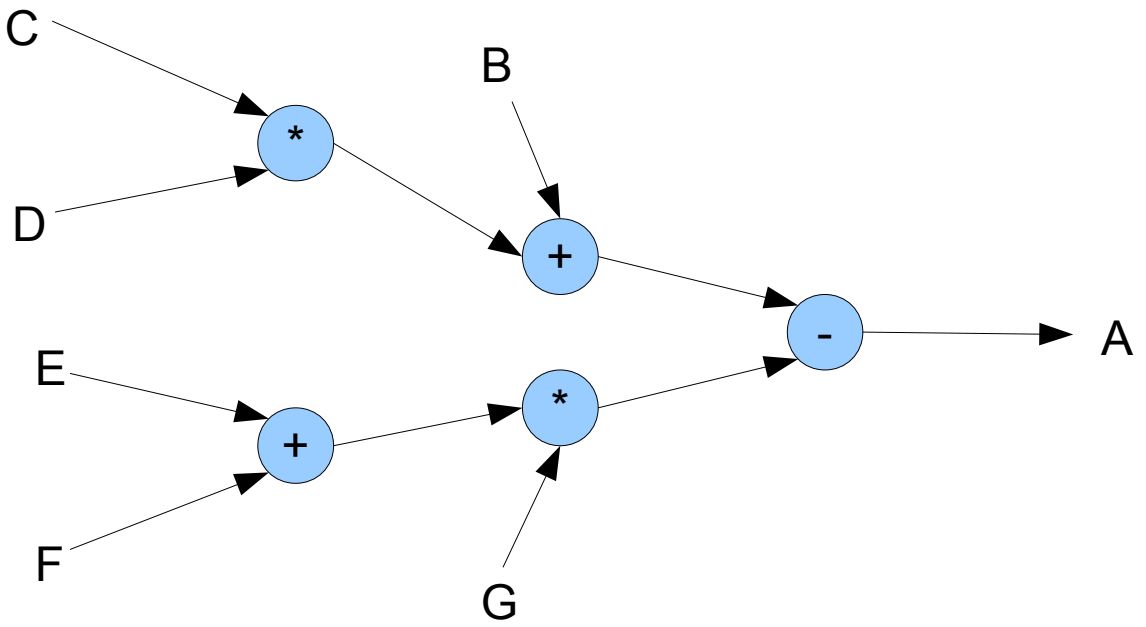
Datapath Synthesis

Grzegorz Jabłoński

Based on „TLM-Driven Design and Verification Methodology” by B. Bailey et al.

