A SystemC profiling framework to improve fixed-point hardware utilization

Alisson Linhares,

Henrique Rusa, Daniel Formiga and Rodolfo Azevedo

alisson.carvalho@students.ic.unicamp.br

Institute of Computing

University of Campinas (Unicamp)

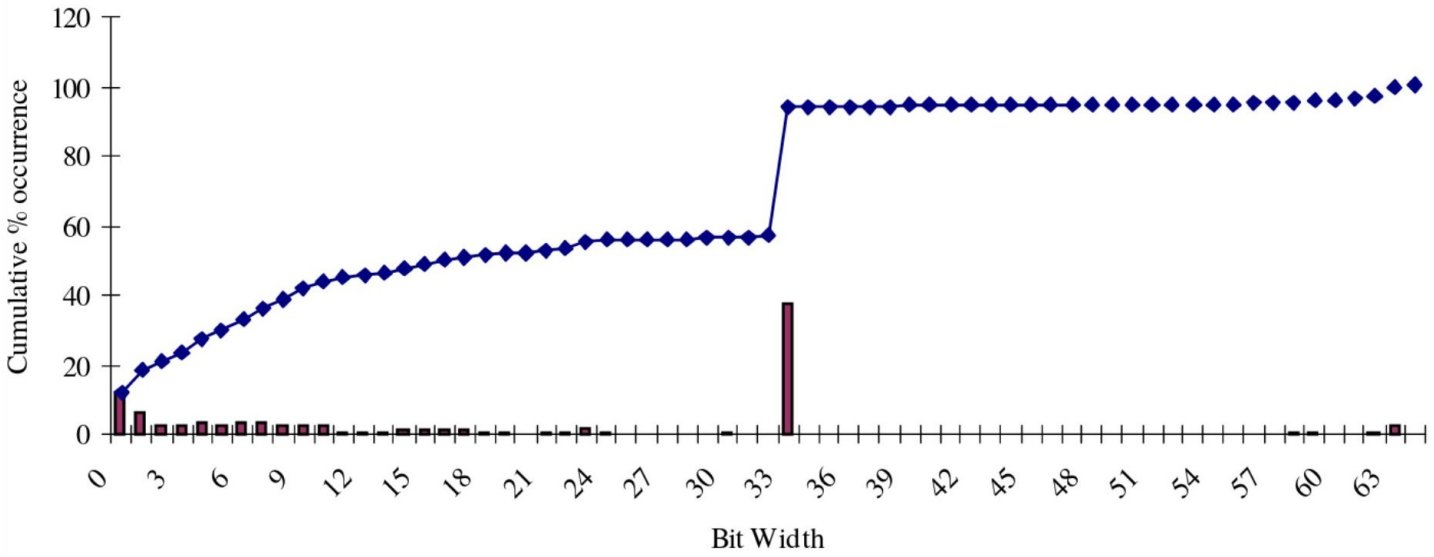
Campinas, SP, Brazil

August, 2020

Motivation

Hardware Subutilization

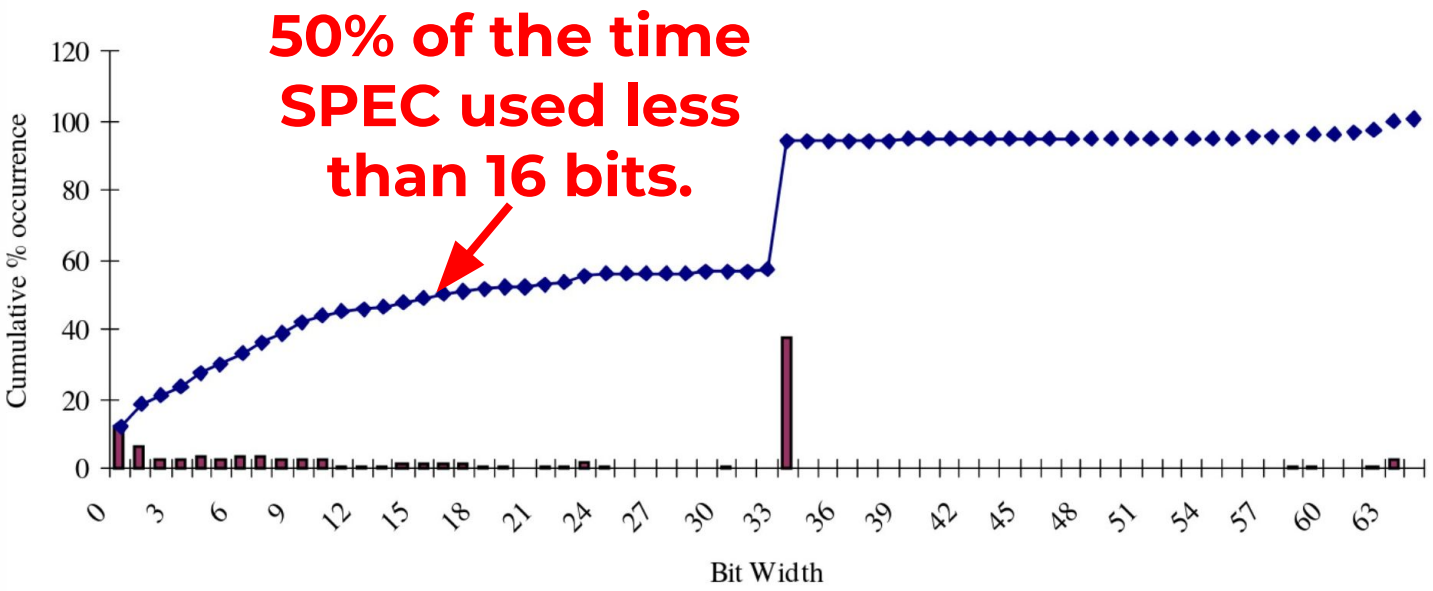
Several publications have shown indications that arithmetic units are usually underutilized.



(David Brooks and Margaret Martonosi)
Dynamically exploiting narrow width operands to improve processor power and performance.

Hardware Subutilization

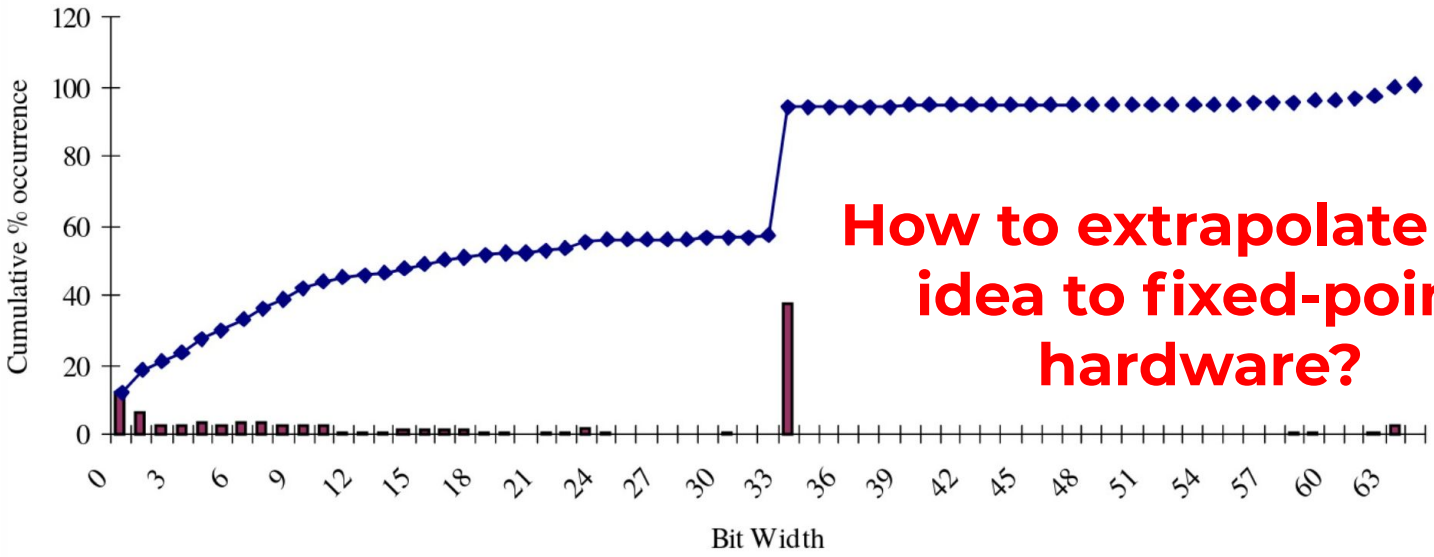
Several publications have shown indications that arithmetic units are usually underutilized.



