



XILINX

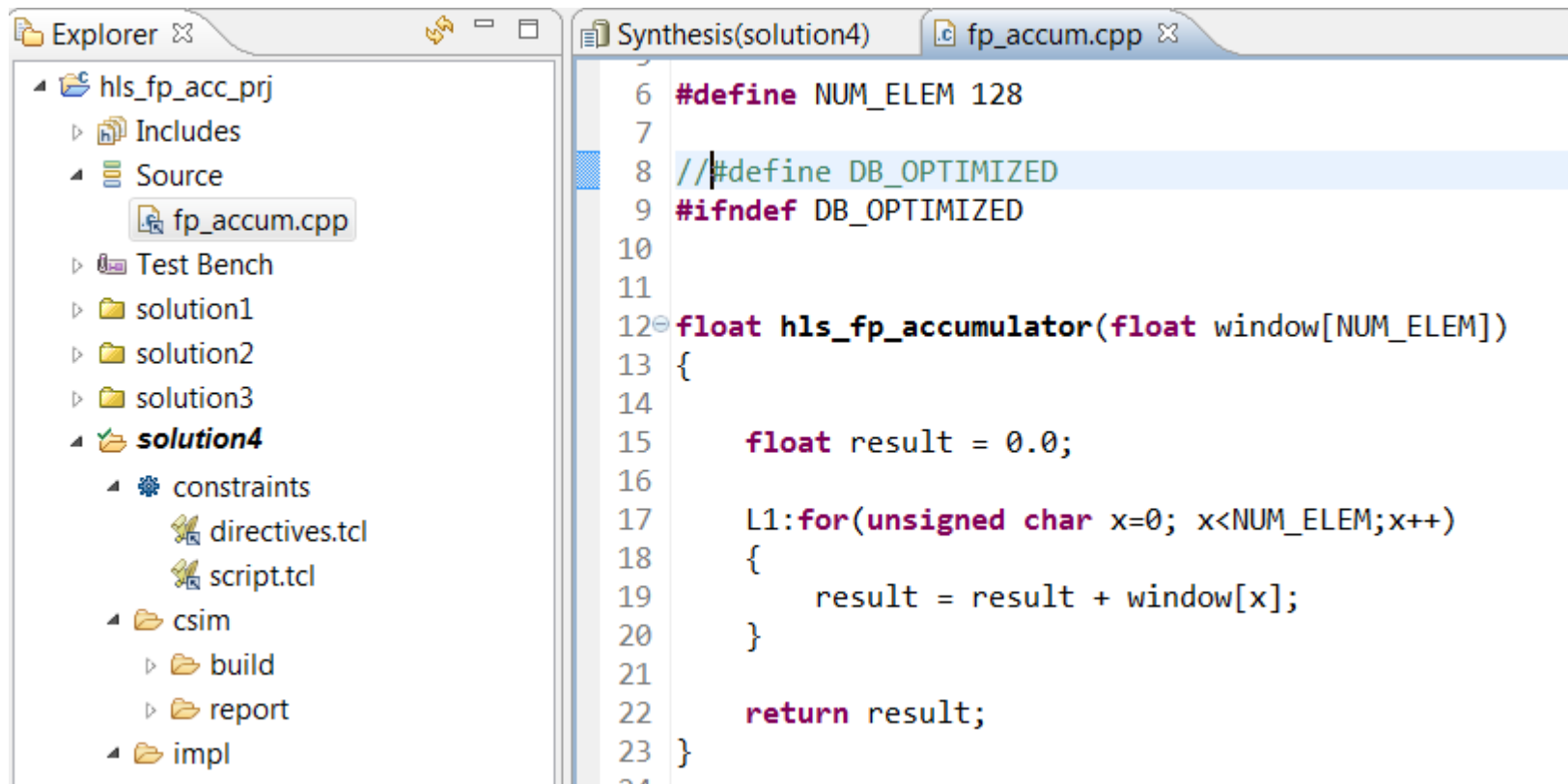
ALL PROGRAMMABLE™

Floating-Point accumulator

**Daniele Bagni,
DSP Specialist for EMEA
July, 2015**

FP Accumulator: the original code 1/2

- The original code is the worst from HLS point of view: purely sequential and with a **feedback**
 - UNROLL will never work here