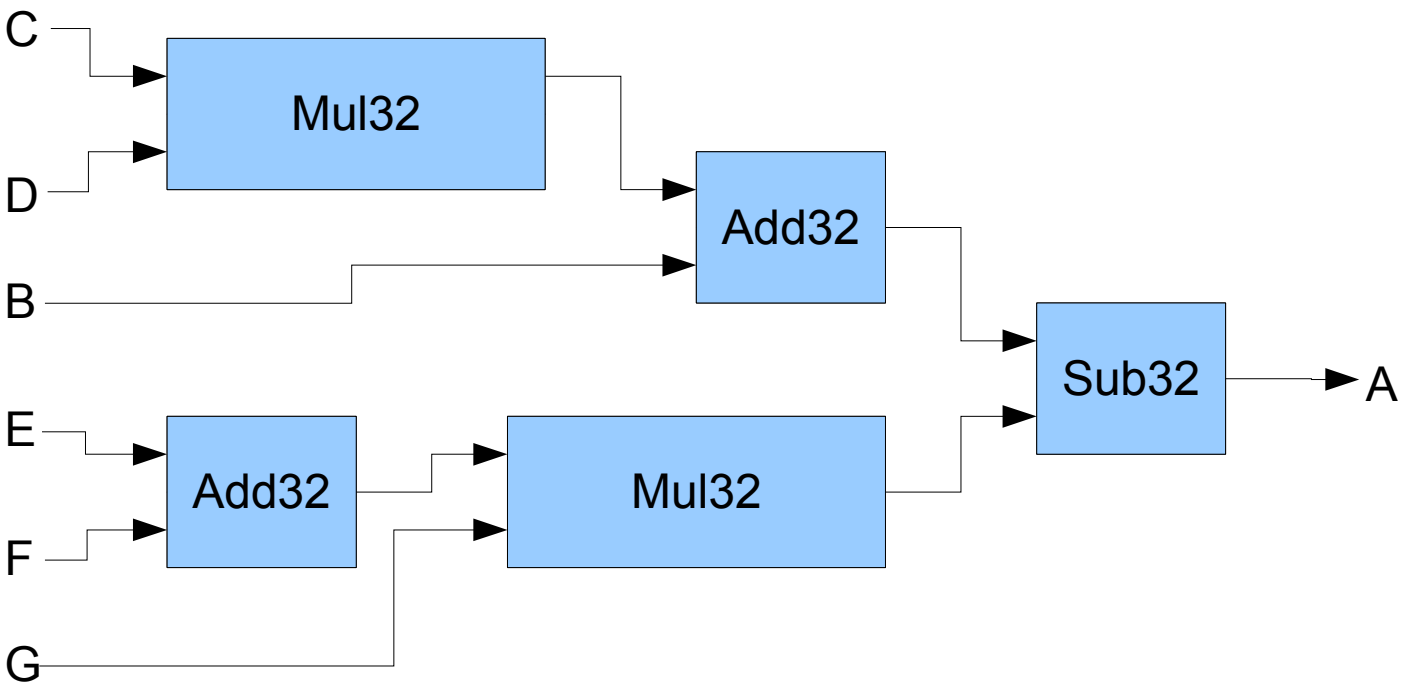
Data Flow Graph

A = B + (C * D) – (E + F) * G

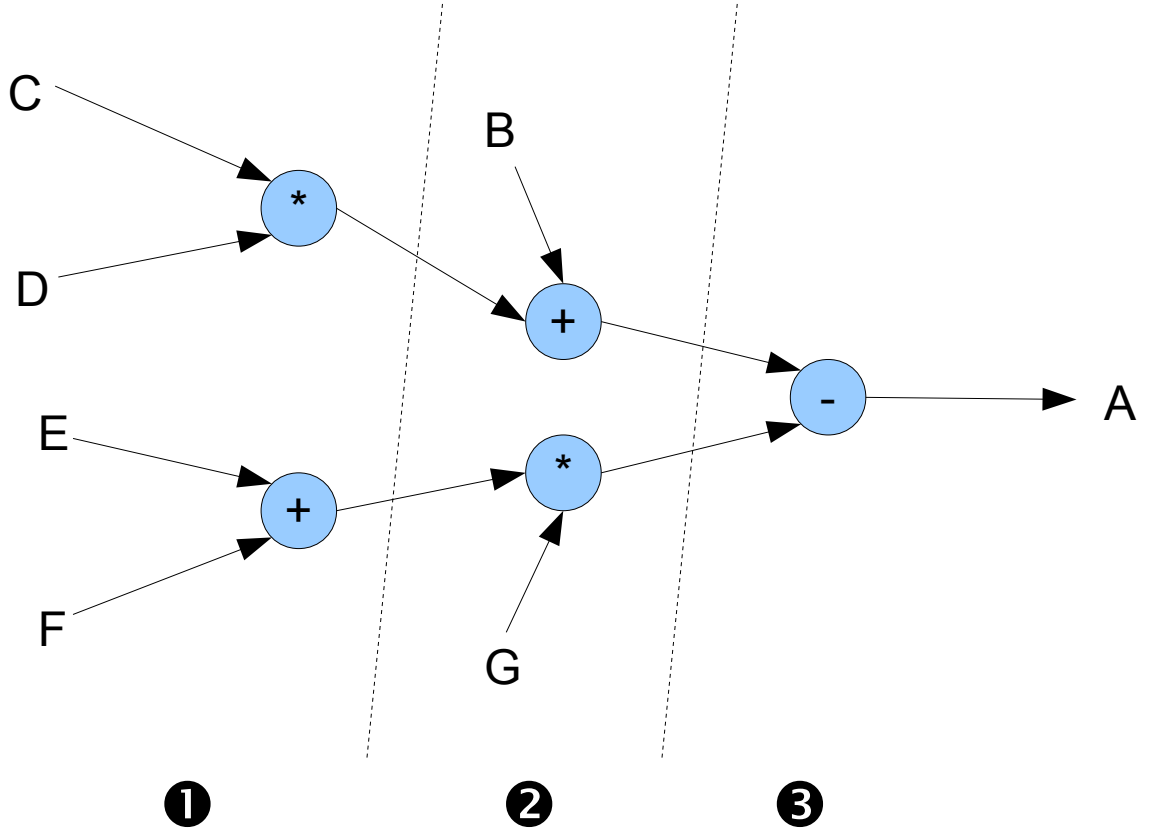



```
T1 = C  * D;
T2 = E  + F;
T1 = T1 + B;
T2 = T2 * G;
A  = T1 – T2;
```