(David Brooks and Margaret Martonosi)
Dynamically exploiting narrow width operands to improve processor power and performance.

Hardware Subutilization

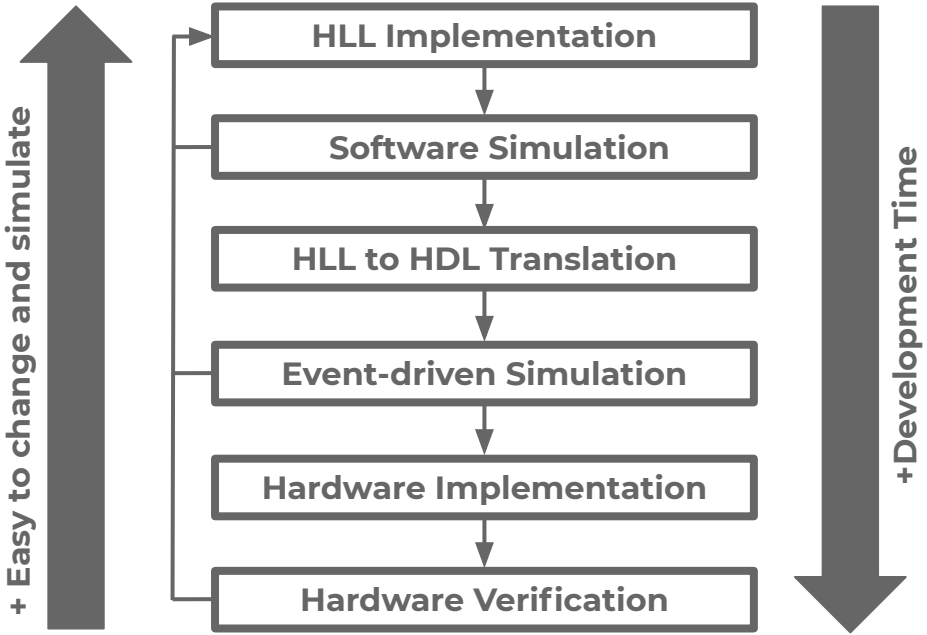
Several publications have shown indications that arithmetic units are usually underutilized.



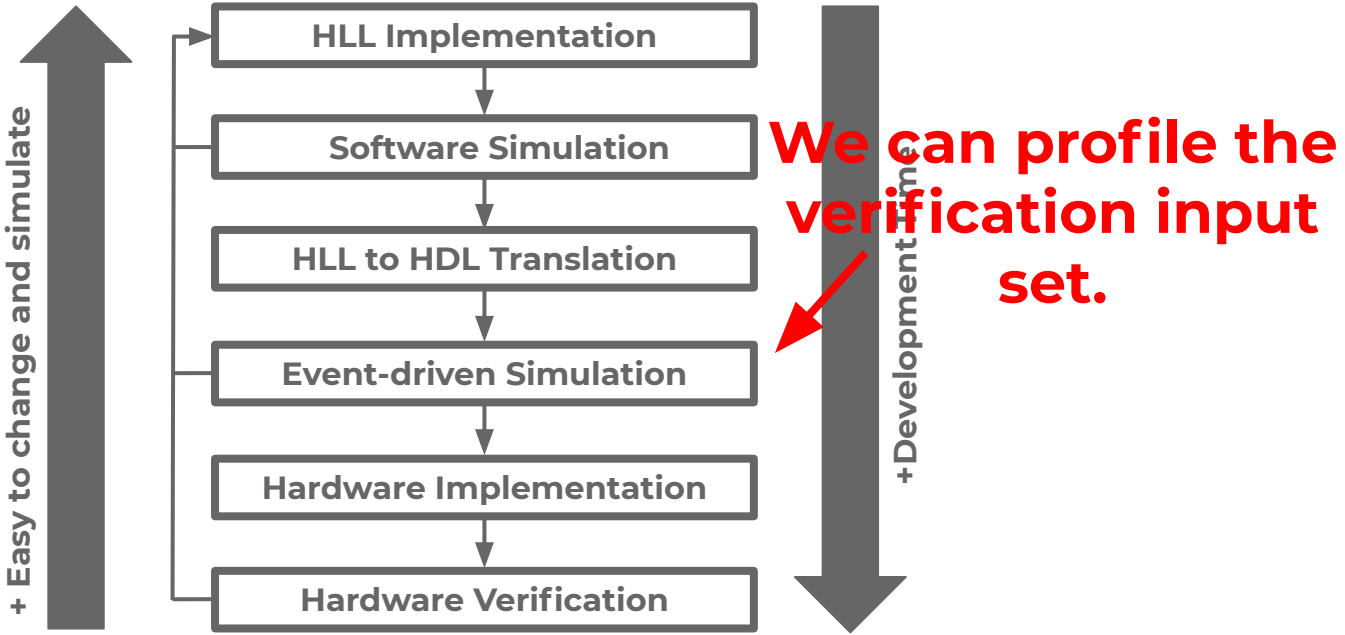
How to extrapolate this idea to fixed-point hardware?

(David Brooks and Margaret Martonosi)
Dynamically exploiting narrow width operands to improve processor power and performance.

