Towards Compilation of an Imperative Language for FPGAs

Baptiste Pauget, Alex Potanin and David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

`@WhileyDave`

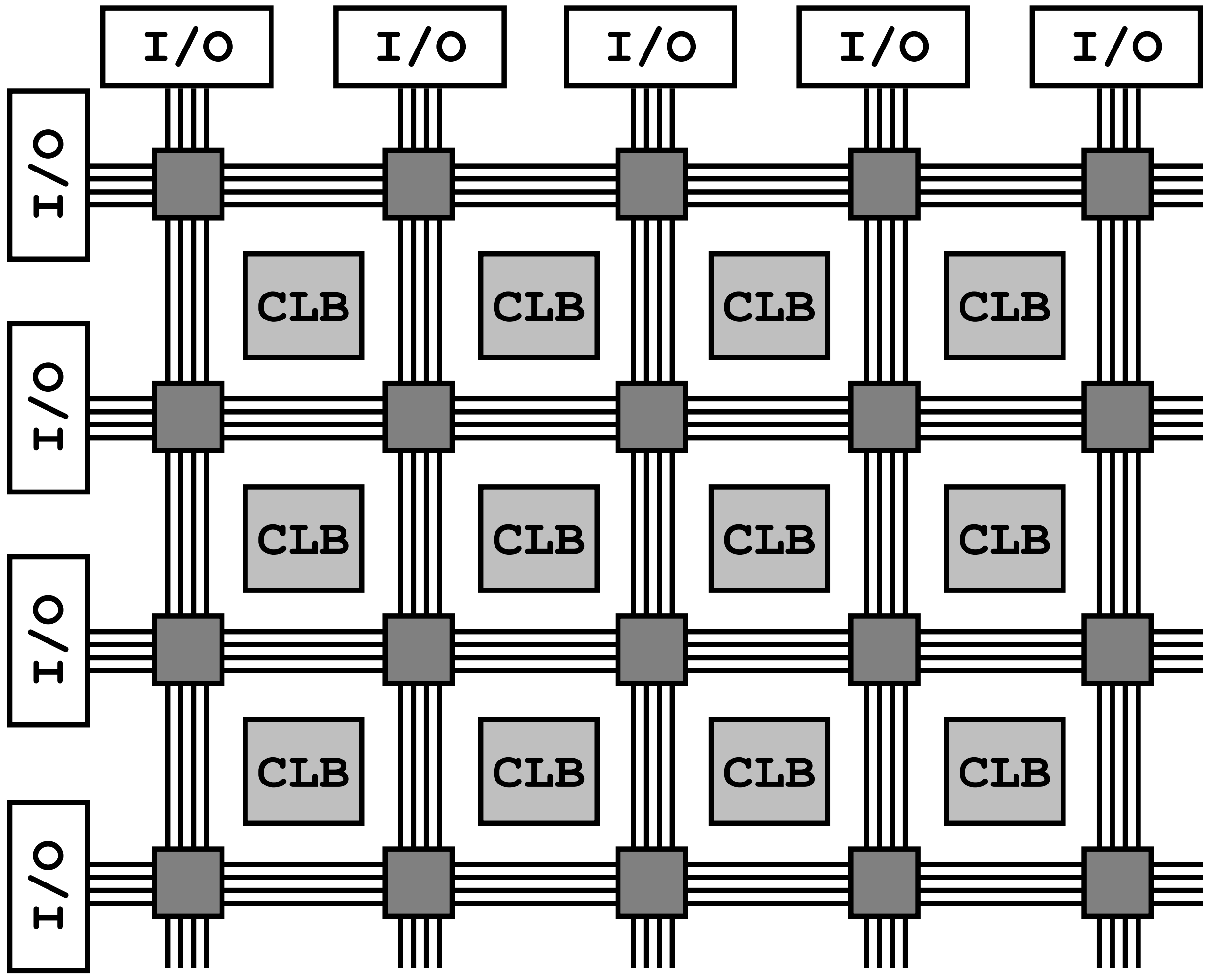
`http://whiley.org`

FPGAS: Overview

“When it comes to power efficiency (performance per watt), however, both CPUs and GPUs **significantly lag** behind FPGAs” – *Bacon* et al.

