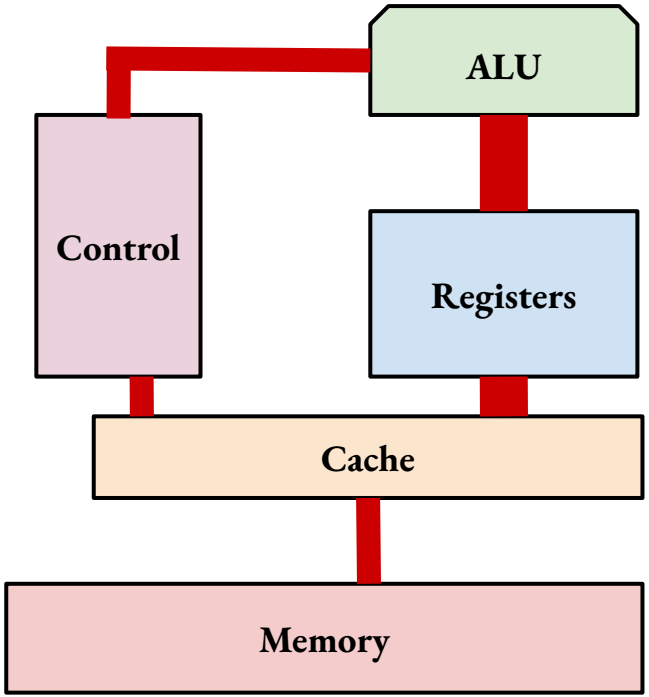
Scalable Pipeline Parallelism on FPGAs with High-Level Synthesis

May 28, 2025

Load Store Architecture -

Load - Store (“von Neumann”)



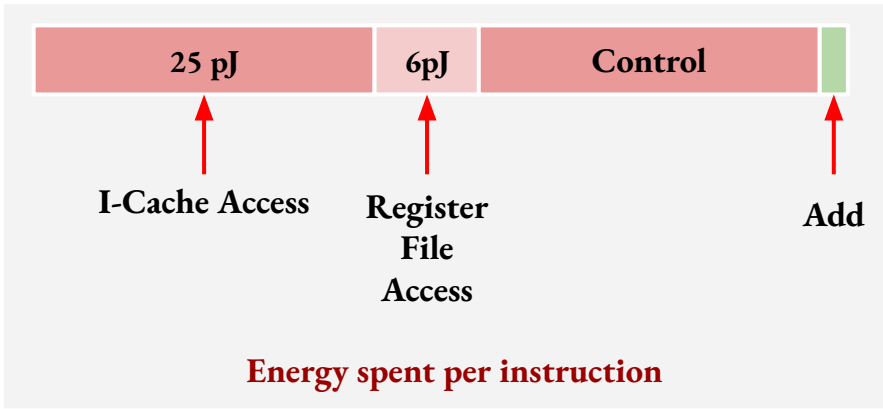

```
1: ld a, r1
2: ld b, r2
3: add r1, r1,
r2
4: st c, r1
```

Addition of two numbers

32-bit FP Add: 0.9 pJ
32-bit FP Mul: 3.2 pJ

2x32 bit from L1 (8 kiB): 10 pJ
2x32 bit from L2 (1MiB): 100 pJ
2x32 bit from DRAM: 1.3 nJ

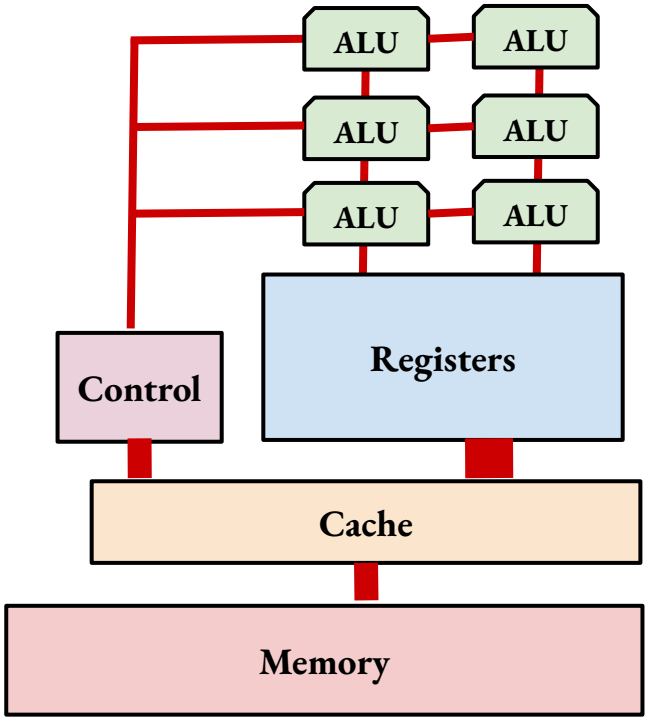
*Electrical data movement loss ~ resistance * length / cr.area*



