Registers are just arrows

Discussion exercise

How can we design our ISA without registers?

Can we simplify the microarchitecture as a result?

Perhaps encoding dependence directly (so it doesn't have to be discovered)

Perhaps avoiding register renaming?

Code is just an encoding for a graph

```
Load r1 A
Load r2 C
Add r3 r2 #1 // r3 = C+1
Mul r4 r1 r3 // r4 = r3*A
Store r4 D    // D = r4
```

Register machine

```
Push A    // load from location A, push onto stack
Push B
Add #1    // add 1 to the value on the top of the stack
Mul       // multiply the top two items on the stack
Store     // store the value on the top of the stack to memory
```

Stack machine

```
Load +3 A  // load from location A, send to Mul instruction below
Load +1 C  // load from location C, send to Add instruction below
Add +1 #1 // add 1 to the value received, send result to next
Mul +1     // multiply value received by from Add by C, send to store
Store      // store the value received from instruction above
```

dataflow machine

```
Load A     // Load A, drop the value on the conveyor "belt"
Load C     // Load C, drop the value on the conveyor "belt"
Add -1 #1 // add 1 to the result of the instruction one instruction earlier
Mul -1 -3  // multiply the results from positions -1 and -3 on the belt
Store -1   // store the result of the multiply to memory
```

"belt" machine

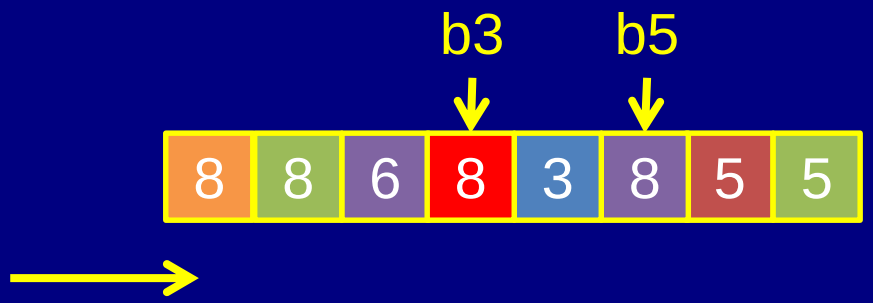
From: http://millcomputing.com/docs/belt/

Belt addressing

Belt operands are addressed by relative position

```
add    b3, b5
```

No result address!