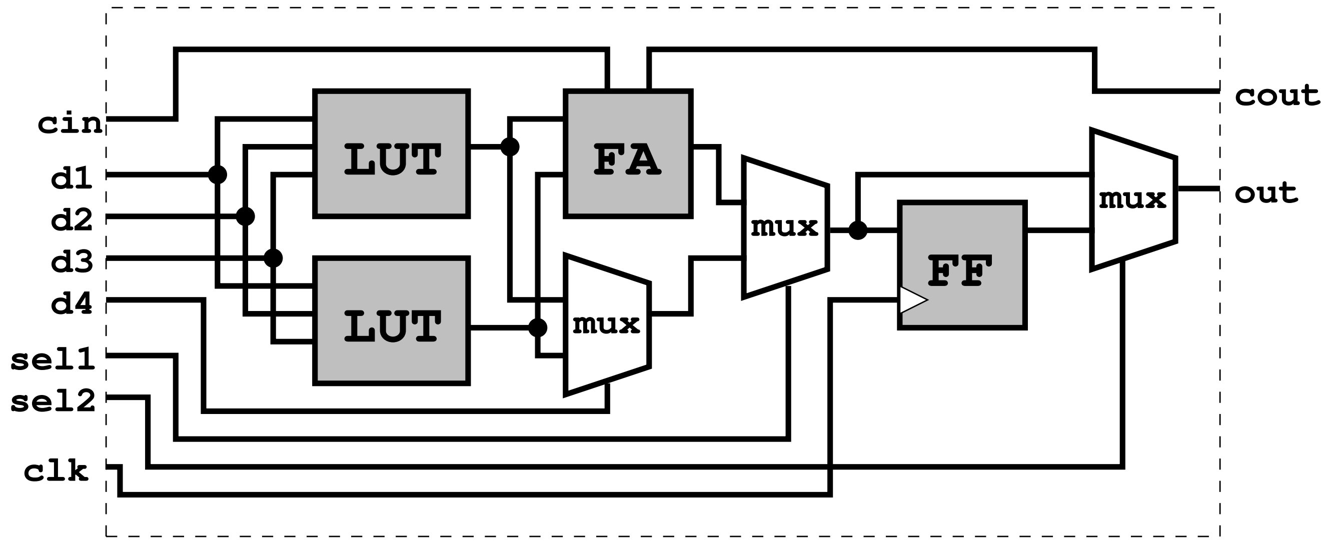
“A key element to obtaining high-performance on FPGAs is to use as **many slices** *as possible for parallel computation. This can be achieved by* **pipelining** *the blocks, trading latency for throughput; by data-parallelism, where data-paths are replicated; or a combination of both”* –*Brodtkorb* et al.

FPGAs: What are they?



- Typically will have **millions** of blocks

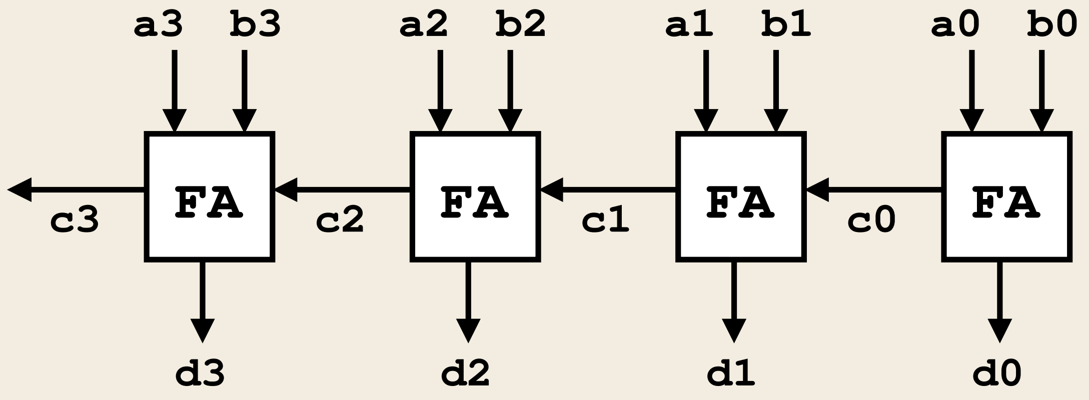
FPGAs: Configurable Logic Blocks (CLBs)