[1] Marc Horowitz: Computing’s Energy Problem, ISSC 2014

Load Store Architecture -

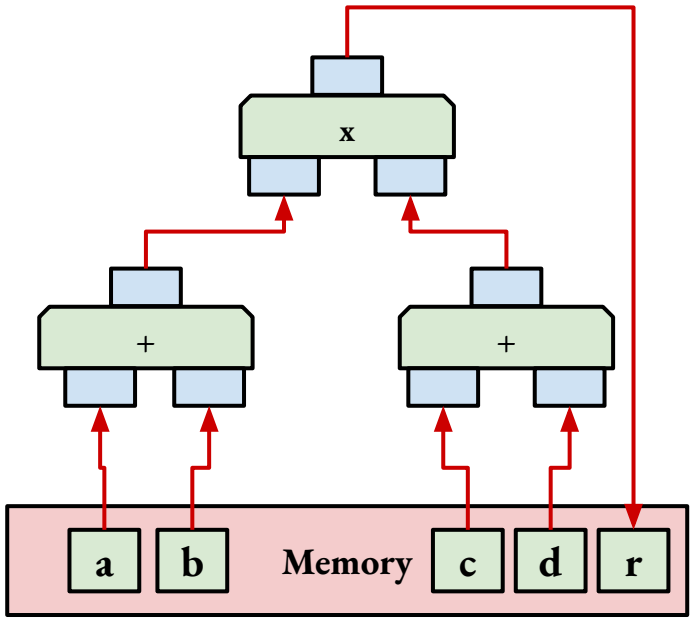
SIMD / SIMT



Dataflow:

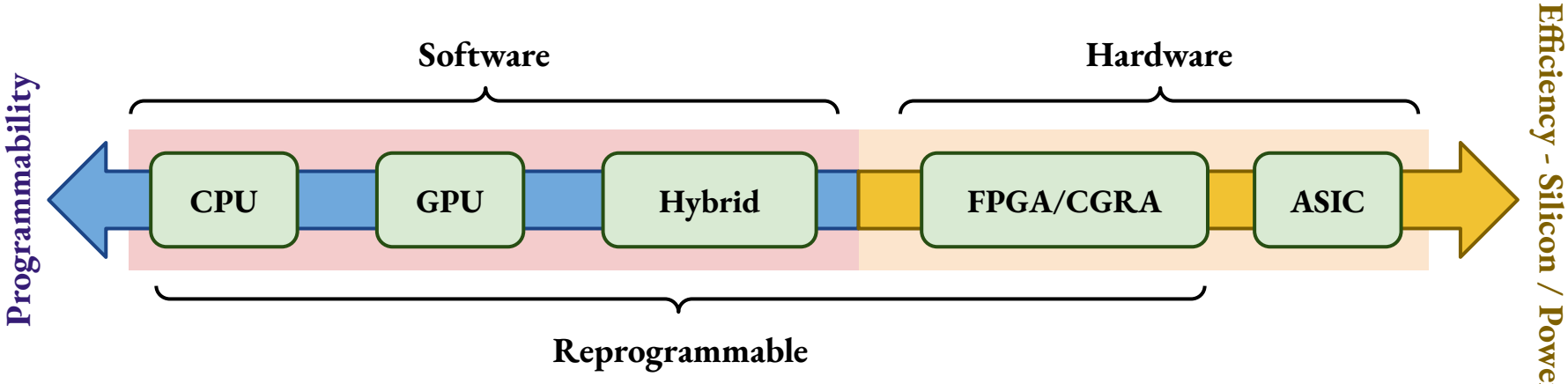
45 nm 0.9V [1]

32-bit: 0.1 pJ



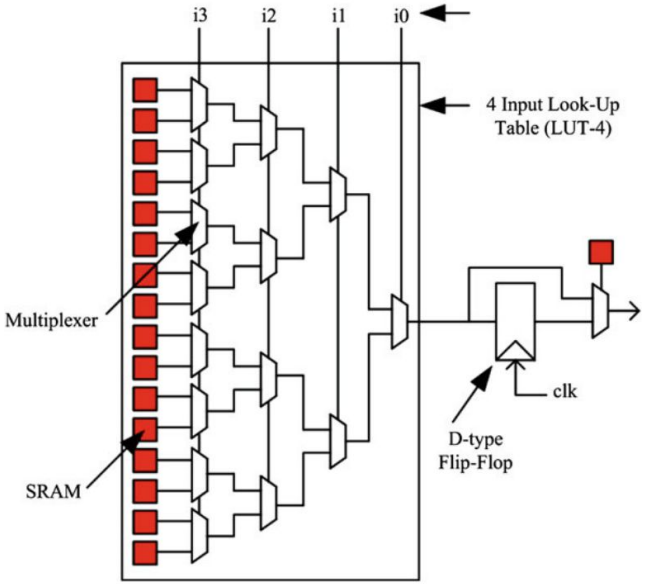
[1] Marc Horowitz: Computing's Energy Problem, ISSC 2014

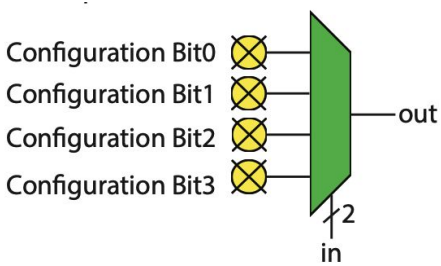
Accelerators for High Performance Computing -



Reducing data movement is a challenge for obtaining higher performance

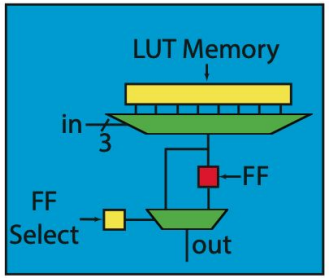
Field Programmable Gate Arrays (FPGA)





in[1]	in[0]	out
0	0	0
0	1	0
1	0	0
1	1	1

out = in[1] & in[0]