The screenshot shows an IDE with two panes. The left pane is the 'Explorer' view showing a project structure for 'hls_fp_acc_prj'. The right pane is the 'Synthesis(solution4)' view showing the source file 'fp_accum.cpp'. The code in 'fp_accum.cpp' defines a function 'hls_fp_accumulator' that takes a float array 'window' of size 'NUM_ELEM' and returns a float 'result'. The function is implemented sequentially, iterating over the array elements and accumulating their values. The code is as follows:

```
6 #define NUM_ELEM 128
7
8 // #define DB_OPTIMIZED
9 #ifndef DB_OPTIMIZED
10
11
12 float hls_fp_accumulator(float window[NUM_ELEM])
13 {
14
15     float result = 0.0;
16
17     L1: for(unsigned char x=0; x<NUM_ELEM; x++)
18     {
19         result = result + window[x];
20     }
21
22     return result;
23 }
```

FP Accumulator: the original code 2/2

➤ The latency of the core prevents any parallelism

The screenshot shows the Vivado HLS interface for the project `hls_fp_acc_prj`. The **Module Hierarchy** window displays the following table:

	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
hls_fp_accumulator	0	2	492	308	1539	1540	none

The **Performance Profile** window shows the following table:

	Pipelined	Latency	Initiation Interval	Iteration Latency	Trip count
hls_fp_accumulator	-	1539	1540	-	-
L1	yes	1537	12	14	128

The **Current Module : hls fp accumulator** window displays the following table:

Operation\Control S...	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
1 L1														
2 result(phi mux)														
3 x(phi mux)														
4 exitcond(icmp)														
5 x 1(+)														
6 window load(read)														
7 result 1(fadd)														

Synthesis Estimations comparison 1/2

Explorer

- hls_fp_acc_prj
 - Includes
 - Source
 - Test Bench
 - solution1
 - solution2
 - solution3
 - solution4**
 - constraints
 - csim
 - impl
 - sim
 - syn

compare reports

Vivado HLS Report Comparison

All Compared Solutions

[solution1](#): xc7k325tffg900-2

[solution2](#): xc7k325tffg900-2

[solution3](#): xc7k325tffg900-2

[solution4](#): xc7k325tffg900-2

Performance Estimates

Timing (ns)

Clock		solution1	solution2	solution3	solution4
ap_clk	Target	2.50	2.50	2.50	2.50
	Estimated	2.10	2.10	2.74	3.38

Latency (clock cycles)

		solution1	solution2	solution3	solution4
Latency	min	1793	1539	356	245
	max	1793	1539	356	245
Interval	min	1794	1540	357	64
	max	1794	1540	357	64

Utilization Estimates

	solution1	solution2	solution3	solution4
BRAM_18K	0	0	4	0
DSP48E	2	2	2	4
FF	489	492	1511	4632
LUT	304	308	1451	2369

Synthesis Estimates Comparison: 2/2

➤ (original code) Solution1: baseline

➤ (original code) Solution2:

