High-level .NET Software Implementations of Unum Type I and Posit with Simultaneous FPGA Implementation Using Hastlayer

Zoltán Lehóczky, Álmos Szabó @ Lombiq

CoNGA 28 March 2018


lombiq

be the hardware

Warning

Hastlayer is currently in alpha stage and posit work is ongoing!

Introducing Hastlayer

computer program → hardware (FPGA) logic

FPGAs (Field-Programmable Gate Array)

- Can behave like any other chip (with limitations)
- Can dynamically be „re-wired”
- Power efficient and highly parallelized

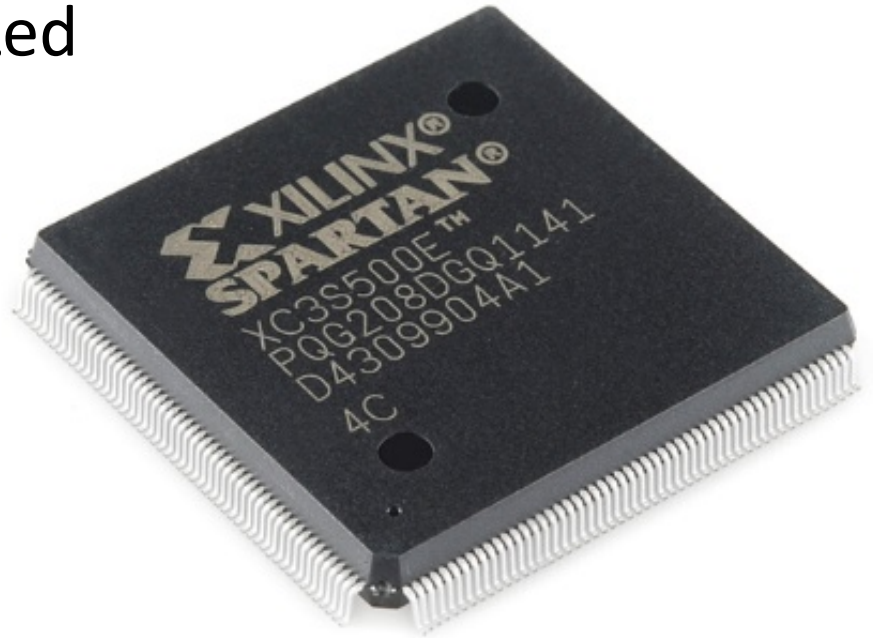


Image by SparkFun Electronics, Boulder, USA

.NET (C#, VB, C++, F#, Python, PHP, JavaScript…) → FPGA logic

The benefits of FPGAs for software developers

- Performance increase for parallel compute-bound algorithms
- Higher power efficiency
- Still only software development

Demo: Hands-on Hastlayer

Hastlayer.SDK - Microsoft Visual Studio (Administrator)

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP

Zoltán Lehóczky

ParallelAlgorithm.cs

Hast.Samples.SampleAssembly | Hast.Samples.SampleAssembly.ParallelAlgorithm | MaxDegreeOfParallelism

```
1  using System.Threading.Tasks;
2  using Hast.Transformer.Abstractions.SimpleMemory;
3
4  namespace Hast.Samples.SampleAssembly
5  {
6      /// <summary>
7      /// A massively parallel algorithm that is well suited to be accelerated with Hastlayer. Also see
8      /// <see cref="ParallelAlgorithmSampleRunner"/> on what to configure to make this work.
9      /// </summary>
10     public class ParallelAlgorithm
11     {
12         private const int MaxDegreeOfParallelism = 280;
13
14         public const int Run_InputUInt32Index = 0;
15         public const int Run_OutputUInt32Index = 0;
16
17
18         public virtual void Run(SimpleMemory memory)
19         {
20             var input = memory.ReadUInt32(Run_InputUInt32Index);
21             var tasks = new Task<uint>[MaxDegreeOfParallelism];
22
23             // Hastlayer will figure out how many Tasks you want to start if you kick them off in a loop lik
24             // If this is more involved then you'll need to tell Hastlayer the level of parallelism, see the
```

100 %

Task Runner Explorer | Error List | Output

Ready | Ln 10 | Col 35 | Ch 35 | INS | Add to Source Control

Hastlayer.SDK - Mi... | Desktop Windows -...

HUN 22:58 2018. 03. 21.

Our posit work

Implementing unum and posit in .NET

- Our implementations are open source
- We transformed our code to FPGA using Hastlayer

Overview of our implementations

- Universal unum Type I
- Universal posit
- Fix sized posit

Our Type I unum implementation

- Handles any size of unum environment
- Addition and subtraction working
- Underlying structure: BitMask