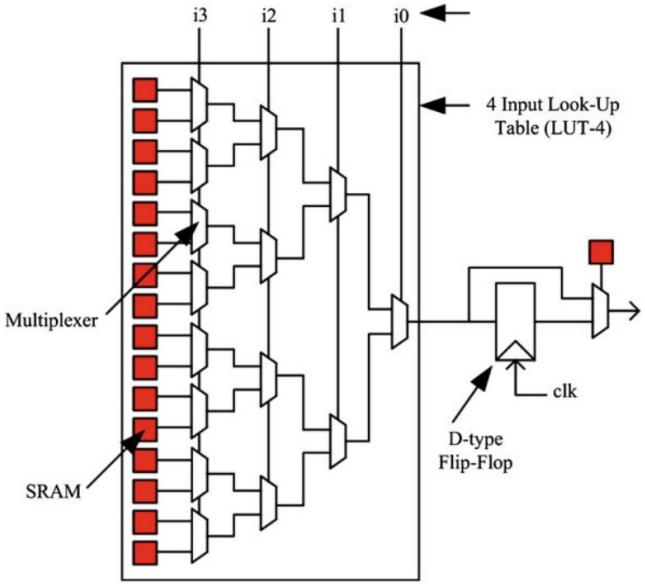


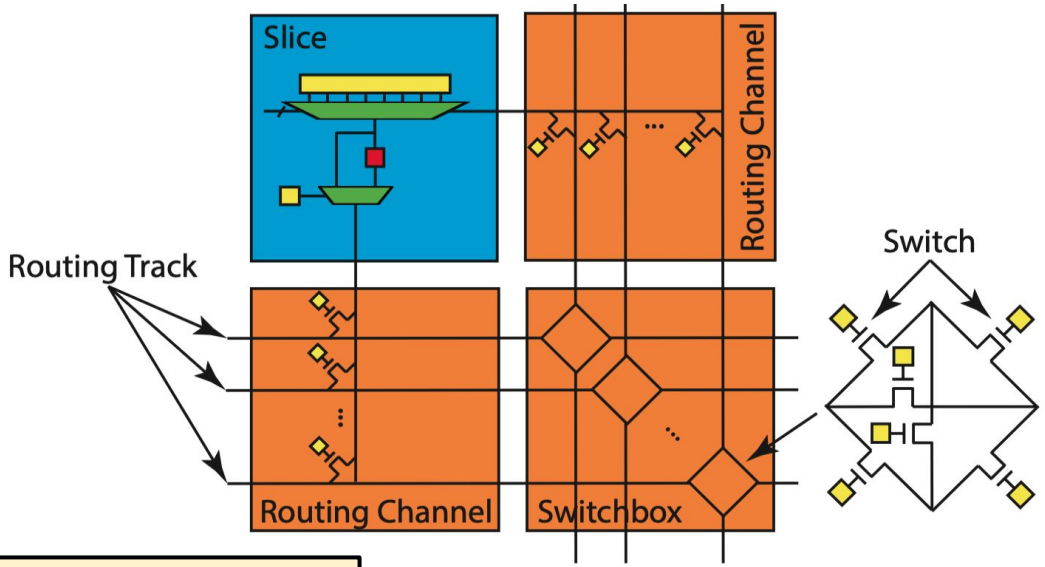
Configuration logic block

Implementing a circuit on the FPGA is about filling the red memory cells / configuration bits.

Images: Parallel Programming for FPGAs https://kastner.ucsd.edu/hlsbook/

Field Programmable Gate Arrays (FPGA)