Typical DSP design workflow

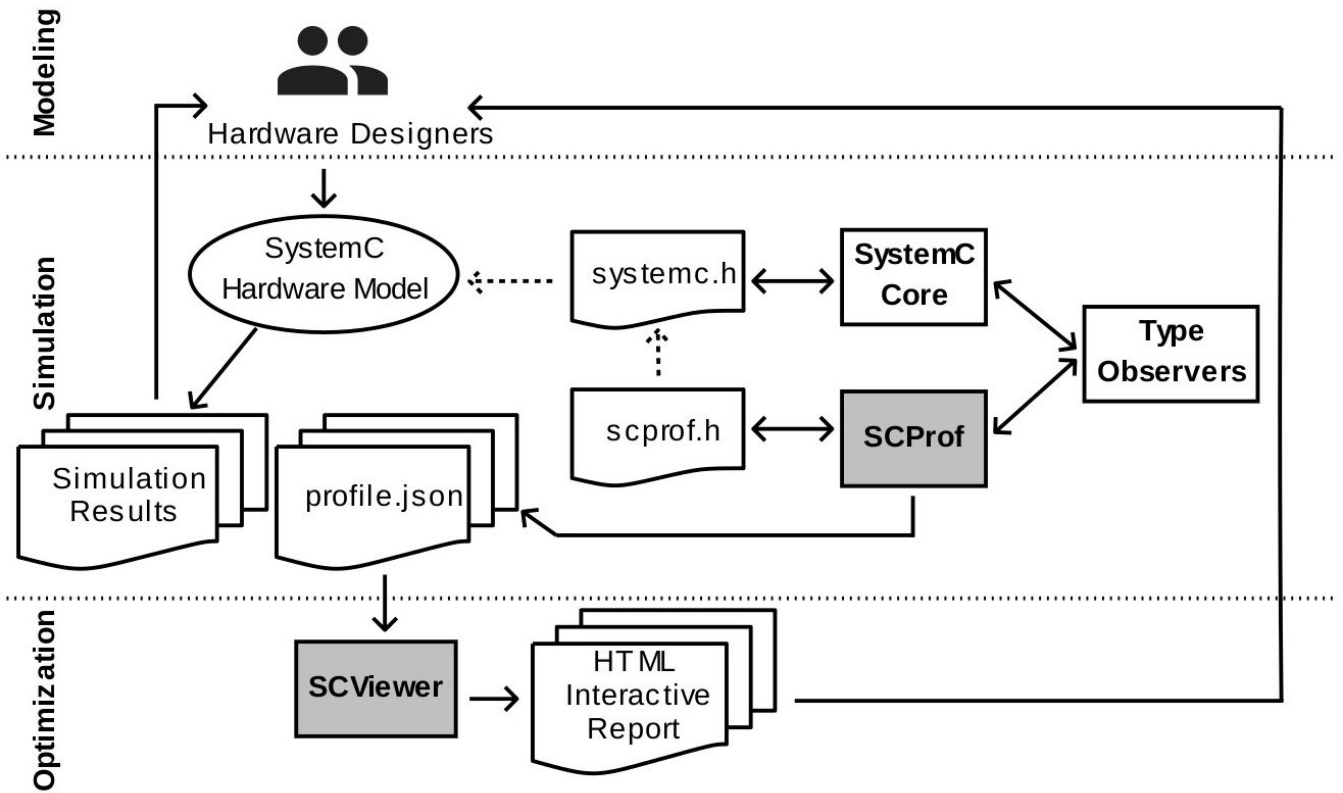


Typical DSP design workflow

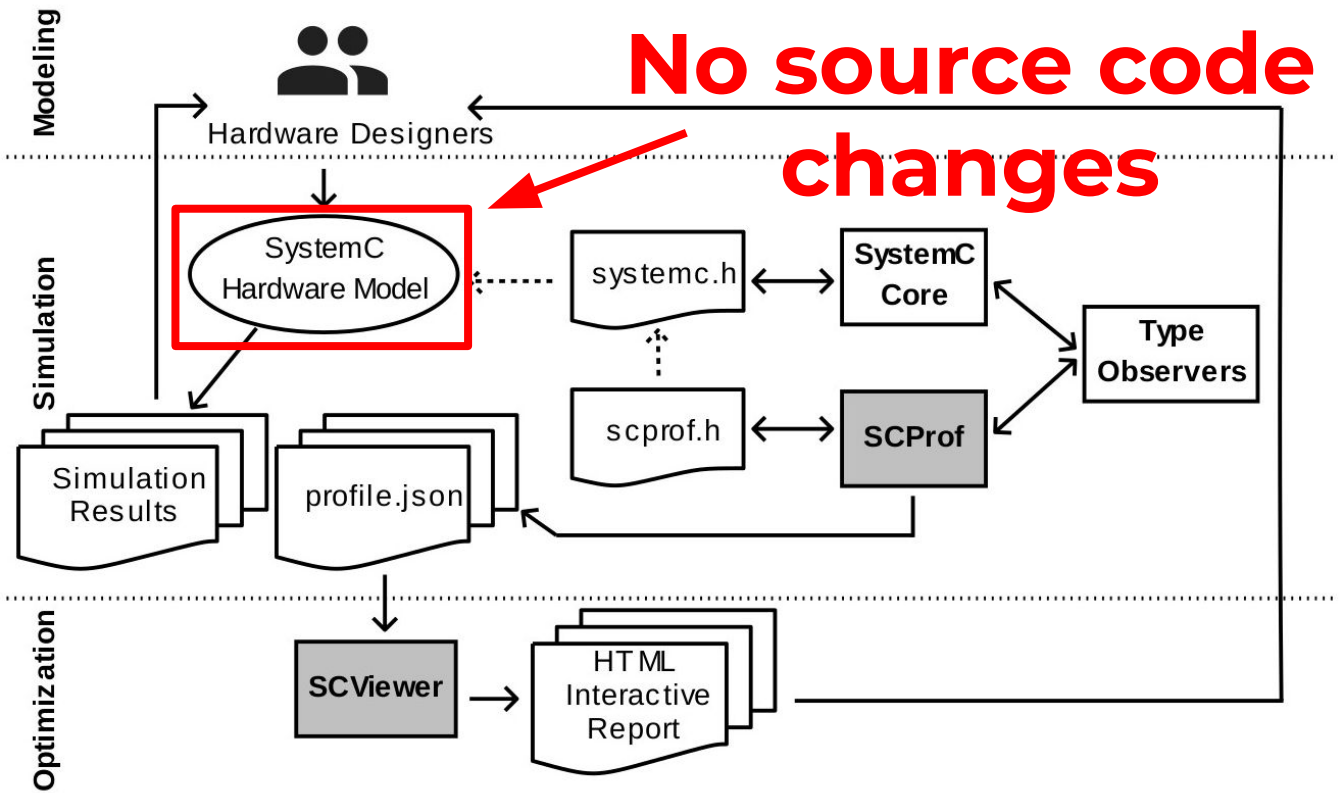


Profiling Infrastructure

Profiling Infrastructure

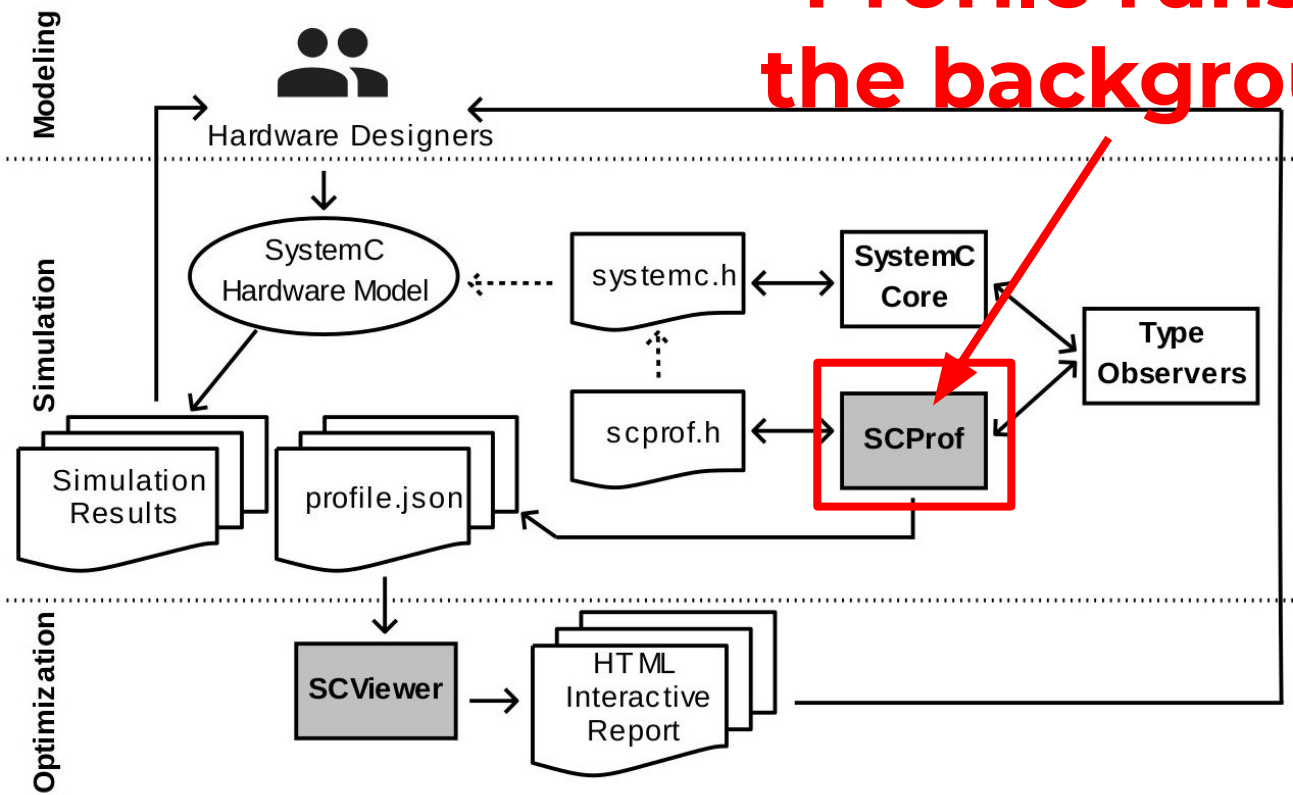


Profiling Infrastructure

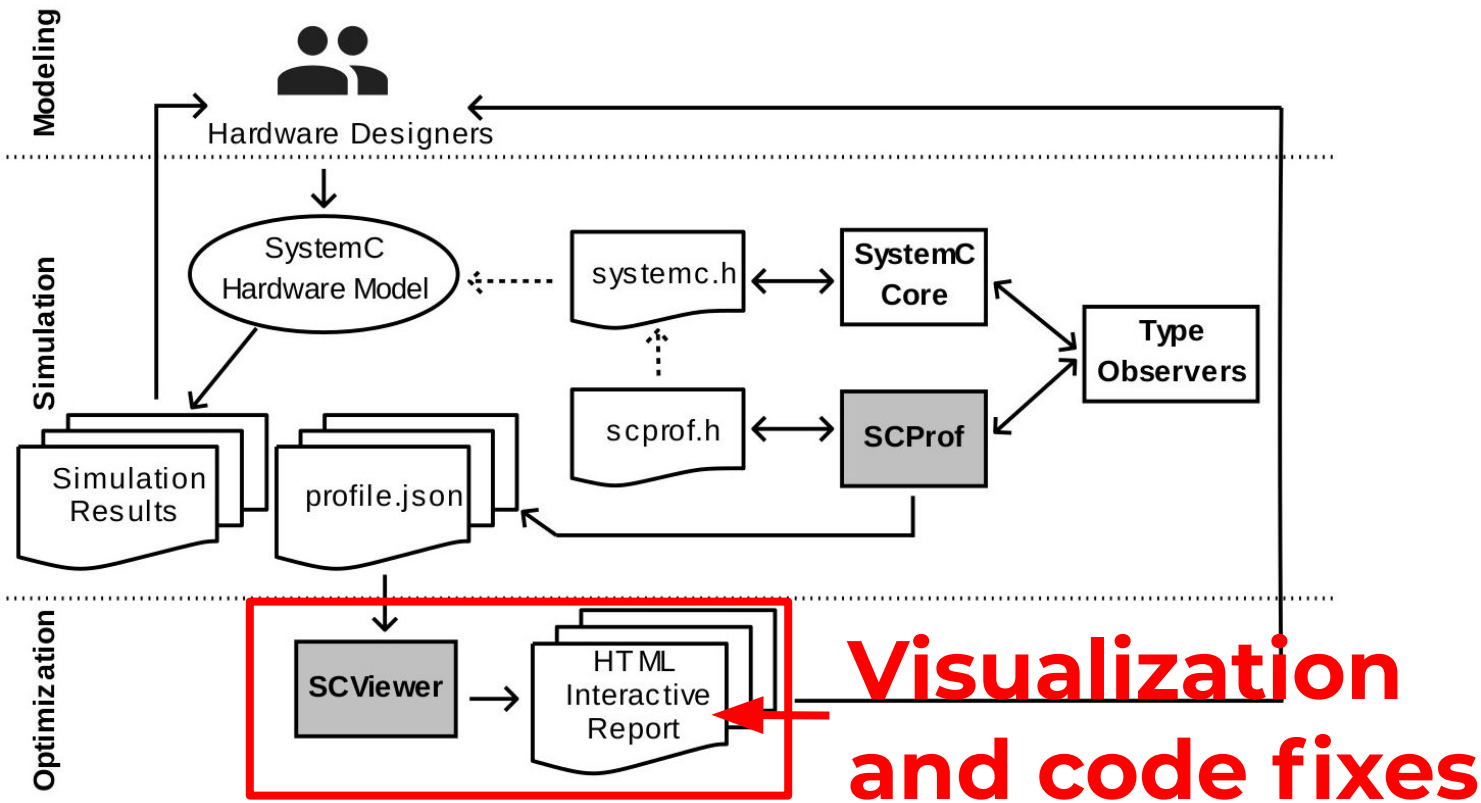


Profiling Infrastructure

Profile runs in the background



Profiling Infrastructure

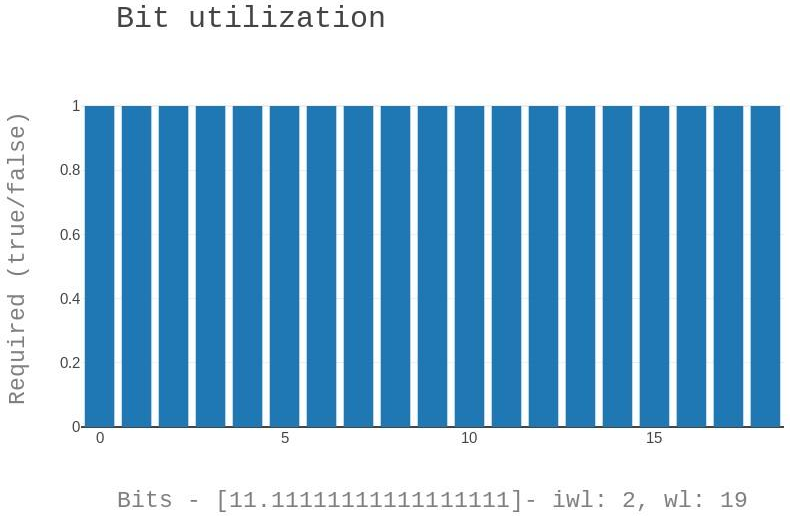


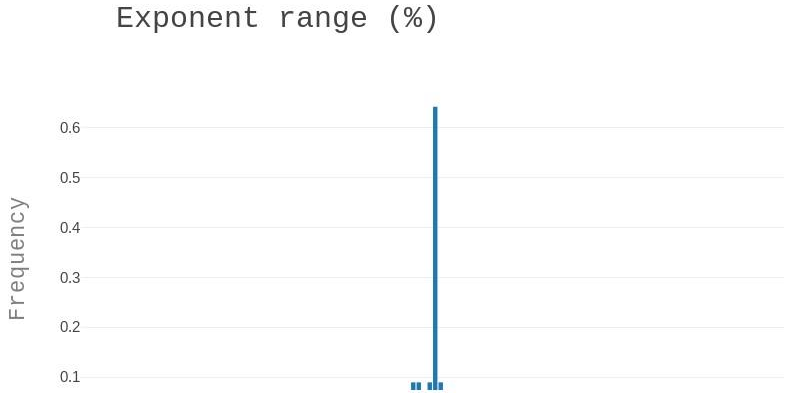
SCView

filt_decim.cpp

```
25    sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
26    bool stage_control_1 = false;
27
28    // signals for STAGE 2
29    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
30    sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
31    sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
32    sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
33    bool stage_control_2 = false;
34
35    // signals for STAGE 3
36    sc_fixed
37    sc_fixed
38    sc_fixed
39    sc_fixed
40    bool sta
41
42    // signa
43    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_4;
44    sc_fixed<19,2,SC_RND,SC_SAT> buffer_4;
45    sc_fixed<36,4,SC_RND,SC_SAT> SoP4;
46    sc_fixed<19, 2, SC_RND, SC_SAT> stage_4;
