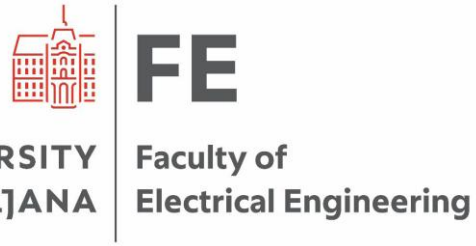
Digital Integrated Circuits and Systems

Andrej Trost

Lab 1: Basic arithmetic operations and averaging filter

Tool: Vivado 2025.1
Task: design a circuit model for basic operations, analyze implementation, make an averaging filter, analyze the implementations and verify operation with simulation



Technology & boards

AMD (Xilinx) Zynq

- FPGA + ARM Cortex-A9

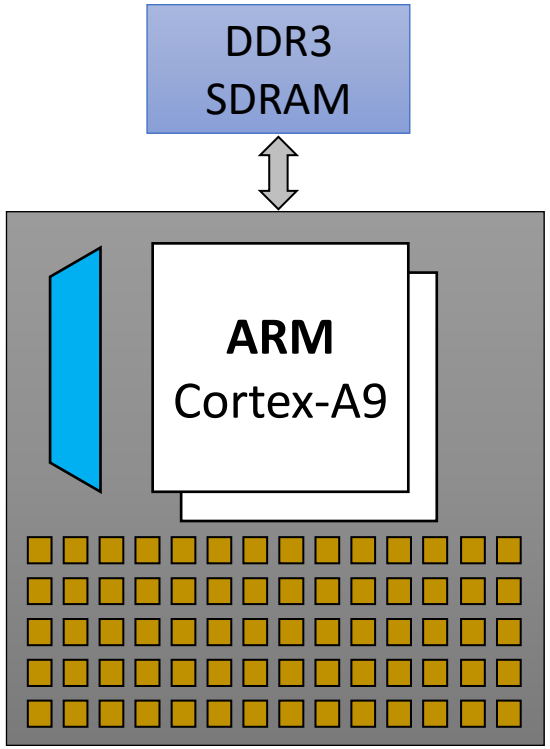
	Z-7007S	Z-7010	Z-7020
LUT (DFF)	14k (29k)	17k (35k)	53k (106k)
BRAM	200 KB	240 KB	560 KB
DSP	66	80	220
I/O	50-100	100	200