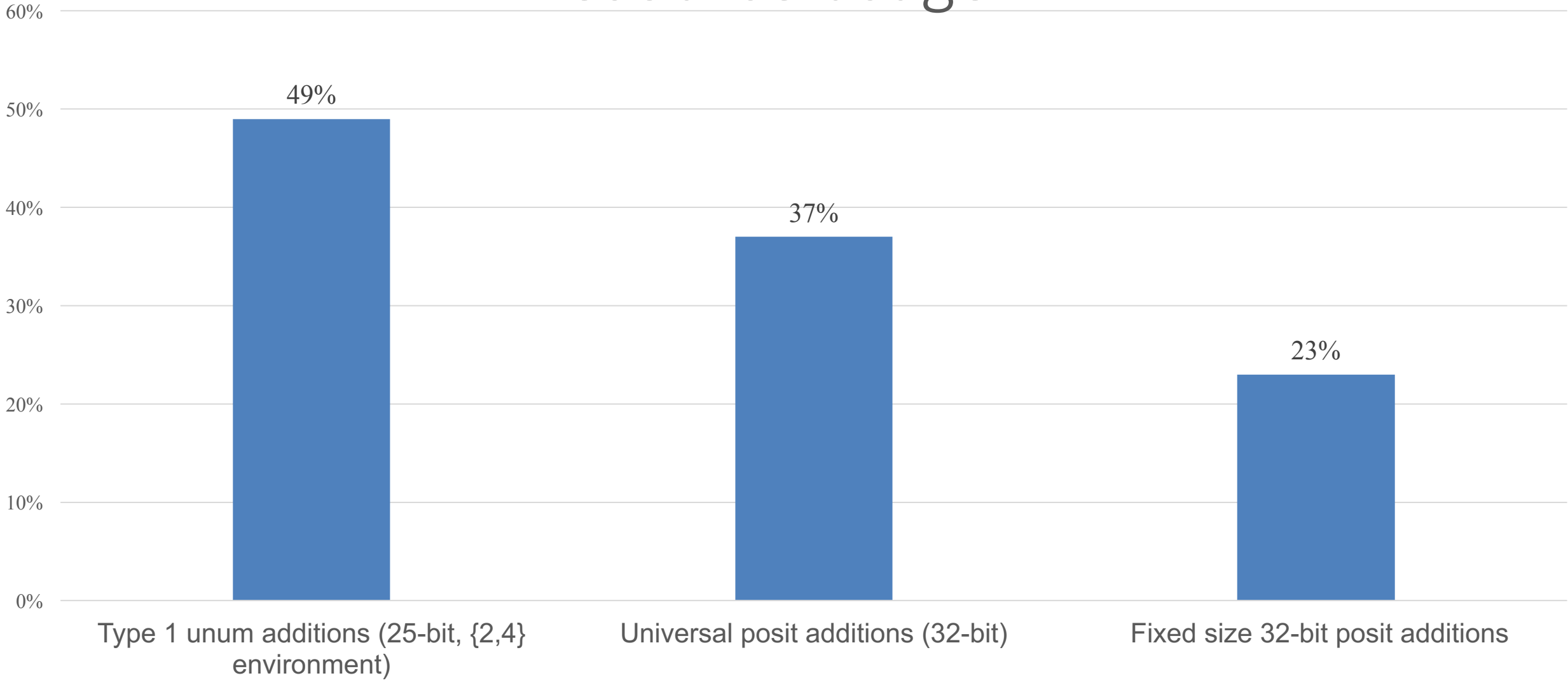
Our „universal” posit implementation

- Handles any size of posit environment
- Uses BitMask
- Slow

Our posit implementation based on built-in .NET types

- 32-bit version based on unsigned integers
- Addition, subtraction, multiplication
- Uses less resources
- Much faster

Resource usage



Performance measurements I.

100 000 additions

	CPU	FPGA
Universal posit	2 204 ms	9 646 ms
32bit posit	13 ms	203 ms

- CPU: 7.69 MPOPS
- FPGA: 0.49 MPOPS

Performance measurements I.

Clock cycles per addition

	CPU	FPGA
Universal posit	70 528	9 646
32bit posit	416	203

Parallelized performance I.

500 000 additions in parallel (5 threads)

	CPU	FPGA
32bit posit	25 ms	203 ms

- CPU: 23.8 MPOPS
- FPGA: 2.46 MPOPS

Parallelized performance II.

Clock cycles per addition

	CPU	**FPGA**
32bit posit	160	40.6

Next steps

- More operations (division, exponentiation, trigonometric functions, fused operations, etc.)
- 8, 16, 64 bit versions with code generation
- Valids

Demo: 32-bit posits in action

Hastlayer.SDK - Microsoft Visual Studio (Administrator)

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP

Zoltán Lehóczky

Posit32Calculator.cs

Hast.Samples.SampleAssembly | Hast.Samples.SampleAssembly.Posit32Calculator | CalculateIntegerSumUpToNumber(S

```
29          // 80% designs can be unstable.
30          public const int MaxDegreeOfParallelism = 5;
31
32
33          public virtual void CalculateIntegerSumUpToNumber(SimpleMemory memory)
34          {
35              var number = memory.ReadUInt32(CalculateLargeIntegerSum_InputInt32Index);
36
37              var a = new Posit32(1);
38              var b = a;
39
40              for (uint i = 1; i < number; i++)
41              {
42                  a += b;
43              }
44
45              var result = (int)a;
46              memory.WriteInt32(CalculateLargeIntegerSum_OutputInt32Index, result);
47          }
48
49          public virtual void CalculatePowerOfReal(SimpleMemory memory)
50          {
51              var number = memory.ReadInt32(CalculatePowerOfReal_InputInt32Index);
52              var positToMultiply = memory.ReadUInt32(CalculatePowerOfReal_InputPosit32Index);
53
```

100 %

Task Runner Explorer Error List Output

Ready Ln 33 Col 58 Ch 58 INS Add to Source Control

Hastlayer.SDK - Mi... HUN 17:05 2018. 03. 21.

Thank you for your attention!

- crew@hastlayer.com
- https://hastlayer.com
- https://github.com/Lombiq/Hastlayer-SDK
- https://hastlayer.com/arithmetics
- https://lombiq.com