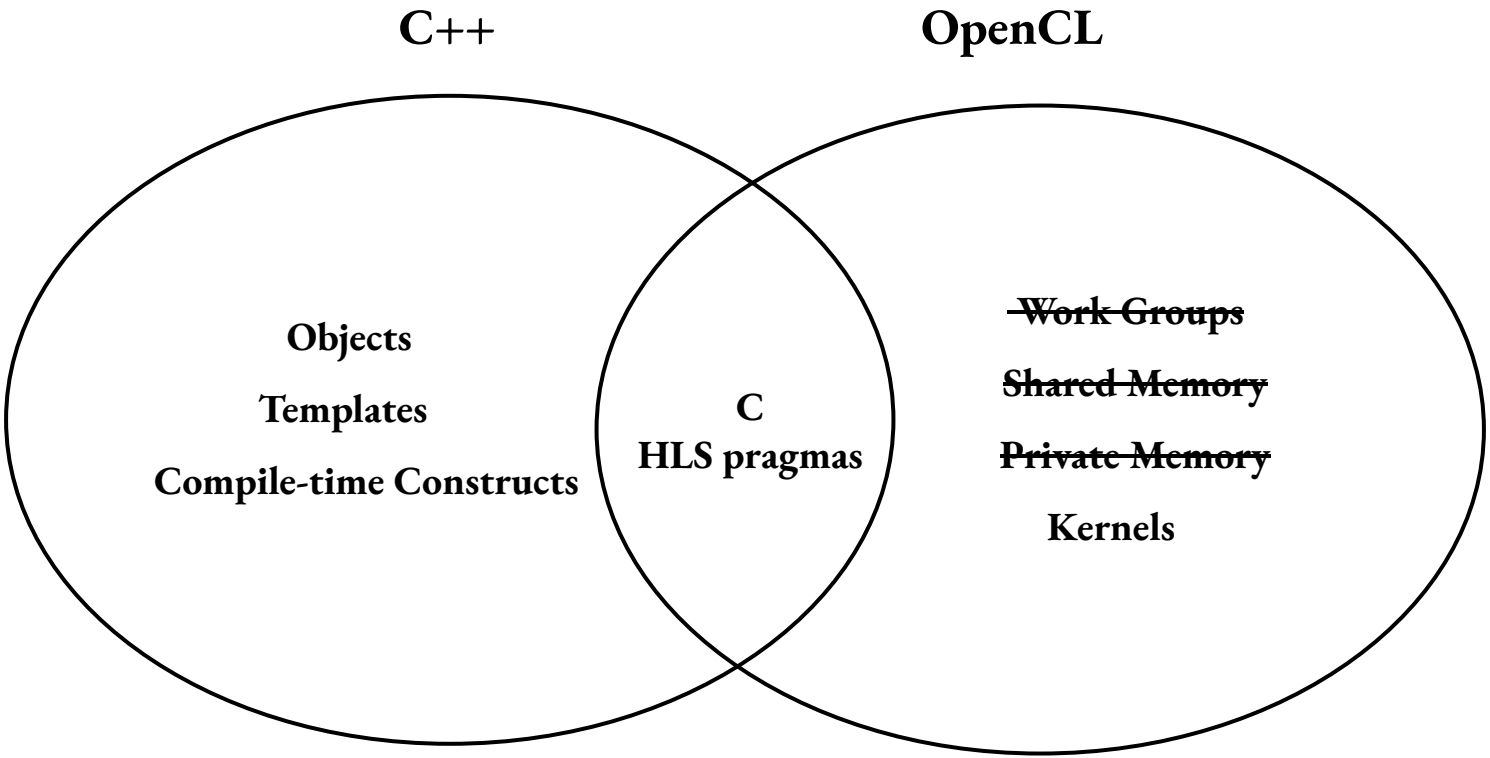




Implementing a circuit on the FPGA is about filling the red memory cells / configuration bits.

Images: Parallel Programming for FPGAs https://kastner.ucsd.edu/hlsbook/

Programming FPGAs with HLS

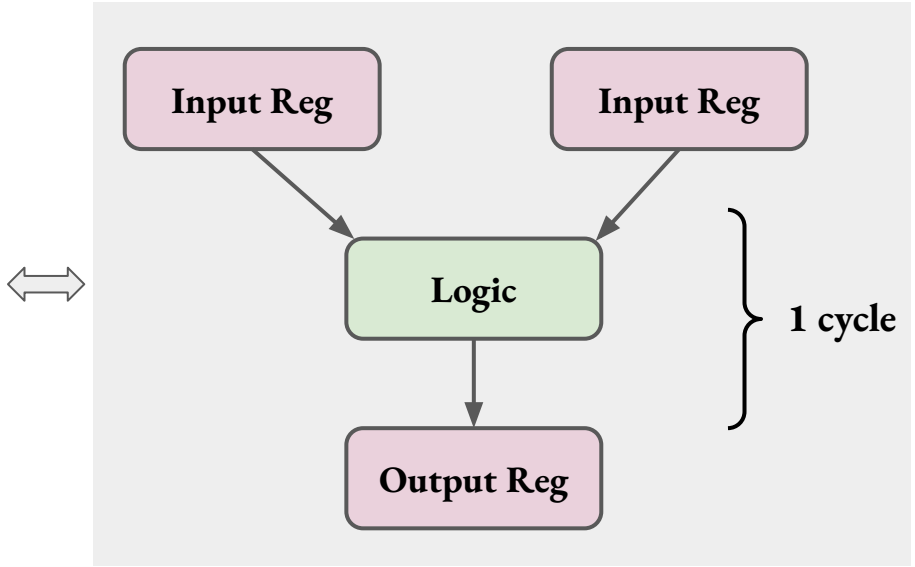


End Goal: Perform compute in every piece of available logic in every cycle.

