld EDPS Pres. at 14:55 (“The processing element is connected to an NoC router . . . which allows you to send data packages to your left neighbor, your right neighbor, your top neighbor, and your bottom neighbor, or to pull the data into the processing element, or inject it into the network. That’s how every processing element looks like. They’re all identical.”)</p>
--	--

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect



Valavan Manohararajah

The diagram shows the Cerebras architecture with a 630 x 630 grid of processing elements. A callout box details a single processing element, consisting of a CPU and 48Kb SRAM. This element is connected to a 'Network on a Chip' router, which supports 1 word/clock cycle in each direction, resulting in a 100 PetaByte/sec bandwidth. The Cerebras logo is visible in the bottom left, and a Zoom watermark is in the bottom right.

[13.1] a set of scratchpad memory modules;

The Cerebras CS-1 and CS-2 include a set of scratchpad memory modules (e.g., “SRAM memory”).

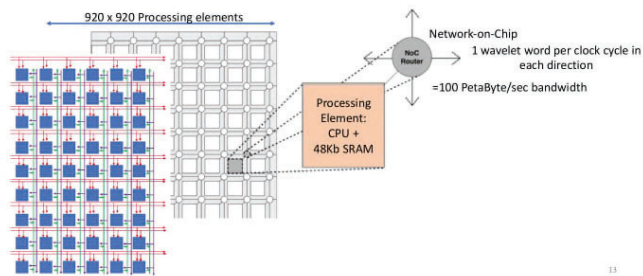
See claim element [9.1], *supra*.

See, e.g., Groeneveld EDPS Pres. at 15:02 (“Every processing element is a CPU plus 48 kilobytes of super-fast SRAM.”).

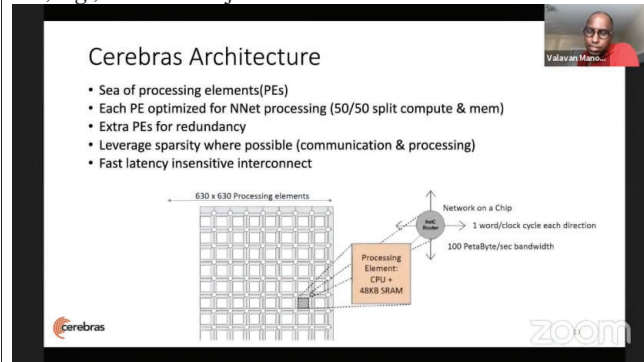
See, e.g., Groeneveld EDPS Pres. at 17:16 (“Each of those processing elements then runs a little program to perform the actions and the actions would be: read a piece of data from the input that comes into the PE; multiply that by the data that comes from the memory; put that output in the memory and, after a while, send it back out on its merry way to the neighboring processing elements.”).

See, e.g., Groeneveld EDPS 2020 at 13.

Wafer Scale Engine: General Arrangement



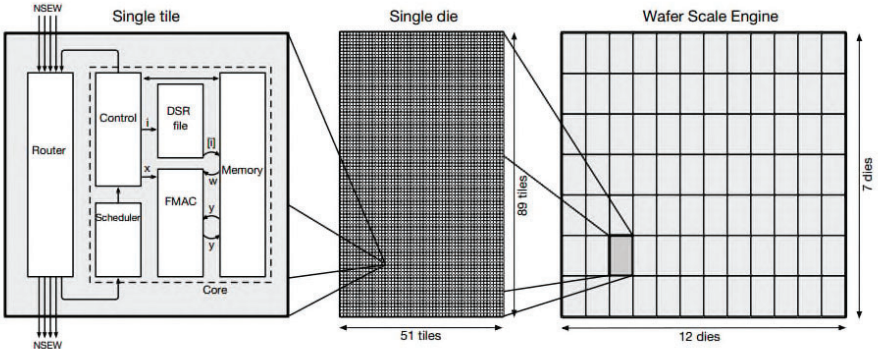
See, e.g., Manohararajah Pres. at 50:45



See, e.g., Rocki at 2 (“The system comprises 380,000 processor cores, each with 48 KB of dedicated SRAM memory (for a total of 18GB)”).