47    bool stage_control_4 = false;
48
49    // signals for STAGE 5
50    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_5;
51    sc_fixed<36,4,SC_RND,SC_SAT> SoP5;
52    sc_fixed<19, 2, SC_RND, SC_SAT> stage_5;
53    bool stage_control_5 = false;
54
55  stage_counter_1 = 0;
56  stage_counter_2 = 0;
57  stage_counter_3 = 0;
58  stage_counter_4 = 0;
59  stage_counter_5 = 0;
60
61  unsigned int x=0;;
62
63  sc_fixed<17,2,SC_RND,SC_SAT> incoef1_var[TAPS_STAGE1] ;
64  sc_fixed<17,2,SC_RND,SC_SAT> incoef2_var[TAPS_STAGE2] ;
65  sc_fixed<17,2,SC_RND,SC_SAT> incoef3_var[TAPS_STAGE3];
66  sc_fixed<17,2,SC_RND,SC_SAT> incoef4_var[TAPS_STAGE4] ;
67  sc_fixed<17,2,SC_RND,SC_SAT> incoef5_var[TAPS_STAGE5];
68
69  wait();
70
71  // Main stage filter body
72
73  while( true )
74  {
75
76    if(load_coeff.read()){
77      // Read coefficients into coef arrays
78      for (x = 0; x < TAPS_STAGE1; x++) {
79        incoef1_var[x] = incoef1[x].read();
80
81      }
82      for (x = 0; x < TAPS_STAGE2; x++) {
83
```

The analysis shows that some bits did not affect the overall accuracy of the application. To improve hardware utilization, you can reduce the word length (WL) to a value bigger than 7, in exchange for a reduction in precision.

