Digital Integrated Circuits and Systems

Andrej Trost

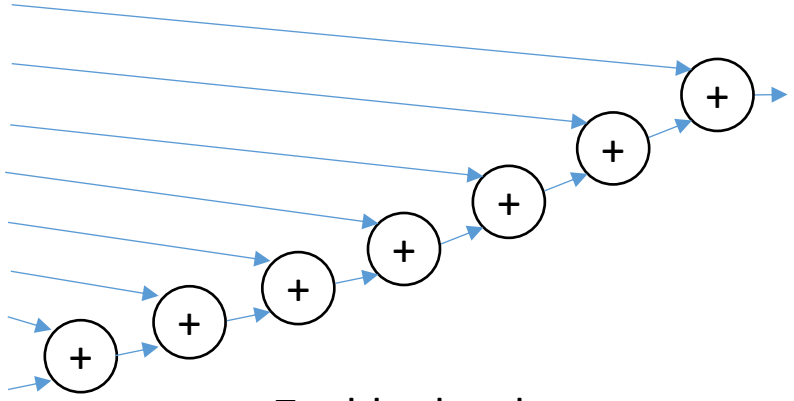
Lab 4: introduction to HLS

Tool: Vitis 2025.1
Task: Describe in C++ and synthesize pipeline circuit for arbitrary precision integer divider, check performance, use C++ (co)simulation

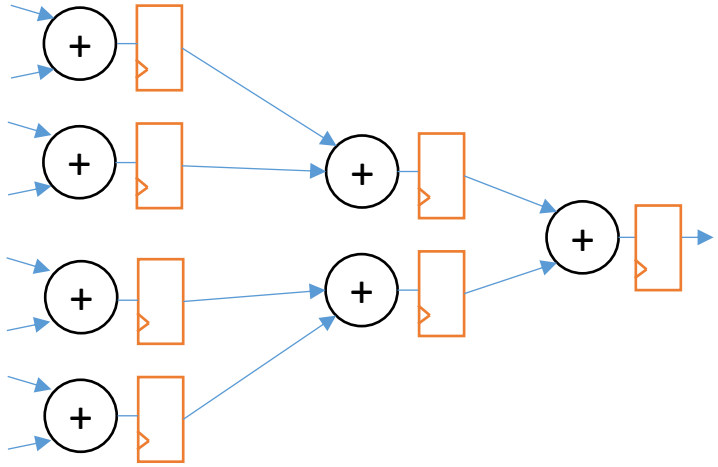
UNIVERSITY OF LJUBLJANA
FE
Faculty of Electrical Engineering

N-Tap averaging filter

- Circuit size and performance depends on algorithm description
  - better algorithm, for example for sequential input we add only new value and subtract the oldest from the sum of values
  - But general filters with weights need all the adders…
- How to make the circuit faster?
  - adder tree and pipeline to decrease logic levels



7 adder levels



1 level with pipeline, 3 cycle latency

- How to pipeline the divider?

Circuit design in (System)Verilog, VHDL...

Pro

- Hardware description languages used for precise design specification and fine control
- Supported in all FPGA and digital ASIC design tools

Con

- Usage of integers but limited support for real numbers
  - circuits computing real numbers can be very complex
- Design productivity gap
  - precise specification and verification takes time
- Circuit performance known only after implementation
  - Synthesis and implementation are complex and slow tasks

High-level circuit model

- Use ap_int datatype to specify 14-bit integers

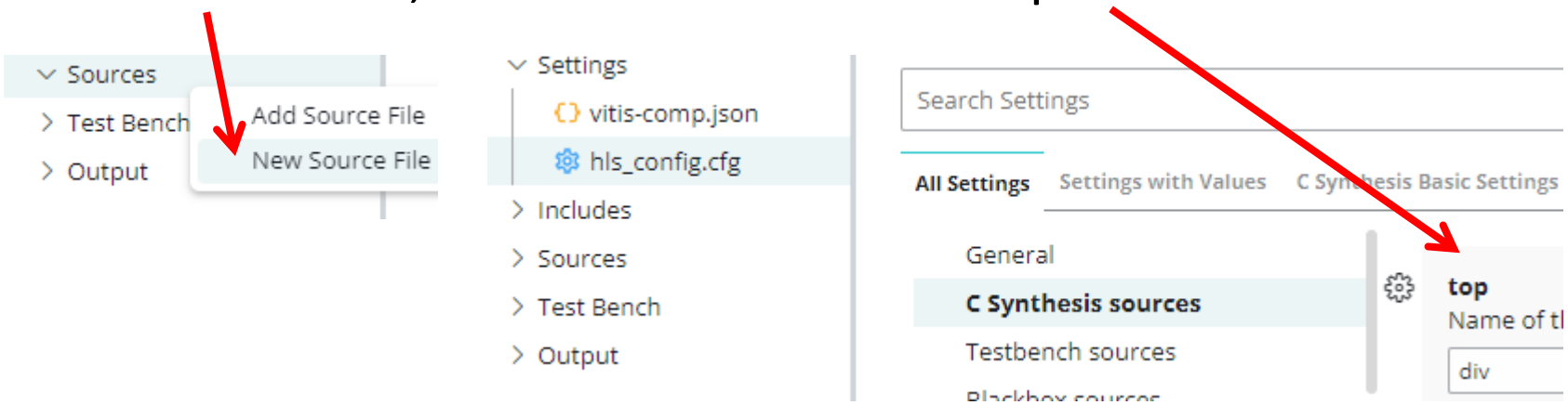
```
#include <ap_int.h>
ap_int<14> div(ap_int<14> a, ap_int<14> b)
{
 return a / b;
}
```