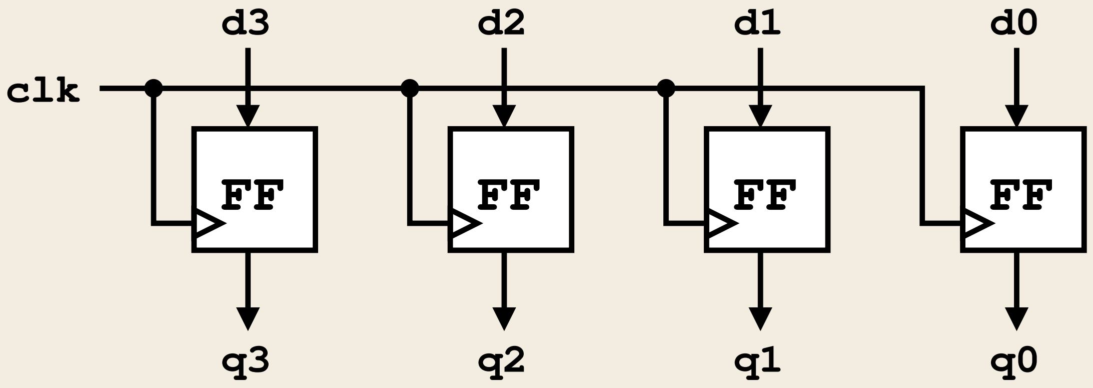
- **Look Up Tables**. With **3 bits**, can implement single gate
- **Storage**. D-type flip flops provide **1 bit** storage
- **Full Adder**. Provides **1 bit** addition with carry
- **Multiplexors**. Allow for different configurations

FPGAs: Building Blocks

Ripple Carry Adder

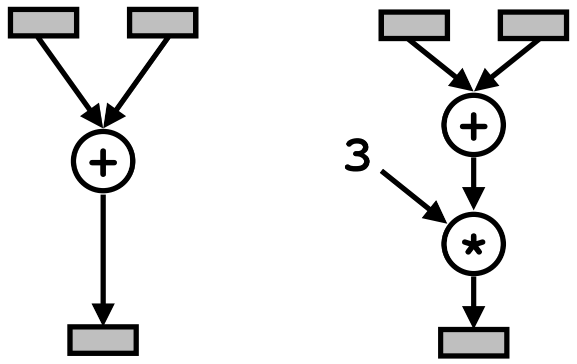


Register



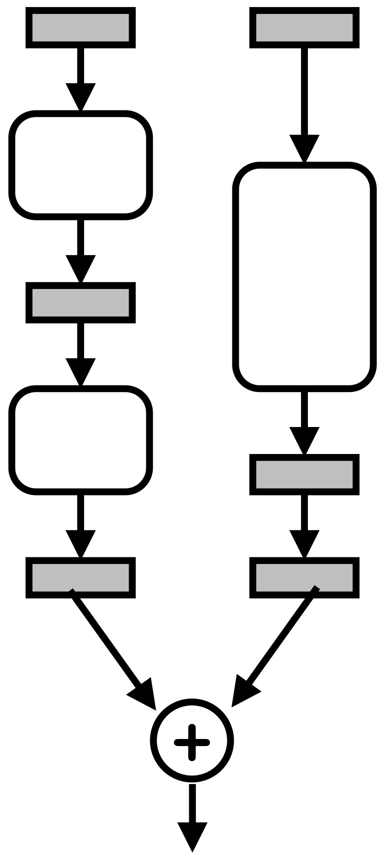
- Different implementations with various **trade offs** available