Register Transfer Level

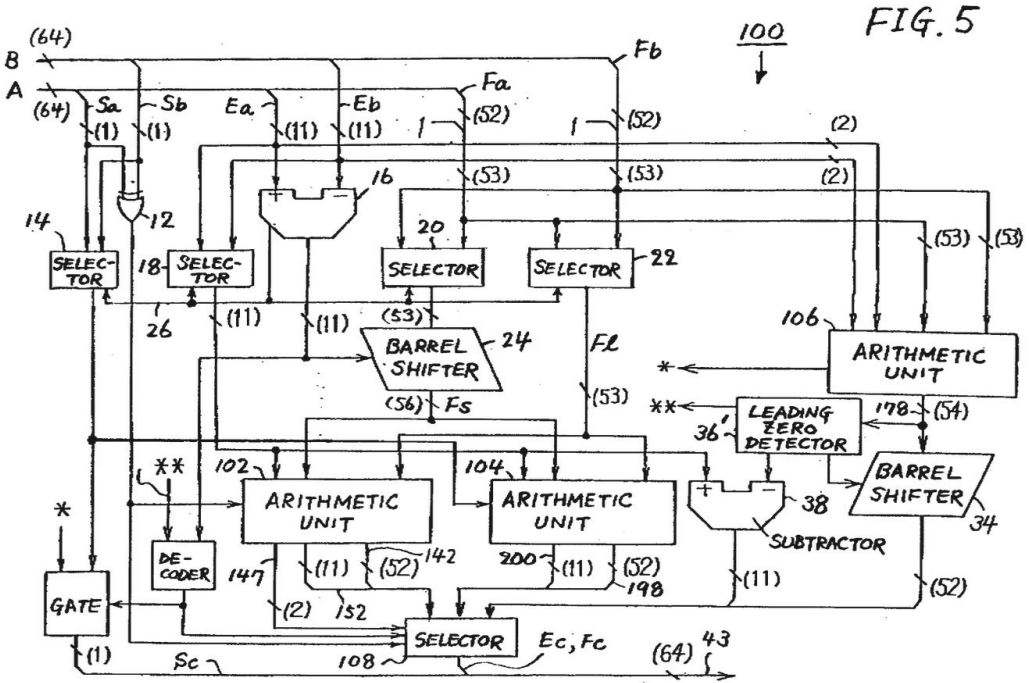
```
int c = a + b;
```

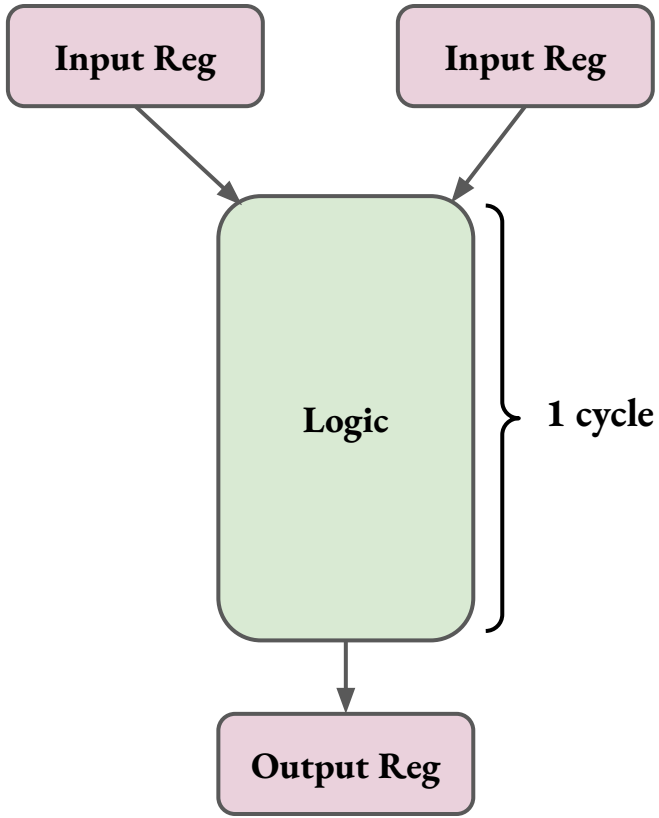
```
always @(posedge clk)
    if (start) begin
        c <= a + b;
    end
```