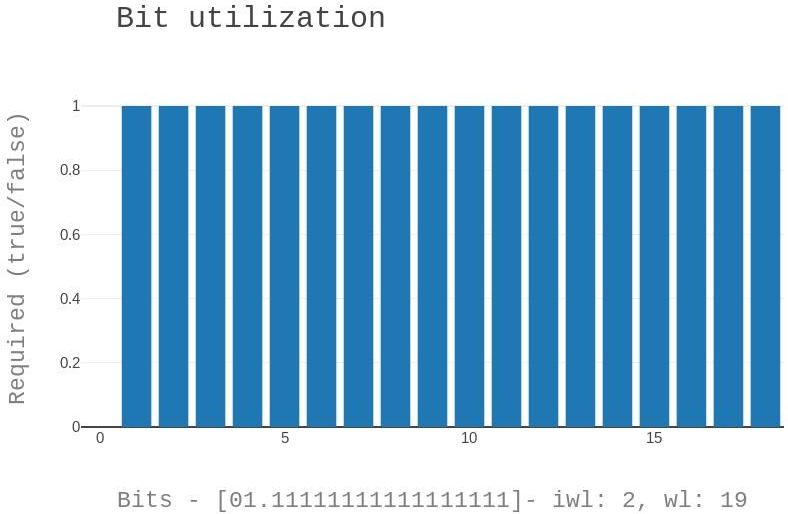


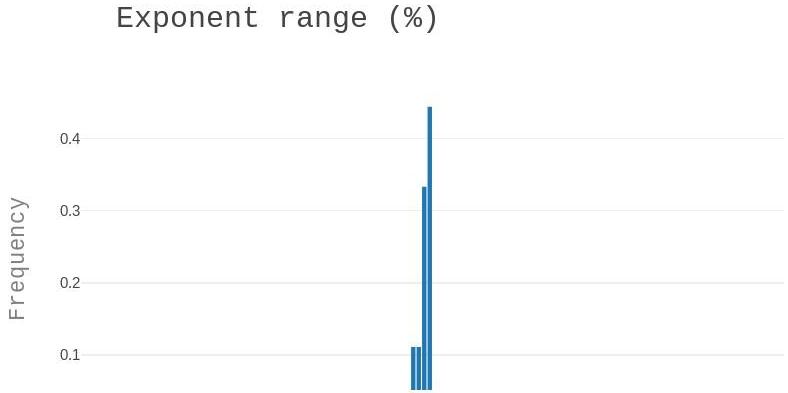


filt_decim.cpp

```
25    sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
26    bool stage_control_1 = false;
27
28    // signals for STAGE 2
29    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
30    sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
31    sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
32    sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
33    bool stage_control_2 = false;
34
35    // signals for STAGE 3
36    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_3;
37    sc_fixed<19,2,SC_RND,SC_SAT> buffer_3;
38    sc_fixed<36,4,SC_RND,SC_SAT> SoP3;
39    sc_fixed<19, 2, SC_RND, SC_SAT> stage_3;
40    bool sta
41
42    // signa
43    sc_fixed
44    sc_fixed
45    sc_fixed
46    sc_fixed<19, 2, SC_RND, SC_SAT> stage_4;
47    bool stage_control_4 = false;
48
49    // signals for STAGE 5
50    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_5;
51    sc_fixed<36,4,SC_RND,SC_SAT> SoP5;
52    sc_fixed<19, 2, SC_RND, SC_SAT> stage_5;
53    bool stage_control_5 = false;
54
55  stage_counter_1 = 0;
56  stage_counter_2 = 0;
57  stage_counter_3 = 0;
58  stage_counter_4 = 0;
59  stage_counter_5 = 0;
60
61  unsigned int x=0;;
62
63  sc_fixed<17,2,SC_RND,SC_SAT> incoef1_var[TAPS_STAGE1] ;
64  sc_fixed<17,2,SC_RND,SC_SAT> incoef2_var[TAPS_STAGE2] ;
65  sc_fixed<17,2,SC_RND,SC_SAT> incoef3_var[TAPS_STAGE3];
66  sc_fixed<17,2,SC_RND,SC_SAT> incoef4_var[TAPS_STAGE4] ;
67  sc_fixed<17,2,SC_RND,SC_SAT> incoef5_var[TAPS_STAGE5];
68
69  wait();
70
71  // Main stage filter body
72
73  while( true )
74  {
75
76    if(load_coeff.read()){
77      // Read coefficients into coef arrays
78      for (x = 0; x < TAPS_STAGE1; x++) {
79        incoef1_var[x] = incoef1[x].read();
80
81      }
82      for (x = 0; x < TAPS_STAGE2; x++) {
83
```

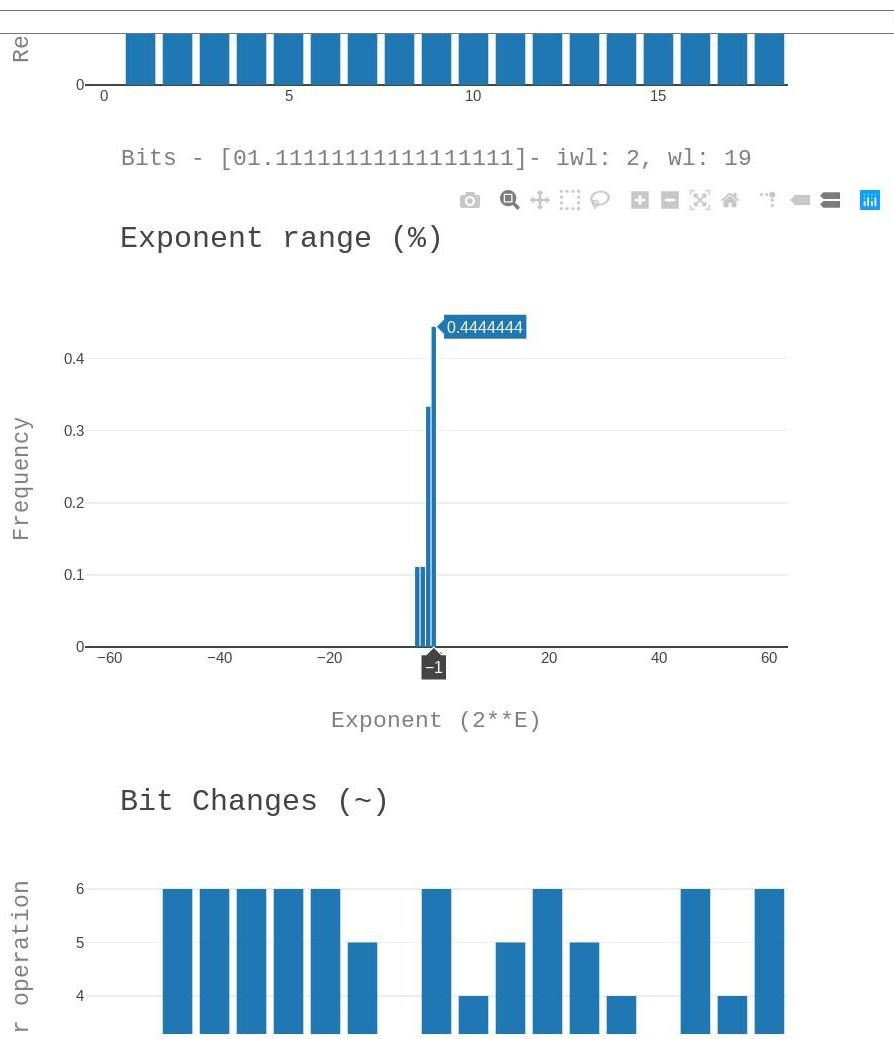
Some bits are never utilized. This may be a logic problem or the data used in your simulation is not representative. If this is not the case, change the precision to WL: 18 and IWL: 1 to improve hardware utilization.