FPGAs: Register Pipelining



- Datapath **width** needs to be determined
- Max clock frequency determined by **minimum gate delay**
- Can clock **left-most** circuit faster than **right-most**

FPGAs: Pipelining & Balancing

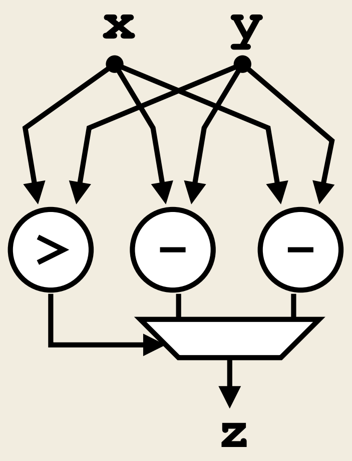


- Combining different data paths requires **balancing them**

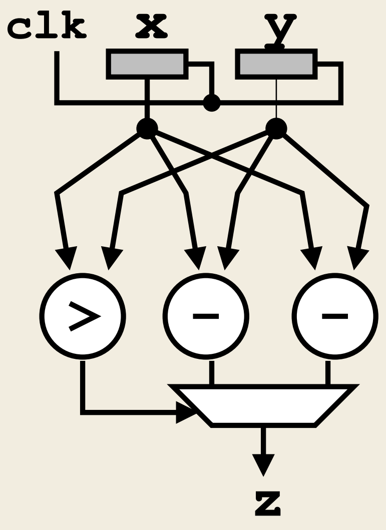
Q) *How to compile for an FPGA?*

Conditionals

```
function diff(int x, int y) → (int z):
    if x > y:
        return x - y
    else:
        return y - x
```

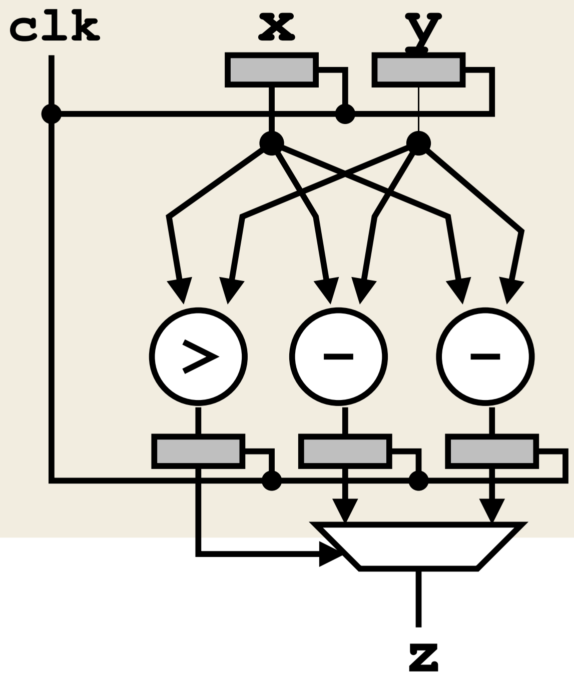


```
function diff(int x, int y) → (int z):
    stage
    if x > y:
        return x - y
    else:
        return y - x
```