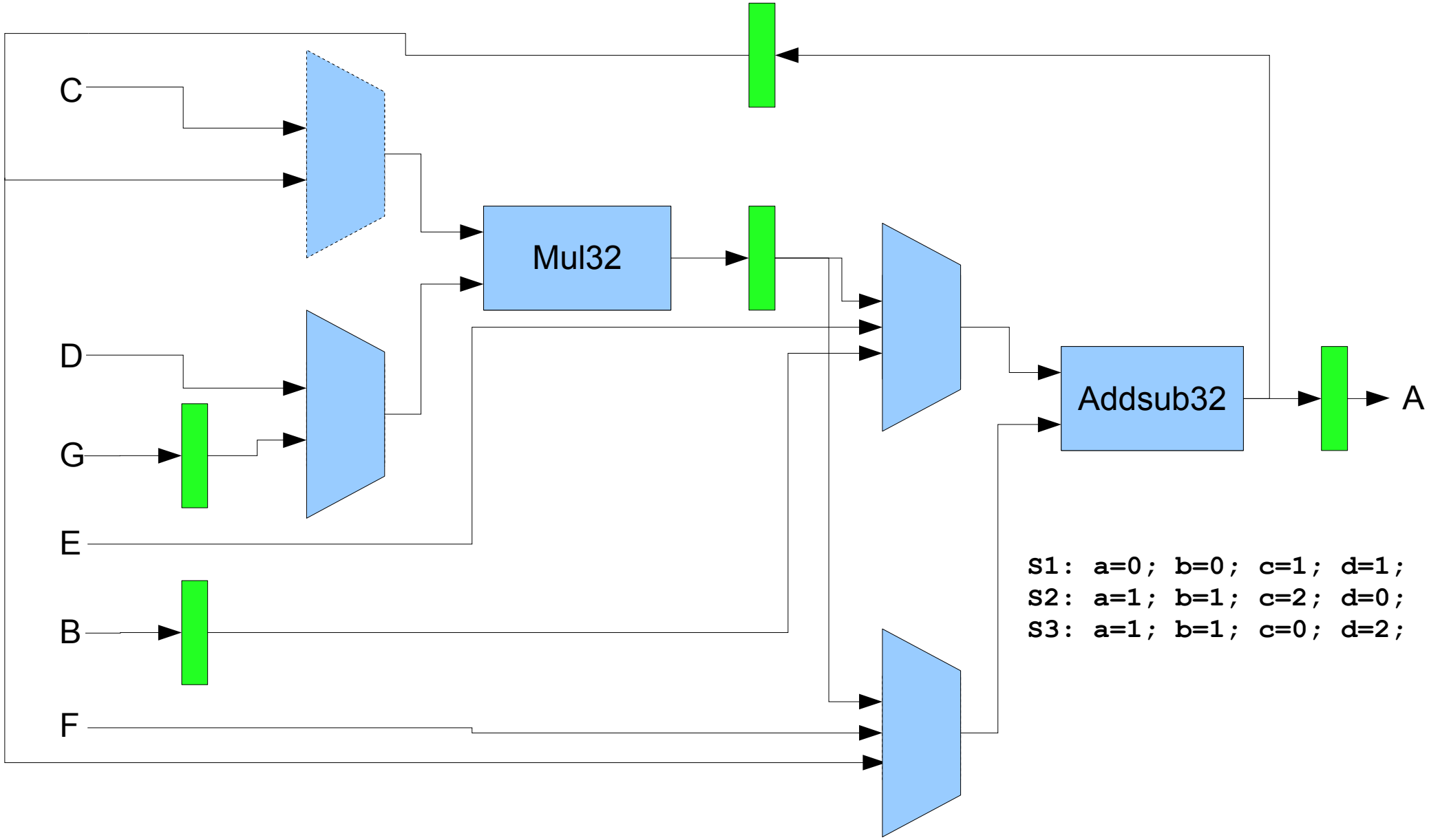
Fast Micro-architecture



Resource Sharing



Three-cycle Micro-architecture



Combinational loops

```
for (i=0; i<5; i++)
  array1[i] = input[i].read * array2[i];
}
```

```
array1[0] = input[0].read * array2[0];
array1[1] = input[1].read * array2[1];
array1[2] = input[2].read * array2[2];
array1[3] = input[3].read * array2[3];
array1[4] = input[4].read * array2[4];
```