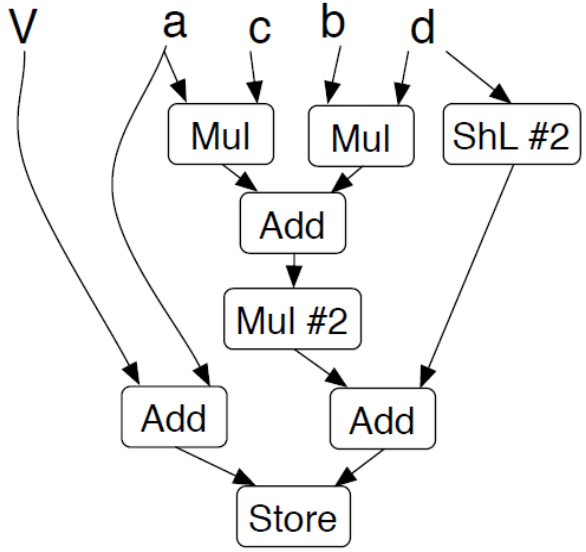


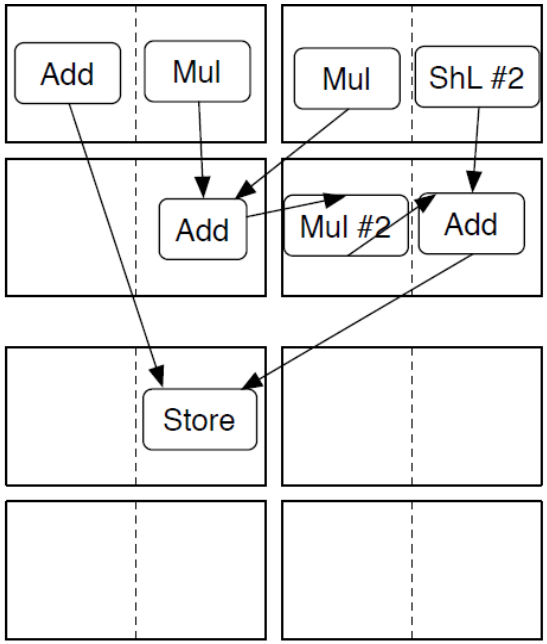
"b3" is the fourth most recent value to drop to the belt
"b5" is the sixth most recent value to drop to the belt

This is *temporal addressing*

Wavescalar's ISA encodes graph's forward arrows

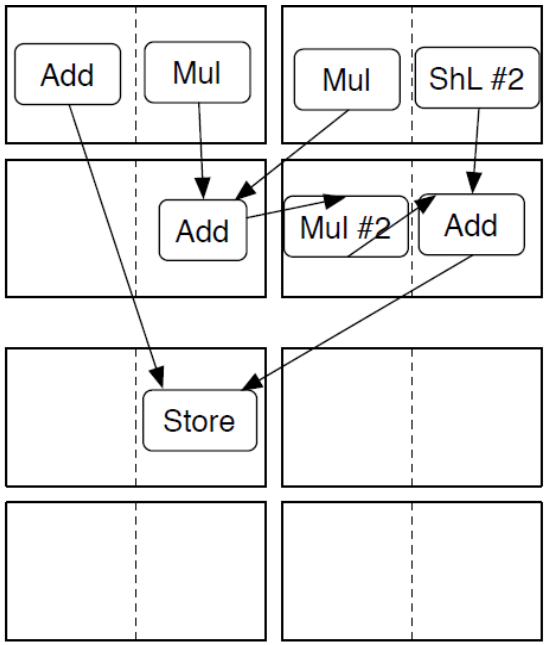
```
int *V;
int a, b;
int c, d, r;

r = a*c + b*d;
V[a] = 2*r + d << 2;
```