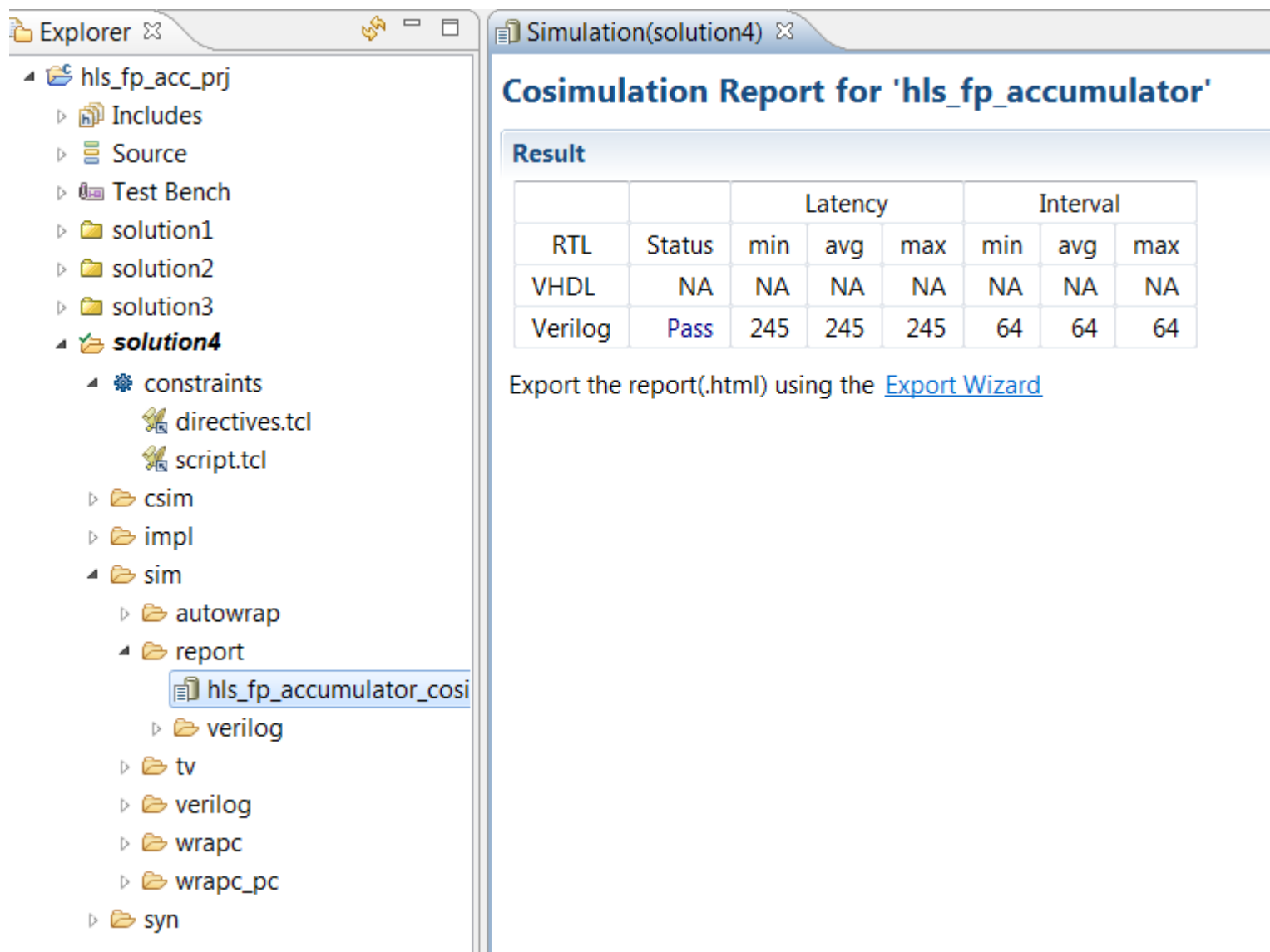
- `set_directive_pipeline "hls_fp_accumulator/L1"`