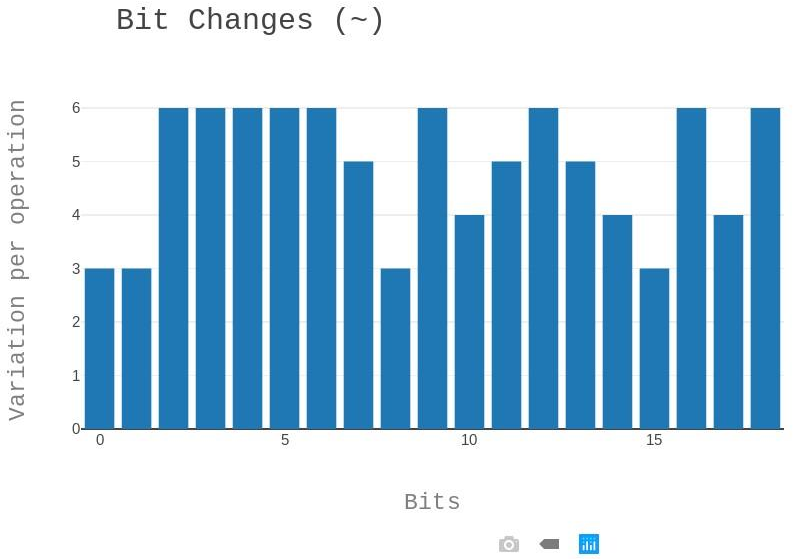
filt_decim.cpp

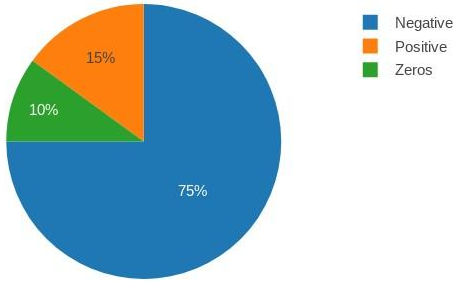
```
    sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
    bool stage_control_1 = false;

    // signals for STAGE 2
    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
    sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
    sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
    sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
    bool stage_control_2 = false;

    // signals for STAGE 3
    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_3;
    sc_fixed<19,2,SC_RND,SC_SAT> buffer_3;
    sc_fixed<36,4,SC_RND,SC_SAT> SoP3;
    sc_fixed<19, 2, SC_RND, SC_SAT> stage_3;
    bool stage_control_3 = false;

    // signals for STAGE 4
    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_4;
    sc_fixed<19,2,SC_RND,SC_SAT> buffer_4;
    sc_fixed<36,4,SC_RND,SC_SAT> SoP4;
    sc_fixed<19, 2, SC_RND, SC_SAT> stage_4;
    bool stage_control_4 = false;

    // signals for STAGE 5
    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_5;
    sc_fixed<36,4,SC_RND,SC_SAT> SoP5;
    sc_fixed<19, 2, SC_RND, SC_SAT> stage_5;
    bool stage_control_5 = false;

  stage_counter_1 = 0;
  stage_counter_2 = 0;
  stage_counter_3 = 0;
  stage_counter_4 = 0;
  stage_counter_5 = 0;

  unsigned int x=0;;

  sc_fixed<17,2,SC_RND,SC_SAT> incoef1_var[TAPS_STAGE1] ;
  sc_fixed<17,2,SC_RND,SC_SAT> incoef2_var[TAPS_STAGE2] ;
  sc_fixed<17,2,SC_RND,SC_SAT> incoef3_var[TAPS_STAGE3];
  sc_fixed<17,2,SC_RND,SC_SAT> incoef4_var[TAPS_STAGE4] ;
  sc_fixed<17,2,SC_RND,SC_SAT> incoef5_var[TAPS_STAGE5];

  wait();

  // Main stage filter body

  while( true )
  {

    if(load_coeff.read()){
      // Read coefficients into coef arrays
      for (x = 0; x < TAPS_STAGE1; x++) {
        incoef1_var[x] = incoef1[x].read();

      }
      for (x = 0; x < TAPS_STAGE2; x++) {
```



filt_decim.cpp

```
25    sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
26    bool stage_control_1 = false;
27
28    // signals for STAGE 2
29    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
30    sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
31    sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
32    sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
33    bool stage_control_2 = false;
34
35    // signals for STAGE 3
36    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_3;
37    sc_fixed<19,2,SC_RND,SC_SAT> buffer_3;
38    sc_fixed<36,4,SC_RND,SC_SAT> SoP3;
39    sc_fixed<19, 2, SC_RND, SC_SAT> stage_3;
40    bool stage_control_3 = false;
41
42    // signals for STAGE 4
43    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_4;
44    sc_fixed<19,2,SC_RND,SC_SAT> buffer_4;
45    sc_fixed<36,4,SC_RND,SC_SAT> SoP4;
46    sc_fixed<19, 2, SC_RND, SC_SAT> stage_4;
47    bool stage_control_4 = false;
48
49    // signals for STAGE 5
50    sc_fixed<19,2,SC_RND,SC_SAT> in_stage_5;
51    sc_fixed<36,4,SC_RND,SC_SAT> SoP5;
52    sc_fixed<19, 2, SC_RND, SC_SAT> stage_5;
53    bool stage_control_5 = false;
54
55  stage_counter_1 = 0;
56  stage_counter_2 = 0;
57  stage_counter_3 = 0;
58  stage_counter_4 = 0;
59  stage_counter_5 = 0;
60
61  unsigned int x=0;;
62
63  sc_fixed<17,2,SC_RND,SC_SAT> incoef1_var[TAPS_STAGE1] ;
64  sc_fixed<17,2,SC_RND,SC_SAT> incoef2_var[TAPS_STAGE2] ;
65  sc_fixed<17,2,SC_RND,SC_SAT> incoef3_var[TAPS_STAGE3];
66  sc_fixed<17,2,SC_RND,SC_SAT> incoef4_var[TAPS_STAGE4] ;
67  sc_fixed<17,2,SC_RND,SC_SAT> incoef5_var[TAPS_STAGE5];
68
69  wait();
70
71  // Main stage filter body
72
73  while( true )
74  {
75
76    if(load_coeff.read()){
77      // Read coefficients into coef arrays
78      for (x = 0; x < TAPS_STAGE1; x++) {
79        incoef1_var[x] = incoef1[x].read();
80
81      }
82      for (x = 0; x < TAPS_STAGE2; x++) {
83
```