Register Transfer Level

```
float c = a + b;
```

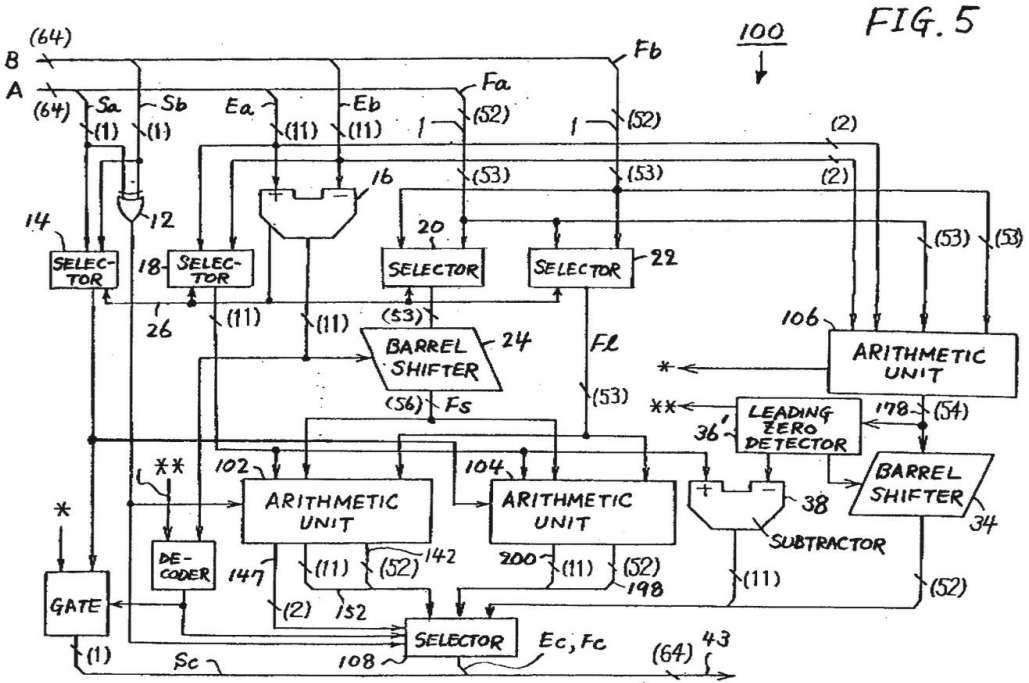


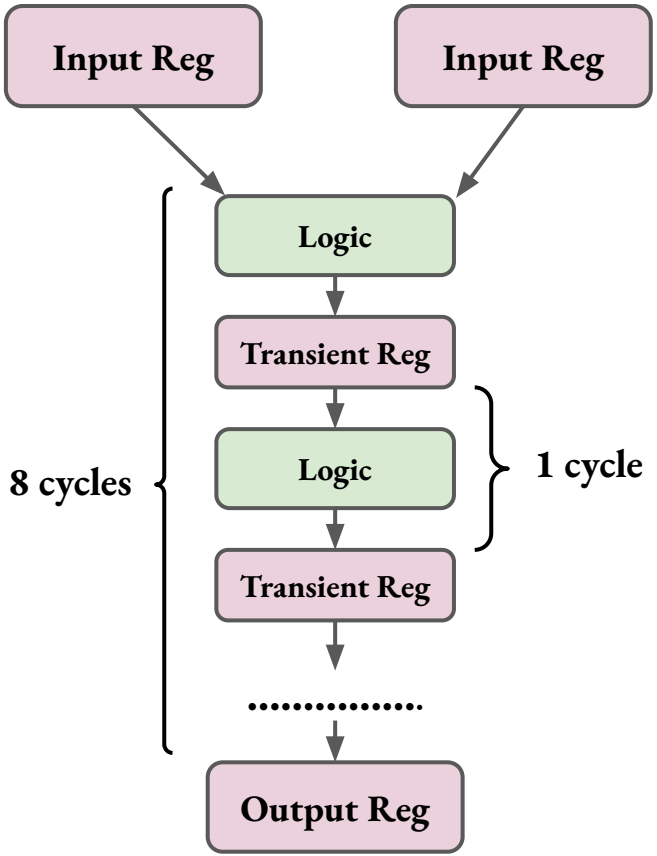


Patent EP0483864A2: ***Hardware arrangement for floating-point addition and subtraction*** **(Takashi Nakayama)**

Register Transfer Level

```
float c = a + b;
```

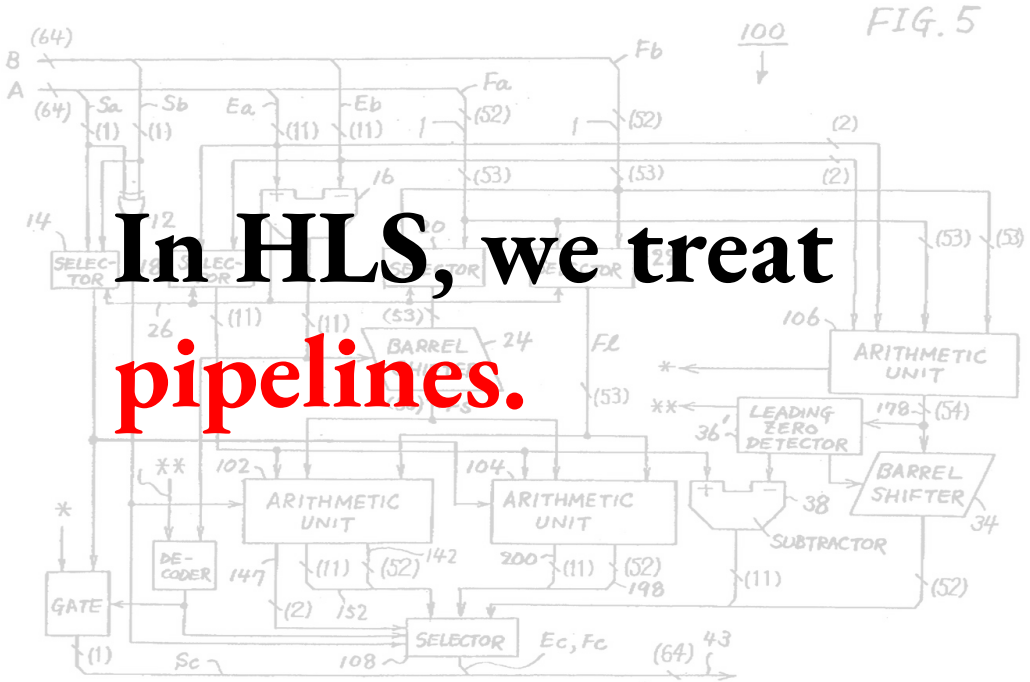




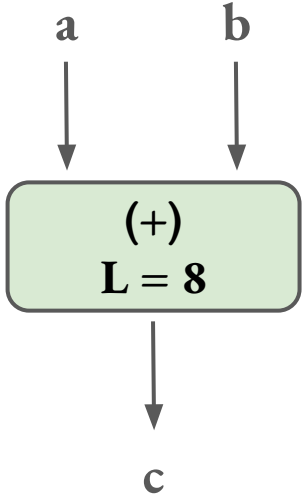
Patent EP0483864A2: ***Hardware arrangement for floating-point addition and subtraction*** **(Takashi Nakayama)**