See, e.g., Rocki at 2 (“The memory, functional units, and instruction set are designed for high throughput numerical computation. The roughly 380,000 tiles each have their own fast SRAM memory. There is no shared memory. Local memory is 48 KB, which totals 18 GB across the wafer. The load-to-use latency is one cycle.”).

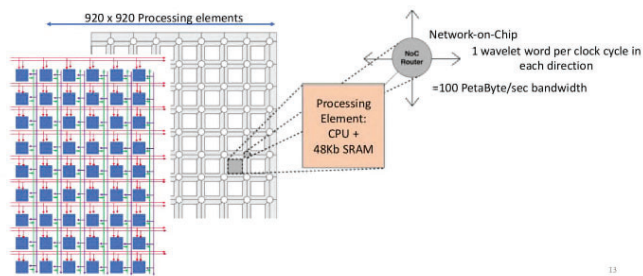
See, e.g., Rocki at 3

	 <p>Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2.</p>
<p>[13.2] a set of tiles arranged in a grid configuration, each tile including a processor core and a router communicatively coupled with one another, wherein:</p>	<p>The Cerebras CS-1 and CS-2 include a set of tiles arranged in a grid configuration, each tile including a processor core and a router communicatively coupled with one another.</p> <p><i>See claim elements [1.5] and [8.1], supra.</i></p>
<p>[13.3] each processor core corresponds to and is communicatively coupled with a different scratchpad memory module of the set of scratchpad memory modules,</p>	<p>In the Cerebras CS-1 and CS-2, each processor core corresponds to and is communicatively coupled with a different scratchpad memory module of the set of scratchpad memory modules.</p> <p><i>See claim element [9.1], supra.</i></p> <p><i>See, e.g., Groeneveld EDPS Pres. at 15:02 (“Every processing element is a CPU plus 48 kilobytes of super-fast SRAM.”).</i></p> <p><i>See, e.g., Groeneveld EDPS Pres. at 17:16 (“Each of those processing elements then runs a little program to perform the actions and the actions would be: read a piece of data from the input that</i></p>

comes into the PE; multiply that by the data that comes from the memory; put that output in the memory and, after a while, send it back out on its merry way to the neighboring processing elements.”).

See, e.g., Groeneveld EDPS 2020 at 13.


Wafer Scale Engine: General Arrangement



See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements(PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect





Valavan Manohararajah

630 x 630 Processing elements

Processing Element: CPU + 48KB SRAM

Network on a Chip: 1 word/clock cycle each direction, 100 PetaByte/sec bandwidth

See, e.g., Rocki at 2 (“The system comprises 380,000 processor cores, each with 48 KB of dedicated SRAM memory (for a total of 18GB)”).

See, e.g., Rocki at 2 (“The memory, functional units, and instruction set are designed for high throughput numerical computation. The roughly 380,000 tiles each have their own fast SRAM memory. There is no shared memory. Local memory is 48 KB, which totals 18 GB across the wafer. The load-to-use latency is one cycle.”).