- use #define

```
#define DT ap_int<14>

DT div(DT a, DT b)
```

- Run Vitis 2025, HLS Development > Create (Empty) Component
  - Example name: div, location: C:\proj\div
  - Empty Config File
  - Part: xc7z007... Clock: 33MHz
- Create new Source, write code and define top level function



C Synthesis

TARGET	ESTIMATED	UNCERTAINTY
30.30 ns	2.952 ns	8.18 ns

Performance & Resource Estimates

Modules Loops H

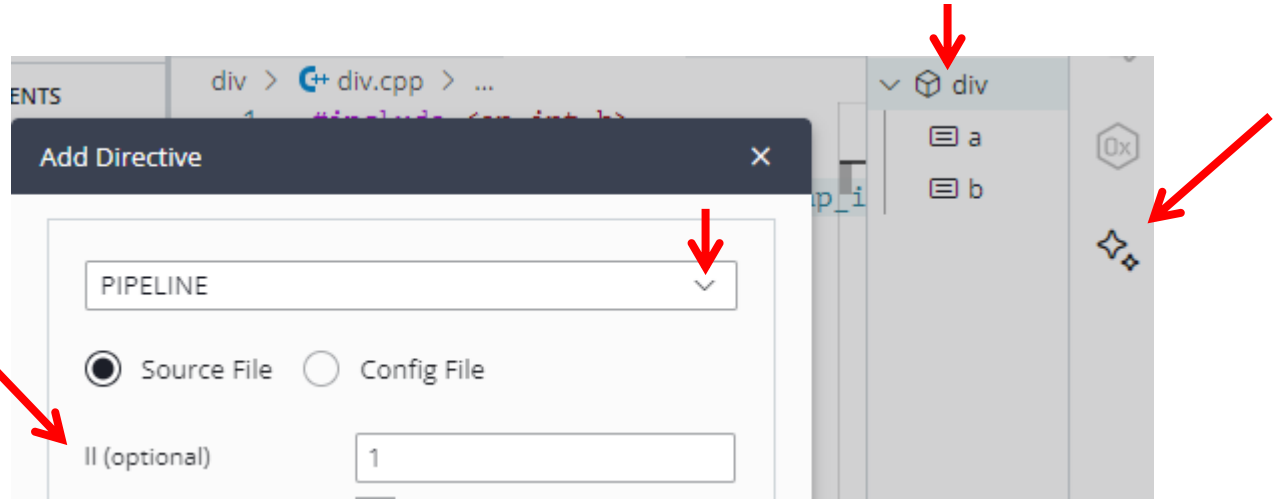
MODULES & LOOPS	LATENCY(NS)	INTERVAL	PIPELINED	STRUCTURE	BRAM	DSP	FF	LUT
div	515.000	18	no	function	0	0	196	194

- Run and check Synthesis report
  - 18 cycles for next value (sequential divider)
  - resource usage like SystemVerilog circuit model in Vivado

Name	LUT	FF
synth_1	209	42
impl_1	209	42

- Add Synthesis Directive and Run again
  - request pipeline, set Initiation Interval
  - II = 1, every clock cycle get new inputs
  - Check Synthesis report

    Latency (Cycles), Interval, FF, LUT...



Simulation

- Testbench
- Run C++ code
- check output

```cpp
#include <iostream>

int main() {
    DT x[] = {0, 10, 20, 100};
    DT y[] = {1, 2, 3, 4};

    for (int i = 0; i < 4; i++) {
        DT result = div(x[i], y[i]);
        std::cout << (int)x[i] << "/" << (int)y[i] << " = " << (int)result << "\n";
    }

    return 0;
}
```

- Check RTL operation and timing with Cosimulation
- Change data type to fixed point, for example: `ap_fixed<18,14>` and check synthesis and simulation results

bit size integer bits

circuit	clock [MHz]	latency cycles	interval cycles	LUT / FF / DSP