Register Transfer Level

```
float c = a + b;
```

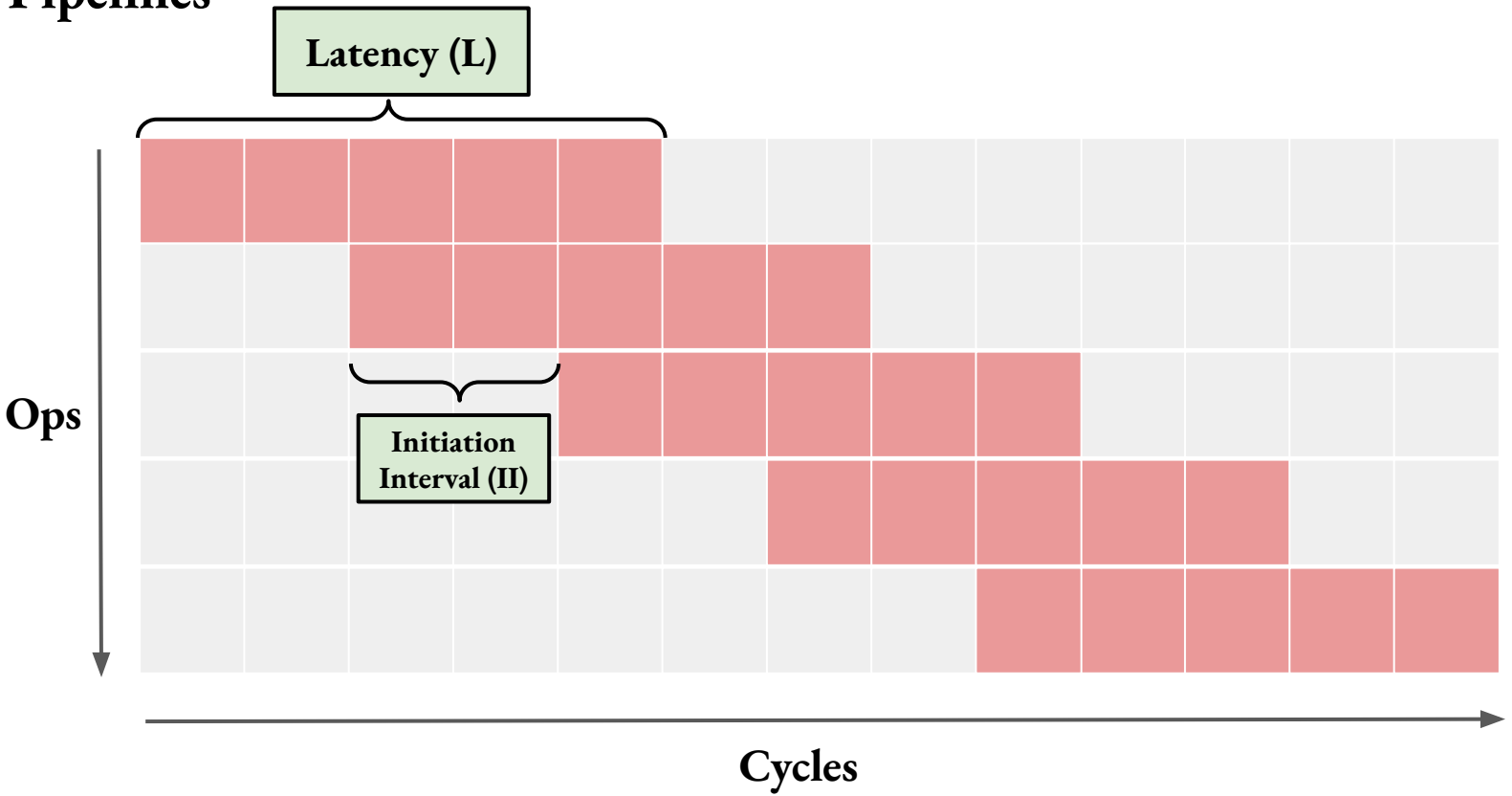


In HLS, we treat pipelines.

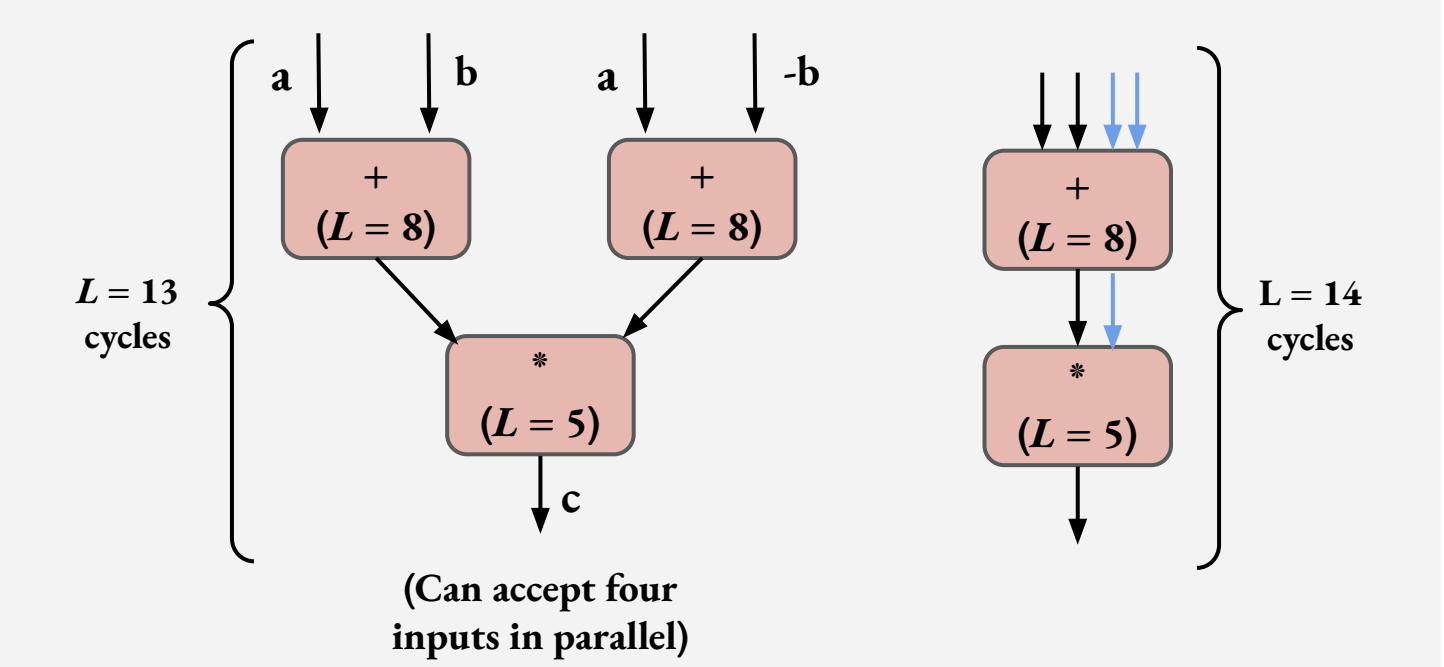


Implies *L - 1* internal pipeline stages.