See, e.g., Rocki at 3

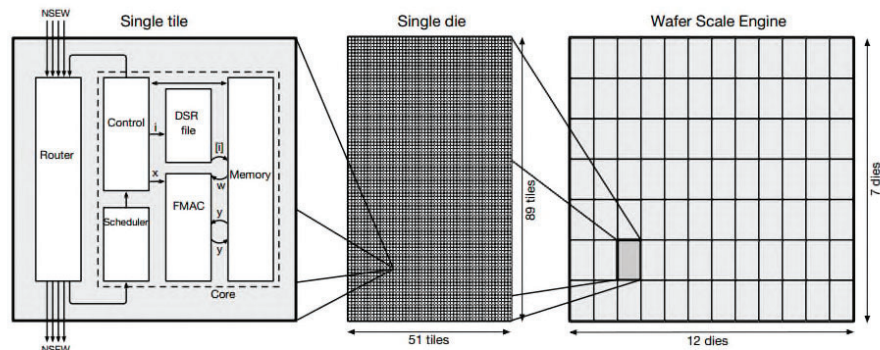


Fig. 2. CS-1 Wafer Scale Engine (WSE). A single wafer (rightmost) contains one CS-1 processor. Each processor is a collection of dies arranged in a 2D fashion (middle). Dies are then further subdivided into a grid of tiles. One die hosts thousands of computational cores, memory and routers (leftmost). There is no logical discontinuity between adjacent dies and there is no additional bandwidth penalty for crossing the die-die barrier. In total, there are 1.2 trillion transistors in an area of 462.25 cm^2 .

[13.4] each router includes a set of input ports and a set of output ports, wherein

In the Cerebras CS-1 and CS-2, each router includes a set of input ports and a set of output ports.

See claim element [1.2], *supra*.

[13.5] each output port includes a FIFO memory element operable to store a data packet for subsequent sending to a router of an adjacent tile, based on a physical destination address of a data packet,	In the Cerebras CS-1 and CS-2, each output port includes a FIFO memory element operable to store a data packet for subsequent sending to a router of an adjacent tile, based on a physical destination address of a data packet. <i>See claim elements [1.8], [4.1], and [4.2], supra.</i>
[13.6] each router is operable to send one or more data packets to routers of one or more adjacent tiles or the processor core corresponding to the router, and	In the Cerebras CS-1 and CS-2, each router is operable to send one or more data packets to routers of one or more adjacent tiles or the processor core corresponding to the router. <i>See claim elements [1.8], supra.</i>
[13.7] each router implements a static priority routing policy in the event of a traffic condition; and	In the Cerebras CS-1 and CS-2, each router implements a static priority routing policy in the event of a traffic condition. <i>See claim elements [1.9] and [10], supra.</i>
[13.8] an optimization module configured to: determine optimal function assignment configurations for groups of tiles of the set of tiles, and	The Cerebras CS-1 and CS-2 include an optimization module configured to determine optimal function assignment configurations for groups of tiles of the set of tiles. <i>See claim element [1.11], supra.</i>
[13.9] assign two or more functions, which communicate at least unilaterally more frequently with one another than with other functions, to groups of adjacent tiles based on an optimal function assignment configuration determination, wherein	The Cerebras CS-1 and CS-2 include an optimization module configured to assign two or more functions, which communicate at least unilaterally more frequently with one another than with other functions, to groups of adjacent tiles based on an optimal function assignment configuration determination. <i>See claim element [1.12], supra.</i>

<p>[13.10] the two or more functions are assigned to groups of tiles communicatively coupled in square configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the square configurations and are assigned to groups of tiles communicatively coupled in linear configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the linear configurations.</p>	<p>In the Cerebras CS-1 and CS-2, two or more functions are assigned to groups of tiles communicatively coupled in square configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the square configurations and are assigned to groups of tiles communicatively coupled in linear configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the linear configurations.</p> <p><i>See claim element [1.13], supra.</i></p>
---	---

Claim 14

<p>[14.0] The network-on-chip microprocessor chip of claim 13, wherein:</p>	<p><i>See Claim 13.</i></p>
<p>[14.1] a traffic condition occurs when packets of more than one input port of a particular router are to be sent via the same output port of the particular router; and</p>	<p>In the Cerebras CS-1 and CS-2, a traffic condition occurs when packets of more than one input port of a particular router are to be sent via the same output port of the particular router.</p> <p><i>See claim element [2.1], supra.</i></p>
<p>[14.2] the static priority routing policy assigns priority levels to each input port of the particular router, causing data packets from the higher priority input</p>	<p>In the Cerebras CS-1 and CS-2, the static priority routing policy assigns priority levels to each input port of the particular router, causing data packets from the higher priority input port to be sent before data packets from the lower priority input ports in the event of the traffic condition.</p> <p><i>See claim element [2.2], supra.</i></p>

port to be sent before data packets from the lower priority input ports in the event of the traffic condition.	
--	--