Balancing

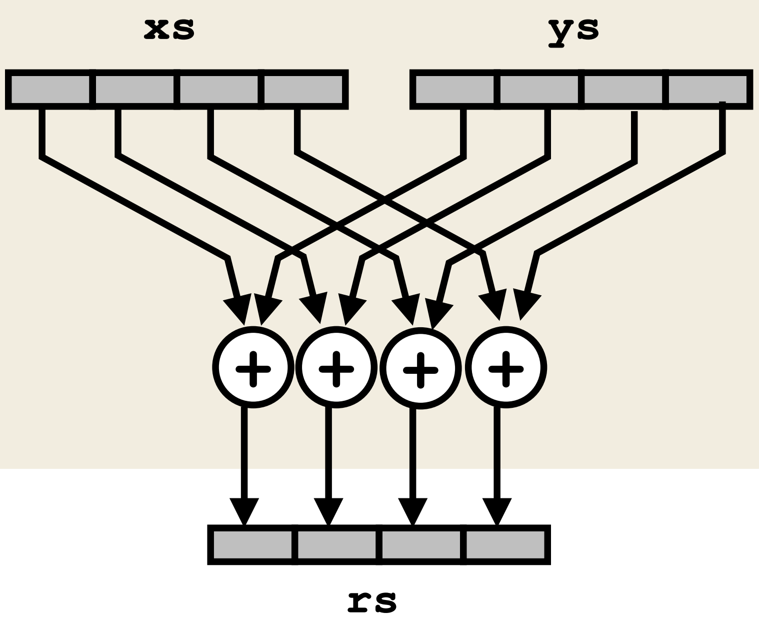
```
function diff(int x, int y) → (int z):
    stage
    if x > y:
        int tmp = x - y
        stage
        return tmp
    else:
        return y - x
```



- Placement of registers must be **balanced**
- Otherwise, data values produced at **different rates**

Data Parallelism

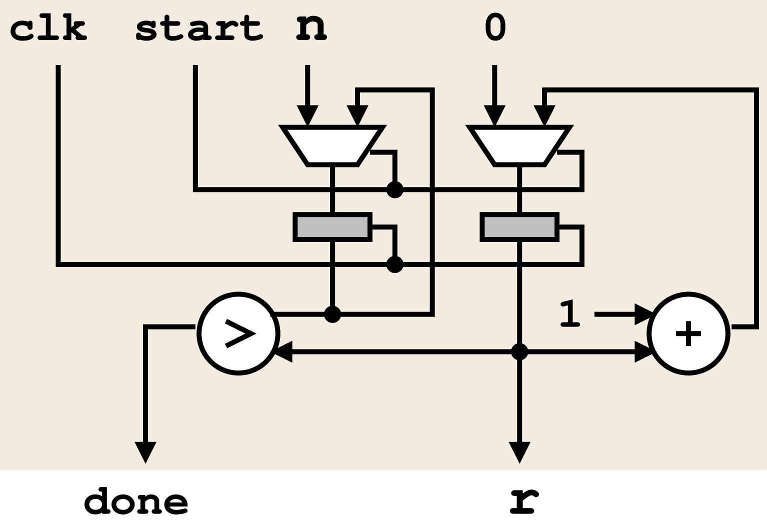
```
type vec is (i8[] vs) where |vs| == 4

function add(vec xs, vec ys) → (vec zs):
    i8 i = 0
    vec rs = [0,0,0,0]
    while i < 4:
        rs[i] = xs[i] + ys[i]
        i = i + 1
    return rs
```



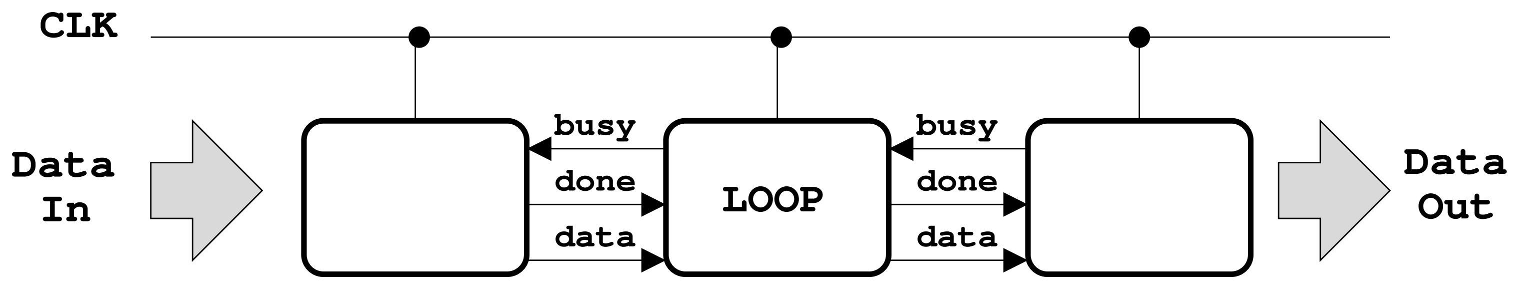
- Relies on compiler's ability to extract **data parallelism**