```
for (i=0; i<5; i++)
  array1[i] = input[i].read * array2[i];
  wait();
}
```

Loop unrolling

```
array1[0] = input[0].read * array2[0];
array1[1] = input[1].read * array2[1];
for (i=2; i<5; i++)
  array1[i] = input[i].read * array2[i];
}
```

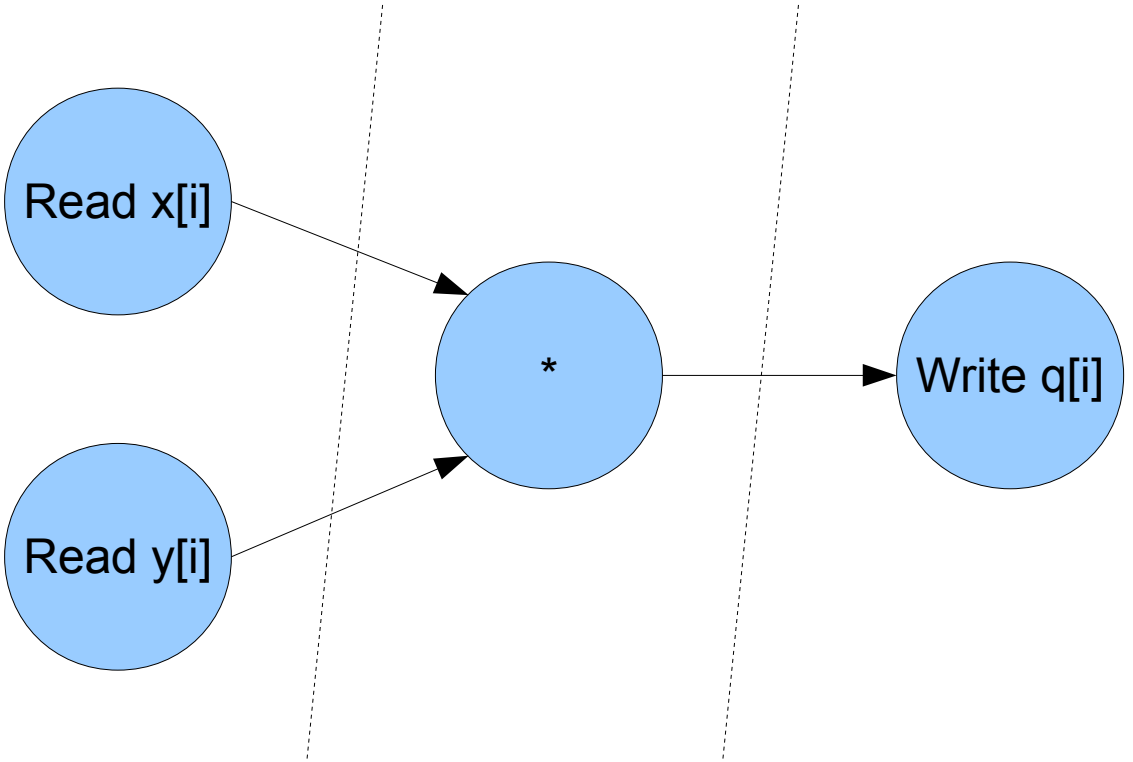
```
for (i=0; i<5; i++)
  array1[i] = input[i].read * array2[i];
  i++;
  array1[i] = input[i].read * array2[i];
  wait();
}
```