Claim 15

[15.0] The network-on-chip microprocessor chip of claim 13, wherein	<i>See</i> Claim 13.
[15.1] a traffic condition occurs when a particular router is ready to send a data packet to a router of an adjacent tile that is not ready to receive the data packet.	In the Cerebras CS-1 and CS-2, a traffic condition occurs when a particular router is ready to send a data packet to a router of an adjacent tile that is not ready to receive the data packet. <i>See</i> claim element [3.1], <i>supra</i> .

Claim 16

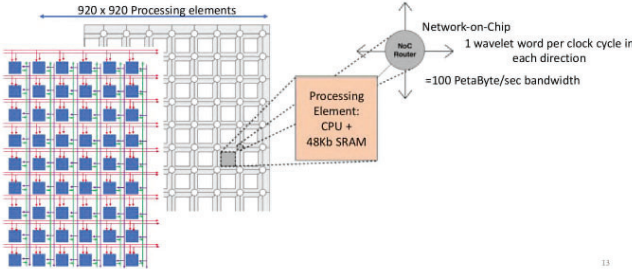
[16.0] The network-on-chip microprocessor chip of claim 13, wherein	<i>See</i> Claim 13.
[16.1] each FIFO memory element is operable to retain a data packet in the event of a traffic condition.	In the Cerebras CS-1 and CS-2, each FIFO memory element is operable to retain a data packet in the event of a traffic condition. <i>See</i> claim element [4.3], <i>supra</i> .

Claim 17

[17.0] The network-on-chip microprocessor chip of claim 13, wherein:	<i>See</i> Claim 13.
--	----------------------

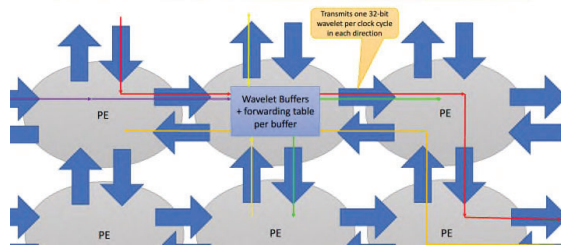
<p>[17.1] adjacent scratchpad memory modules include sequential physical addressing with one another; and</p>	<p>In the Cerebras CS-1 and CS-2, adjacent scratchpad memory modules include sequential physical addressing with one another; and the routers determine to which adjacent router to send a data packet based on a physical destination address of the data packet and the sequential physical addressing.</p> <p><i>See claim element [9.1], supra.</i></p>
---	---

Claim 18

<p>[18.0] The network-on-chip microprocessor chip of claim 13, wherein</p>	<p><i>See Claim 13.</i></p>
<p>[18.1] each router is operable to provide a data packet to a router of an adjacent tile in one clock cycle.</p>	<p>In the Cerebras CS-1 and CS-2, each router is operable to provide a data packet to a router of an adjacent tile in one clock cycle.</p> <p><i>See, e.g., Groeneveld EDPS Pres. at 15:49 (“If you want to send a data packet or a wavelet, a 32 bit wavelet, from one PE to the other, every clock cycle it gets sent”).</i></p> <p><i>See, e.g., Groeneveld EDPS 2020 at 13.</i></p> <p>Wafer Scale Engine: General Arrangement</p>  <p style="text-align: right;">33</p>

See, e.g., Groeneveld EDPS 2020 at 14.