Task Parallelism

```
function count(int n) → (int r):
    int i = 0
    //
    while i <= n:
        i = i + 1
    //
    return i
```



- Cannot know **a priori** how many iterations and allow for **stalls**

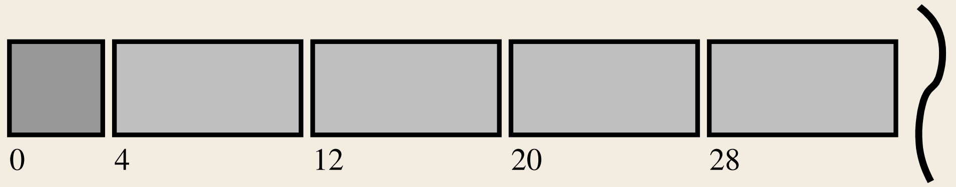


Data Types & Invariants

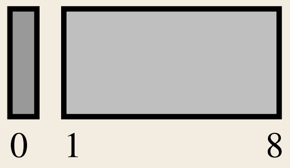
```
type i8 is (int x) where x >= -128 && x <= 127
```

```
type Point is { i8 x, i8 y }
```

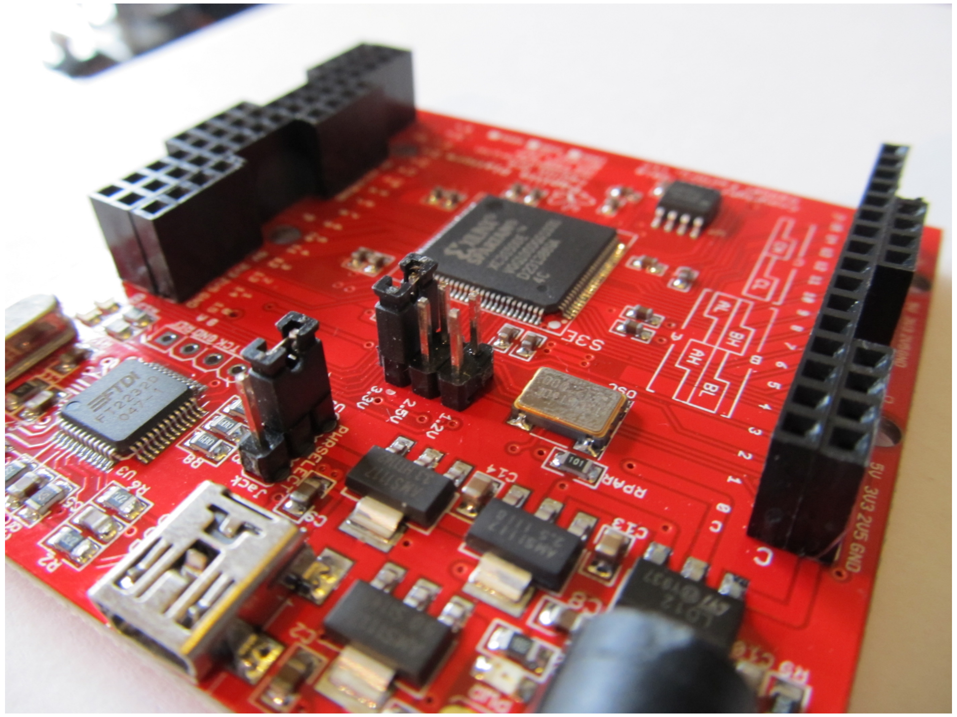
```
type string is (i8[] s) where |s| < 10
```



```
type ni8 is (i8|null)
```



Testing



- **Papilio One** 500K which utilises Xilinx Spartan 3
- Used in conjunction with **LogicStart MegaWing**

http://whiley.org

@WhileyDave
http://github.com/Whiley