MiniZed
Cora Z7

RedPitaya

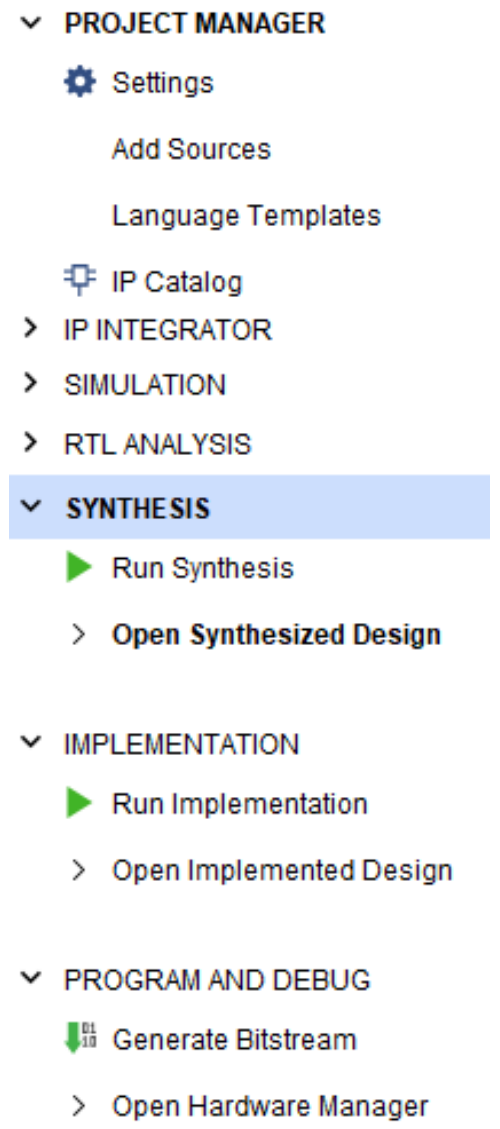
ZedBoard

Chip

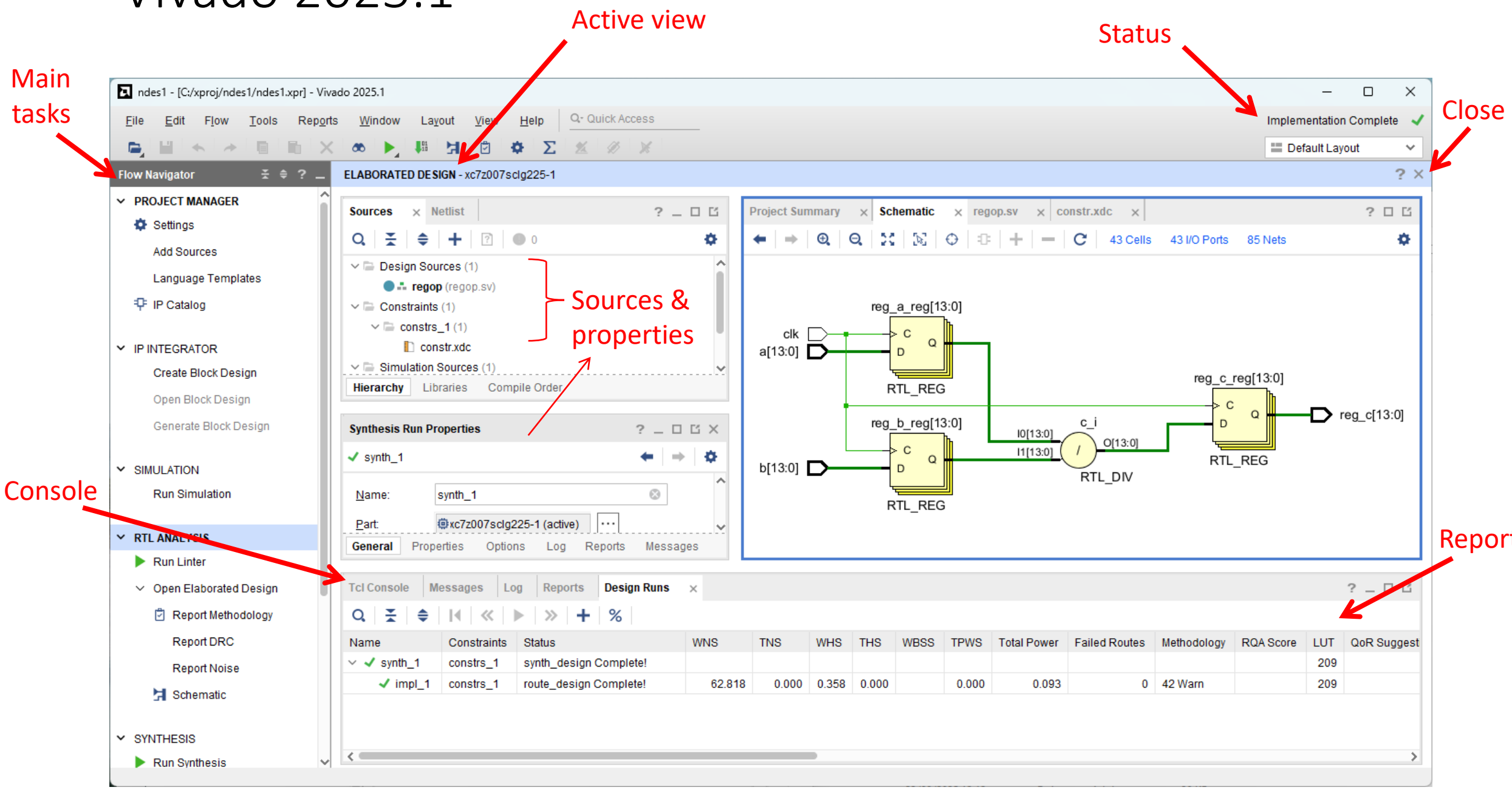


Programmable logic for
signal processing
CPU for control tasks

Design flow



Vivado 2025.1



Model registered operations (add, multiply, divide)

1. Create Vivado project
   - device: MiniZed (xc7z007sclg225-1)
   - constraints: 100 MHz or 10 MHz clock

```
create_clock -name clk -period 10.000 [get_ports clk];
```

2. Design source: regop.sv (SystemVerilog)
   - 14 or 28-bit operation with registers
3. Run Synthesis
   - check utilization and delay (Data Path Delay)

Tcl: `report_timing -delay_type max -max_paths 1`

4. Run Implementation
   - report timing and get precise delay info (clk period - WNS)
   - record utilization and max. clock frequency

```
module regop(
 input logic          clk,
 input logic  [13:0] a,b,
 output logic [13:0] reg_c
);

 logic [13:0] reg_a, reg_b;
 logic [13:0] c;

 always_ff @(posedge clk) begin
  reg_a <= a;
  reg_b <= b;
  reg_c <= c;
 end

 assign c = reg_a + reg_b;
endmodule
```

Report

circuit	LUT / FF / DSP	Data Path Delay [ns]	Logic levels	clk period [ns]	frequency [MHz]
add 14					
add 28					
mul 14					
mul 28					
div 14					
div 28					

Design averaging filter

- compute average of last three samples
- use internal buffers
- check with provided testbench
- explore options for division (1/3)
  - division by power of 2
  - integer divider
  - approximate with multiplication, 11/32

```
module avg_filter(
 input logic clk,
 input logic signed  [13:0] a,
 output logic signed [13:0] avg
);
 logic signed [13:0] z [0:2];
 logic signed [15:0] s;

 always_ff @(posedge clk) begin
  z[0] <= a;
  z[1] <= z[0];
  z[2] <= z[1];
  avg <= ...
 end

 assign s = z[0] + z[1] ...
```

circuit	LUT / FF / DSP	Data Path Delay [ns]	Logic levels	clk period [ns]	frequency [MHz]