Wavelets Flowing on the Network-on-Chip



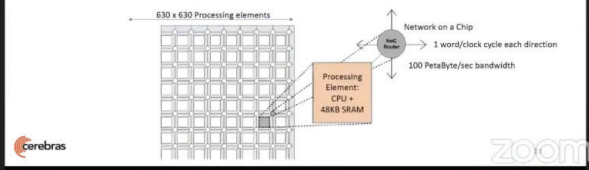
See, e.g., Manohararajah Pres. at 50:45

Cerebras Architecture

- Sea of processing elements (PEs)
- Each PE optimized for NNet processing (50/50 split compute & mem)
- Extra PEs for redundancy
- Leverage sparsity where possible (communication & processing)
- Fast latency insensitive interconnect



Valavan Manohararajah



Cerebras

See, e.g., Manohararajah Pres. at 52:42 (“[Q:] You sometimes overlay two traffic flows on the same links but you do that by scheduling them clock cycle by clock cycle as part of your compilation? [A:] Correct, uh no that’s done but that’s done by the hardware. That is not something that the software deals with. [Q:] So it’s still buffering and it still figures out when it can send it, the software just figures out it’s not going to over allocate it? [A:] Exactly, yeah.”).

	<i>See, e.g.</i> , CRBR000706 at CRBR000720 [REDACTED] [REDACTED]
	<i>See, e.g.</i> , CRBR000706 at CRBR000721 [REDACTED] [REDACTED]

Claim 19

[19.0] The network-on-chip microprocessor chip of claim 13, wherein	<i>See</i> Claim 13.
[19.1] the circuit design defining each tile is identical.	In the Cerebras CS-1 and CS-2, the circuit design defining each tile is identical. <i>See</i> claim element [12.1], <i>supra</i> .

References:

Implementing Machine Learning on Massively Parallel Hardware

Video: <https://ieee-edps.com/archives/2020/c/1007groeneveld.mp4> (last visited April 12, 2021) (“Groeneveld EDPS Pres.”)

PDF: <https://ieee-edps.com/archives/2020/c/1000groeneveld.pdf> (last visited April 12, 2021) (“Groeneveld EDPS 2020”)

Software Co-design for the First Wafer-Scale Processor (and Beyond), in 2020 IEEE Hot Chips 32 Symposium (HCS)

<https://ieeexplore.ieee.org/document/9220504> (last visited December 20, 2021) (“Software Co-design”)

Cerebras Systems Overview

<https://secureservercdn.net/198.12.145.239/a7b.fcb.myftpupload.com/wp-content/uploads/2020/03/Cerebras-Systems-Overview.pdf> (last visited May 4, 2021) (“CS-1 Overview”)

Cerebras Systems: Achieving Industry Best AI Performance Through A Systems Approach, Whitepaper 03,

<https://f.hubspotusercontent30.net/hubfs/8968533/Cerebras-CS-2-Whitepaper.pdf> (last visited December 20, 2021) (“WP03”)

Technical Overview of the Cerebras CS-1 - Neocortex - Pittsburgh Supercomputing Center

Websites: <https://www.cmu.edu/psc/aibd/neocortex/technical-overview-webinar.html> (last visited December 20, 2021);

<https://www.cmu.edu/psc/aibd/neocortex/technical-webinar-post.html> (last visited December 20, 2021).

Video: <https://youtu.be/4p7Hir6VqZk> (last visited December 20, 2021) (“Vassilieva Pres.”)

PDF: https://www.cmu.edu/psc/aibd/neocortex/files/neocortex_introcerebras_20200819.pdf (last visited December 20, 2021) (“Vassilieva Neocortex”)

Verdoolaege et al., Generating SIMD Instructions for Cerebras CS-1 using Polyhedral Compilation Techniques, IMPACT 2020 (January 22, 2020)

http://impact.gforge.inria.fr/impact2020/papers/IMPACT_2020_paper_3.pdf (last visited May 4, 2021) (“Verdoolaege”)

Rocki et al., Fast Stencil-Code Computation on a Wafer Scale Processor, SC20 (November 9-19, 2020)

<https://arxiv.org/pdf/2010.03660.pdf> (last visited December 20, 2021) (“Rocki”)

Manohararajah, Challenges in CAD for Wafer Scale Engines (October 8, 2020)

<https://www.youtube.com/watch?v=31CkHmYX9Ho> (last visited April 12, 2021) (“Manohararajah Pres.”)