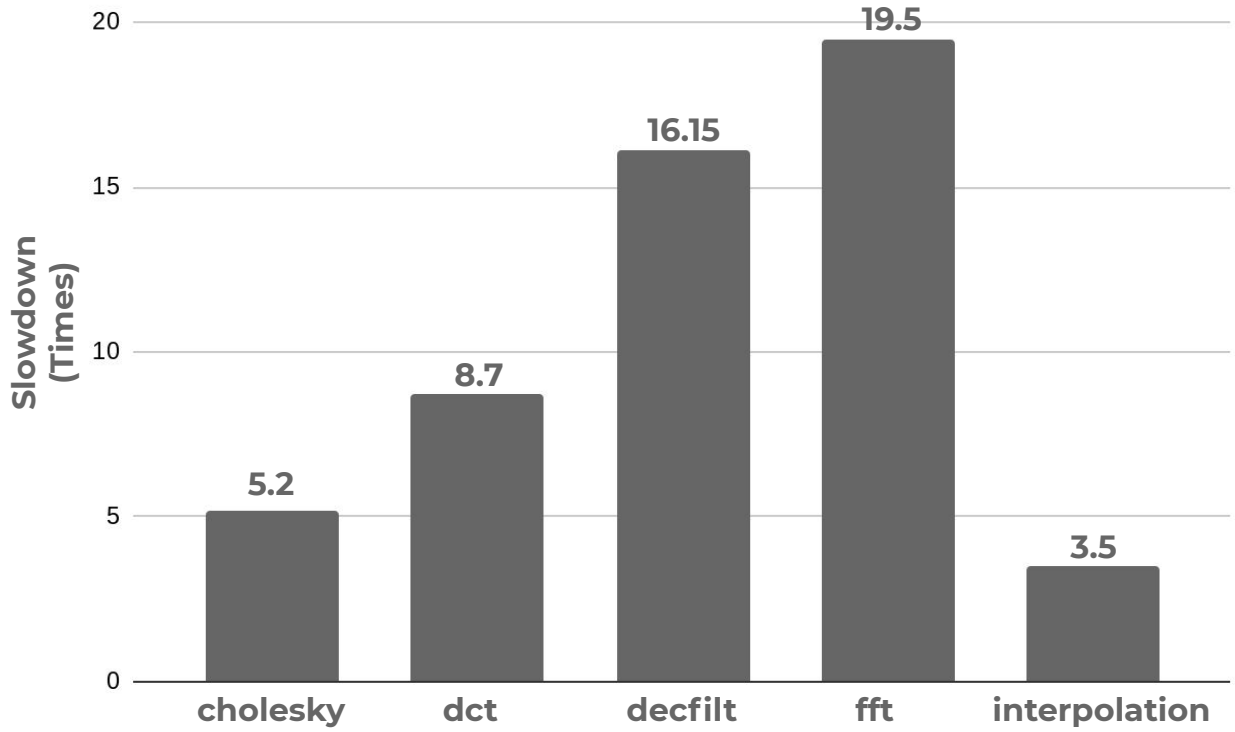
Results

Simulation Environment

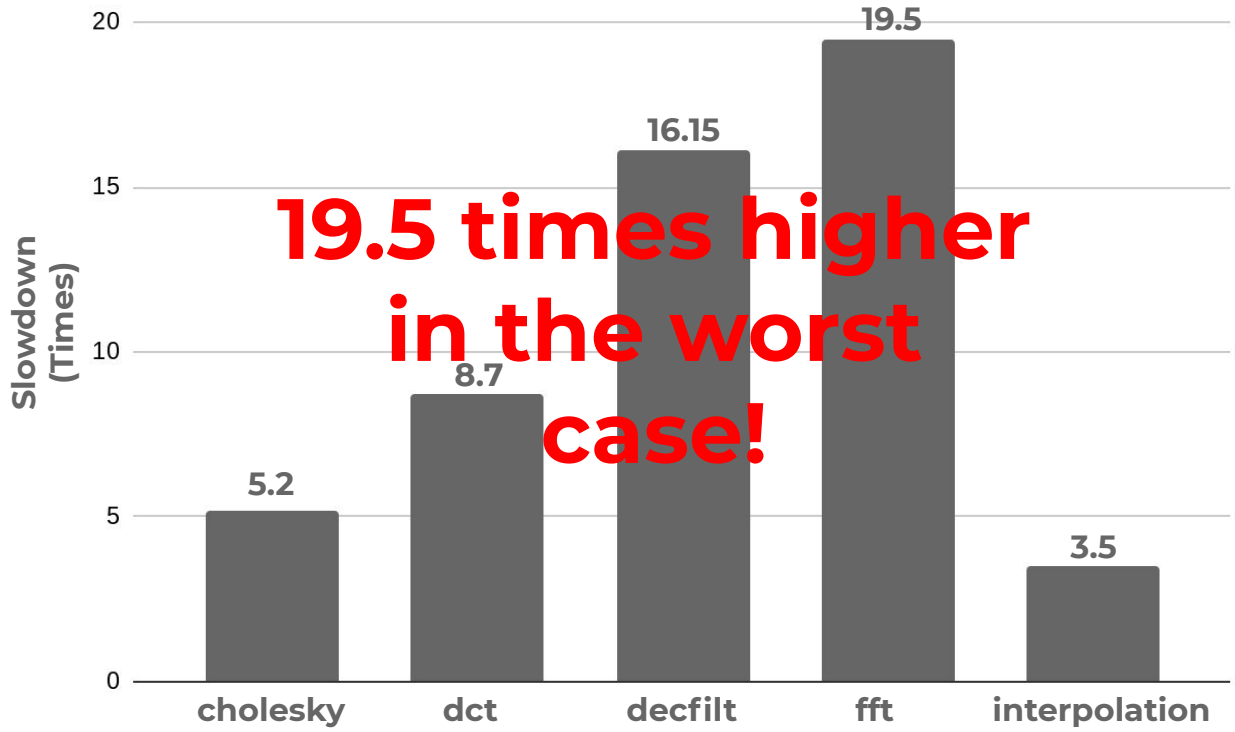
Benchmark	Description
cholesky	Cholesky Decomposition
dct	Fixed-point kernel from a JPEG encoder
decfilt	Decimation Filter System
fft	Fixed-point Fast Fourier Transform
interpolation	Interpolation Filter

- Machine:
  - Intel 6600k,
  - 2x8GB DDR4 2400MHz
  - Debian 4.9.144-3.1
- Target FPGA:
  - Kintex®-7 FPGA (xc7k70tfbv676-1)
    - Vivado Design Suite HLx 2019.2
- Results:
  - 10 consecutive runs (mean average)