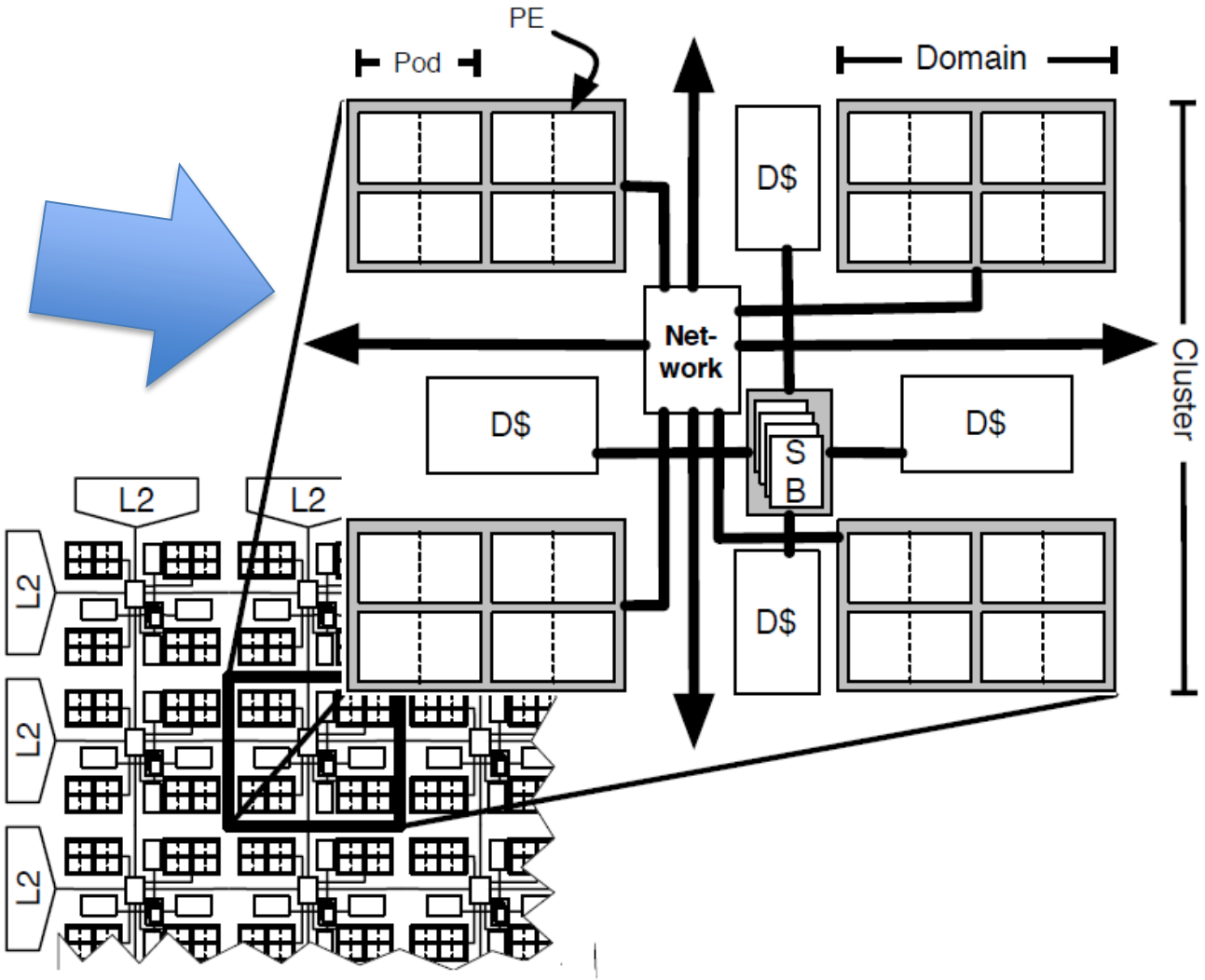




- **Three views of code in WaveScalar:** At left is the C code for a simple computation. The WaveScalar dataflow graph is shown at center, and the same graph is mapped onto 2 8-PE domains in the WaveCache substrate at right.

Wavescalar's ISA encodes graph's forward arrows





- Graph fragments are mapped onto clustered on-chip array of processing elements
- Each processing element waits for its operands
- Operands are tagged – processing element fires when it receives operands with matching tags

References:

- Sinha, Steve & Chatterjee, Satrajit & Ravindran, Kaushik. (2019). **BOOST: Berkeley's Out-of-Order Stack Thingy.** https://www.researchgate.net/publication/228556746_BOOST_Berkeley's_Out-of-Order_Stack_Thingy
- Mill Computing: https://millcomputing.com/docs/
- Schmit, Herman; Benjamin Levine; Benjamin Ylvisaker (2002). **Queue Machines: Hardware Compilation in Hardware**. *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)*: 152. doi:10.1109/FPGA.2002.1106670
- Steven Swanson, Andrew Putnam, Martha Mercaldi, Ken Michelson, Andrew Petersen, Andrew Schwerin, Mark Oskin, and Susan J. Eggers. 2006. **Area-Performance Trade-offs in Tiled Dataflow Architectures**. In Proceedings of the 33rd annual international symposium on Computer Architecture(ISCA '06). IEEE Computer Society, Washington, DC, USA, 314-326. DOI: https://doi.org/10.1109/ISCA.2006.10
- INTEL'S EXASCALE DATAFLOW ENGINE DROPS X86 AND VON NEUMANN. Aug 2018, Timothy Prickett Morgan, https://www.nextplatform.com/2018/08/30/intels-exascale-dataflow-engine-drops-x86-and-von-neuman/ . See also https://en.wikichip.org/wiki/intel/configurable_spatial_accelerator and US Patent No. US20180189231A1.