➤ (optimized code) Solution3: baseline

➤ (optimized code) Solution4: all inner loop are pipelined plus

- `set_directive_pipeline "hls_fp_accumulator"`

CoSimulation report



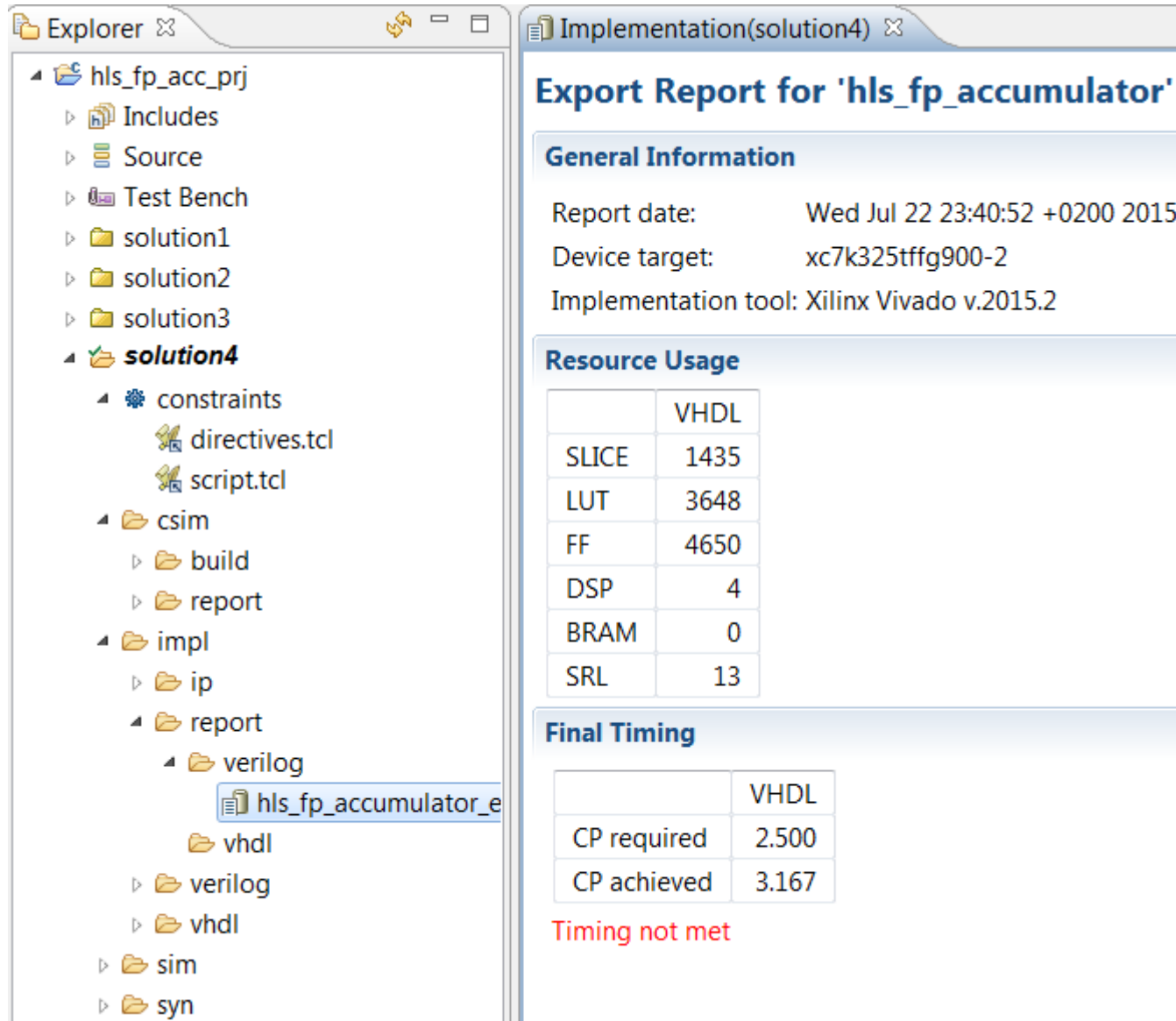
The screenshot displays the Xilinx IDE interface. On the left, the 'Explorer' pane shows a project tree for 'hls_fp_acc_prj'. The tree includes folders for 'Includes', 'Source', 'Test Bench', 'solution1', 'solution2', 'solution3', and 'solution4'. Under 'solution4', there are sub-folders for 'constraints', 'csim', 'impl', 'sim', 'tv', 'verilog', 'wrapc', 'wrapc_pc', and 'syn'. The 'sim' folder is expanded, showing 'autowrap', 'report', and 'hls_fp_accumulator_cosi'. The 'hls_fp_accumulator_cosi' file is selected.

On the right, the 'Simulation(solution4)' pane displays the 'Cosimulation Report for 'hls_fp_accumulator''. The report is titled 'Result' and contains a table with the following data:

RTL	Status	Latency			Interval		
		min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	245	245	245	64	64	64

Below the table, a text prompt states: 'Export the report(.html) using the [Export Wizard](#)'.

Implementation report



The screenshot displays the Vivado IDE interface. On the left, the 'Explorer' window shows a project hierarchy for 'hls_fp_acc_prj'. The 'solution4' folder is expanded, revealing subfolders like 'constraints', 'csim', 'impl', and 'report'. The 'report' folder is further expanded, showing 'verilog' and 'vhdl' subfolders. The 'hls_fp_accumulator_e' file is selected under the 'verilog' folder.

On the right, the 'Implementation(solution4)' window displays the 'Export Report for \'hls_fp_accumulator\''. The report is divided into three sections: 'General Information', 'Resource Usage', and 'Final Timing'.

General Information

- Report date: Wed Jul 22 23:40:52 +0200 2015
- Device target: xc7k325tffg900-2
- Implementation tool: Xilinx Vivado v.2015.2

Resource Usage

	VHDL
SLICE	1435
LUT	3648
FF	4650
DSP	4
BRAM	0
SRL	13

Final Timing

	VHDL
CP required	2.500
CP achieved	3.167

Timing not met

Agenda

- 1) FIR filter case: 1 channel, in fractional fixed-point precision
 - ARRAY_PARTITION, ALLOCATION, PIPELINE, UNROLL
 - LATENCY, LOOP TRIP_COUNT
- 2) FIR filter case: 1 channel, in Floating-Point
 - RESOURCE
- **3) Floating-Point Accumulator**
- 4) Dependency
- 5) image Histogram computation and equalization
- 6) Siemens' application "Gamma LUT" case study
- 7) Image Processing: from HLS to IPI, from IPI to ZC702 board
- 8) cordic arctan2
- 9) cordic sqrt