Simulation overhead

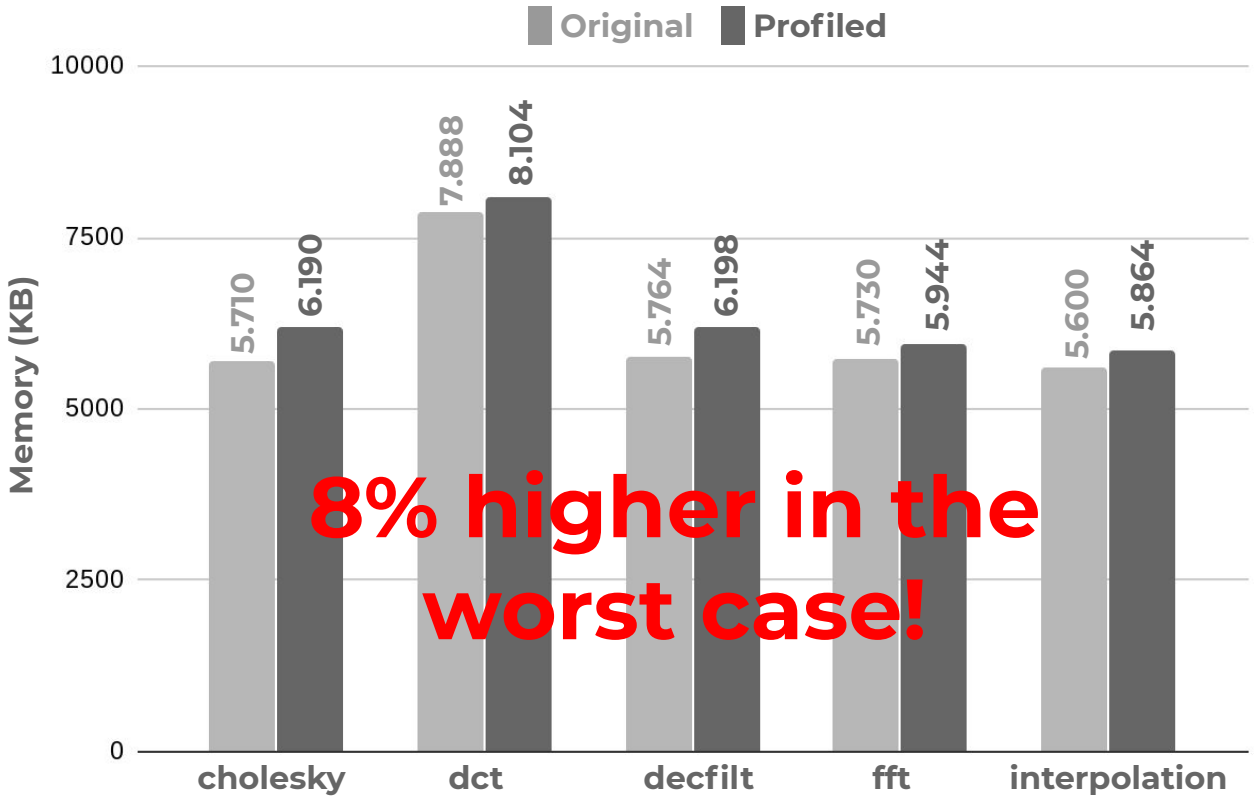


Simulation overhead



Memory usage

Memory usage



FPGA Statistics

Benchmark	DSP		FF		LUT		Latency (ns)		Clock (ns)	
	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.
cholesky	2	2	605	402	1126	1007	475	367	8.703	8.091
dct	2	1	306	299	1013	1023	18358	18358	8.992	8.992
decfilt	5	5	1848	1784	4028	3857	300	300	7.215	7.215
fft	32	16	2458	2030	4197	4324	X	X	8.379	8.490
interp	4	4	433	291	537	335	118	86	8.454	8.454

FPGA Statistics

Benchmark	DSP		FF		LUT		Latency (ns)		Clock (ns)	
	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.
cholesky	2	2	605	402	1126	1007	475	367	8.703	8.091
dct	2	1	306	299	1013	1023	18358	18358	8.992	8.992
decfilt	5	5	1848	1784	4028	3857	300	300	7.215	7.215
fft	32	16	2458	2030	4197	4324	X	X	8.379	8.490
interp	4	4	433	291	537	335	118	86	8.454	8.454

FPGA Statistics

Benchmark	DSP		FF		LUT		Latency (ns)		Clock (ns)	
	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.	Ori.	Opt.
cholesky	2	2	605	402	1126	1007	475	367	8.703	8.091
dct	2	1	306	299	1013	1023	18358	18358	8.992	8.992
decfilt	5	5	1848	1784	4028	3857	300	300	7.215	7.215
fft	32	16	2458	2030	4197	4324	X	X	8.379	8.490
interp	4	4	433	291	537	455	118	86	8.454	8.454

Less DSP -> More LUT

Real Life Application

Profiling a DSP hardware

- We tested the effectiveness of our methodology in a state of the art DSP Module.
  - The test was performed in a proprietary block.
    - IQC (Digital In-phase(I) and Quadrature phase(Q) imbalance correction).
    - Already optimized by professional hardware designers (During 1 year).

Profiling a DSP hardware

- We tested the effectiveness of our methodology in a state of the art DSP Module.
  - The test was performed in a proprietary block.
    - IQC (Digital In-phase(I) and Quadrature phase(Q) imbalance correction).
    - Already optimized by professional hardware designers (During 1 year).
- Methodology:
  - Profile IQC model:
    - Using 5 different real life scenarios.
      - Worst case, Upper Bound, Lower Bound, Normal execution and different variations.
    - And one random input stimulus with 124 million entries (Random Test).
  - Combined all results.
  - Apply optimization on System Verilog and SystemC.
  - Run Verification tests and Power Analysis.
  - Passing on the original test cases.

Results

Signal	Original		Random		Test 1.1		Test 1.2		Test 1.3		Test 1.4		Test 1.5		Optimized		Improvement
	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	
iqc_norm_t	9	1	9	1	9	1	9	1	9	1	9	1	9	1	9	1	0,00%
iqc_cang_rnd_sat_t	8	0	2	-6	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	50,00%
iqc_efilter_t	12	0	11	-1	11	-1	11	-1	11	-1	11	-1	11	-1	11	-1	8,33%
iqc_cang_calc_acc_t	13	0	7	-6	8	-5	8	-5	9	-4	8	-5	9	-4	9	-4	30,77%
iqc_q_cor_t	9	2	7	0	9	2	9	2	9	2	9	2	9	2	9	2	0,00%
iqc_energy2_t	11	-1	9	-3	9	-3	10	-2	9	-3	10	-2	10	-2	10	-2	9,09%
iqc_acc_a_s_t	16	-2	12	-6	13	-5	14	-4	13	-5	14	-4	14	-4	14	-4	12,50%
iqc_acc_i_u_t	11	-1	9	-3	9	-3	10	-2	9	-3	10	-2	10	-2	10	-2	9,09%
iqc_inv_t	14	5	13	4	14	5	13	4	14	5	13	4	13	4	14	5	0,00%
iqc_sqrt_t	12	0	10	-1	11	-1	11	-1	11	-1	11	-1	11	-1	1	-1	8,33%
iqc_inv_t_x	14	5	11	2	11	2	11	2	11	2	11	2	11	2	11	2	21,43%
iqc_cang_div_t	13	0	10	-3	11	-2	11	-2	11	-2	11	-2	11	-2	11	-2	15,38%
iqc_cang_calc_t	13	0	5	-8	6	-7	6	-7	6	-7	6	-7	6	-7	6	-7	53,85%
iqc_acc_a_u_t	15	-3	11	-7	12	-6	13	-5	12	-6	13	-5	13	-5	13	-5	13,33%
iqc_out_t	8	1	7	0	8	1	8	1	8	1	8	1	8	1	8	1	0,00%
iqc_angn_t	9	1	3	-5	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	55,56%

Results

Signal	Original		Random		Test 1.1		Test 1.2		Test 1.3		Test 1.4		Test 1.5		Optimized		Improvement
	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	WL	IWL	
iqc_norm_t	9	1	9	1	9	1	9	1	9	1	9	1	9	1	9	1	0,00%
iqc_cang_rnd_sat_t	8	0	2	-6	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	50,00%
iqc_efilter_t	12	0	11	-1	11	-1	11	-1	11	-1	11	-1	11	-1	11	-1	8,33%
iqc_cang_calc_acc_t	13	0	7	-6	8	-5	8	-5	9	-4	8	-5	9	-4	9	-4	30,77%
iqc_q_cor_t	9	2	7	0	9	2	9	2	9	2	9	2	9	2	9	2	0,00%
iqc_energy2_t	11	-1	9	-3	9	-3	10	-2	9	-3	10	-2	10	-2	10	-2	9,09%
iqc_acc_a_s_t	16	-2	12	-6	13	-5	14	-4	13	-5	14	-4	14	-4	14	-4	12,50%
iqc_acc_i_u_t	11	-1	9	-3	9	-3	10	-2	9	-3	10	-2	10	-2	10	-2	9,09%
iqc_inv_t	14	5	13	4	14	5	13	4	14	5	13	4	13	4	14	5	0,00%
iqc_sqrt_t	12	0	10	-1	11	-1	11	-1	11	-1	11	-1	11	-1	1	-1	8,33%
iqc_inv_t_x	14	5	11	2	11	2	11	2	11	2	11	2	11	2	11	2	21,43%
iqc_cang_div_t	13	0	10	-3	11	-2	11	-2	11	-2	11	-2	11	-2	11	-2	15,38%
iqc_cang_calc_t	13	0	5	-8	6	-7	6	-7	6	-7	6	-7	6	-7	6	-7	53,85%
iqc_acc_a_u_t	15	-3	11	-7	12	-6	13	-5	12	-6	13	-5	13	-5	13	-5	13,33%
iqc_out_t	8	1	7	0	8	1	8	1	8	1	8	1	8	1	8	1	0,00%
iqc_angn_t	9	1	3	-5	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	55,56%

2.46% Power Reduction (No impact in OSNR)

Thank you!

A SystemC profiling framework to improve fixed-point hardware utilization

Alisson Linhares,

Henrique Rusa, Daniel Formiga and Rodolfo Azevedo

alisson.carvalho@students.ic.unicamp.br

Institute of Computing

University of Campinas (Unicamp)

Campinas, SP, Brazil

August, 2020