Memory Structure

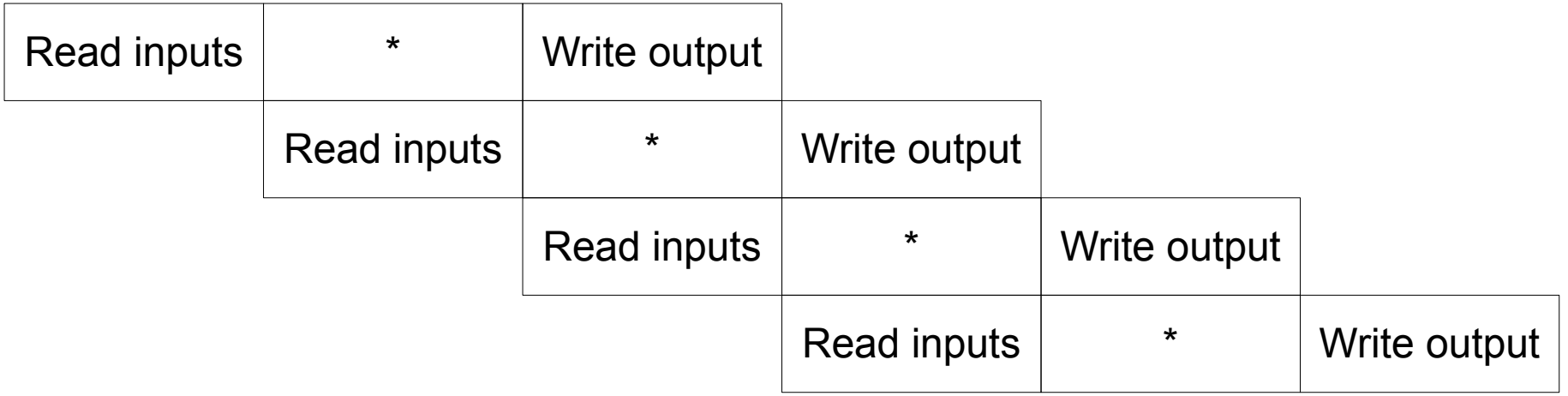
- Memories have limited number of ports
- In case of contention, it is impossible to have each loop iteration in a single clock cycle
  - Refactor the algorithm to change how it accesses memory
  - Use different memories
  - Extend latency
  - Use multiple port memories