Pipelines



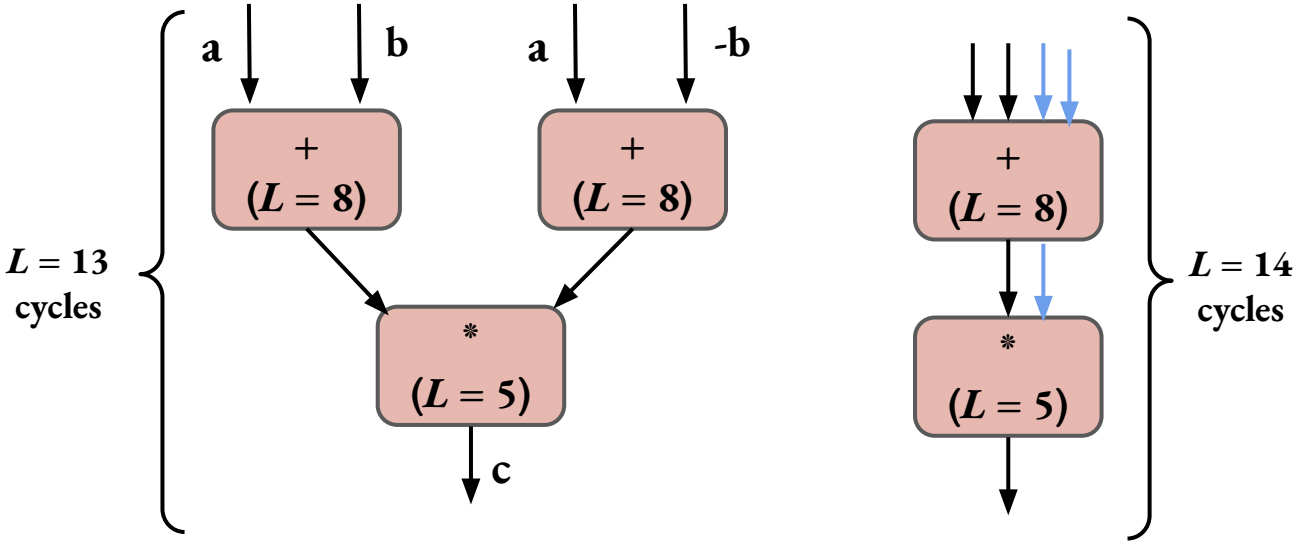
Multiple floating-point operations




```
float c = (a + b) * (a - b);
```

Two or more ways to implement this

Multiple floating-point operations



$L = 13$ cycles
$I = 1$ cycle
2 adds, 1 mult

3 ops / 1 cycle

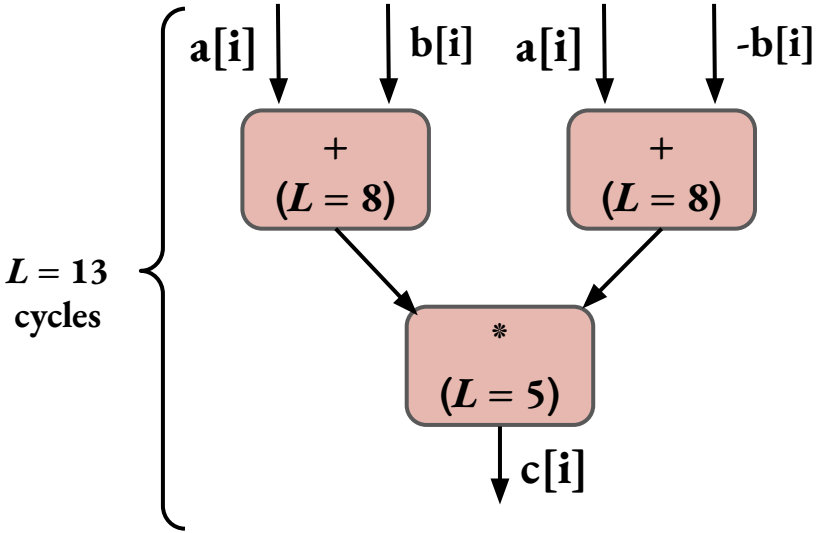
$L = 14$ cycles
$I = 2$ cycle
1 adds, 1 mult

3 ops / 2 cycles

In addition to latency (L), we have initiation interval (II):

1. No of cycles before we can accept new inputs.
2. Inverse throughput of the pipeline.
3. Factor of slowdown of the application.

Multiple floating-point operations




```
for (int i = 0; i < N; ++i) {
    #pragma HLS PIPELINE II=1
    c[i] = (a[i] + b[i]) *
           (a[i] - b[i]);
}
```

$L = 13$ cycles
$I = 1$ cycle
2 adds, 1 mult

3 ops / 1 cycle

1 iteration	13 + 1 = 14 cycles
10 iteration	13 + 10 = 23 cycles
N iteration	13 + N cycles

(Loops iterations affect the running time additively)