Loop pipelining

```
for (i=0; i<32; i++)
  q[i] = x[i] * y[i];
}
```

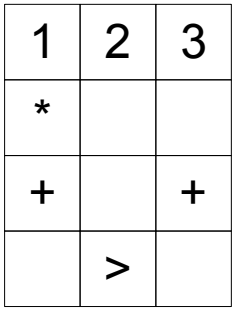


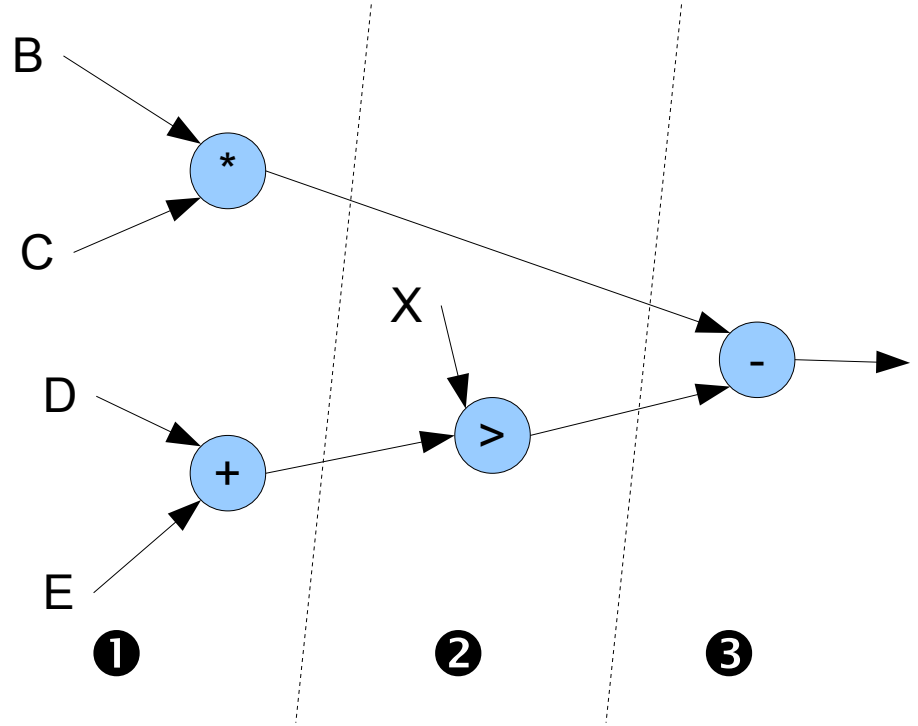
Initiation and Latency Intervals

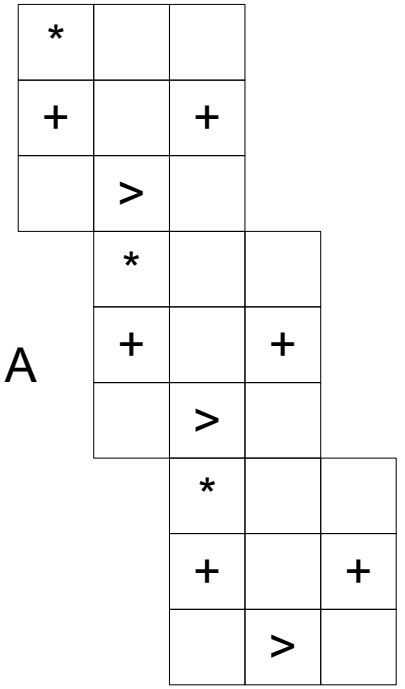


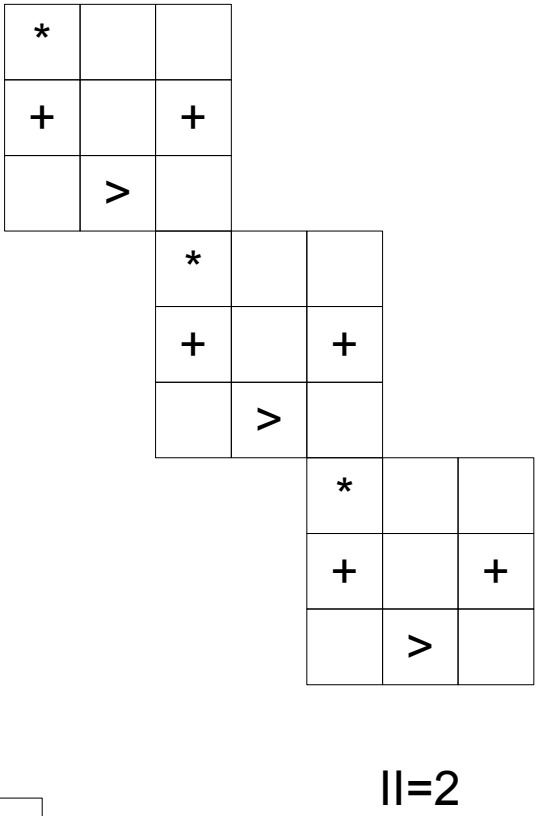
Initiation Interval

```
for (i=0; i<32; i++)
  q[i] = x[i] * y[i];
}
```



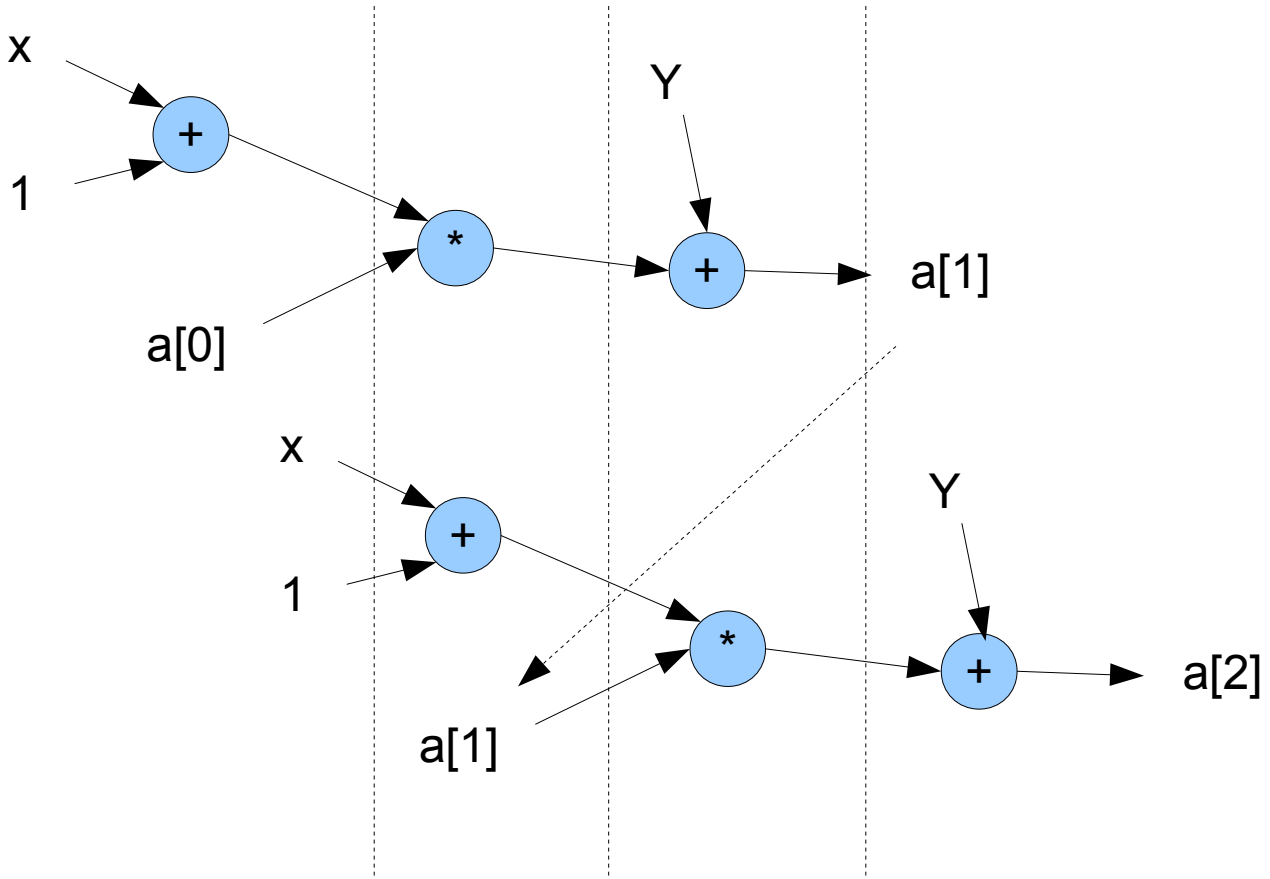






Data Dependencies, II=1

```
for (i=1; i<=32; i++)
  a[i] = Y + ((X + 1) * a [i-1]);
```



Data Dependencies, II=2

```
for (i=1; i<=32; i++)
  a[i] = Y + ((X + 1) * a [i-1]);
```