

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

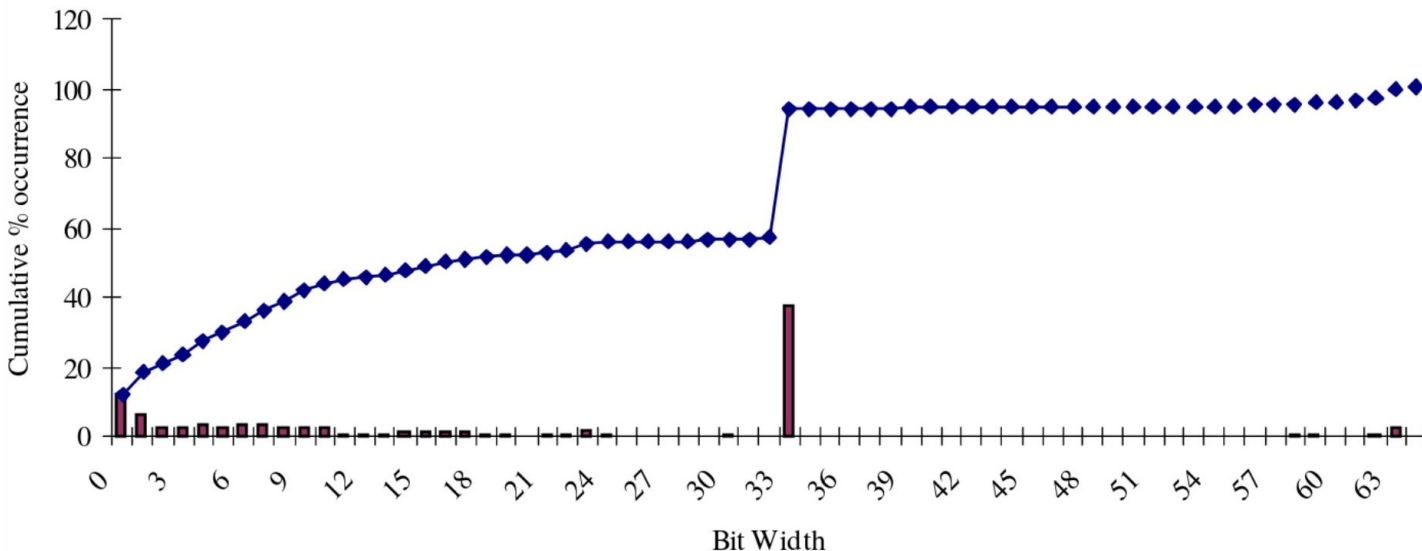
August, 2020



Motivation

Hardware Subutilization

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

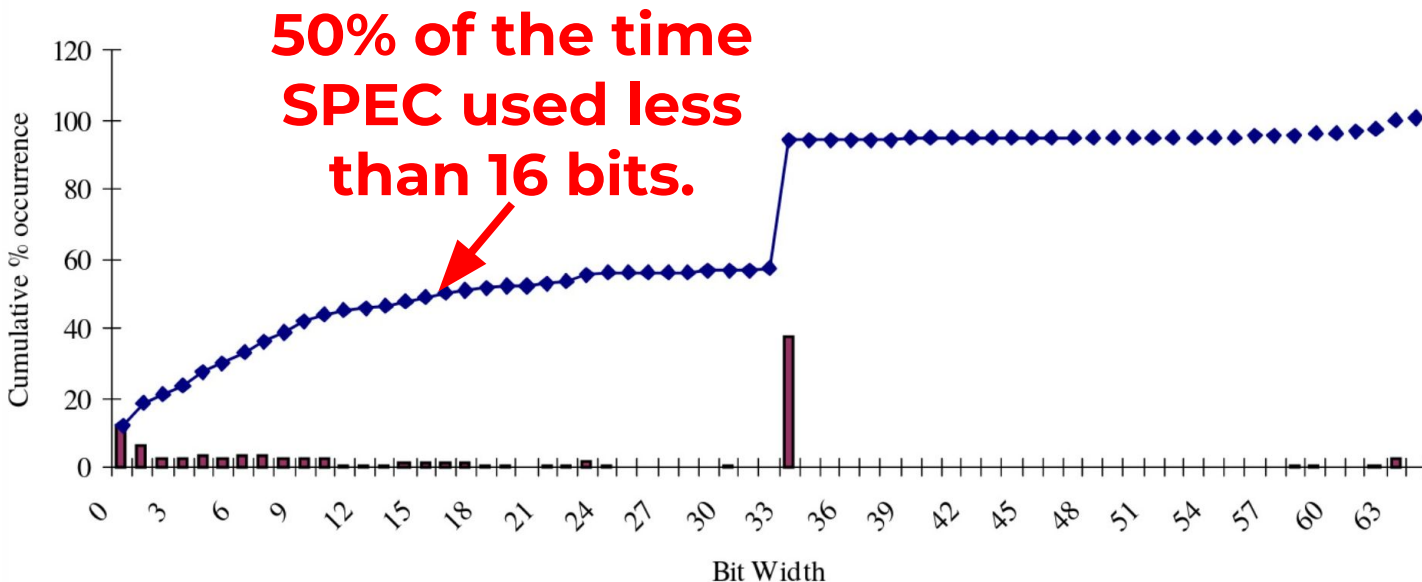


(David Brooks and Margaret Martonosi)

Dynamically exploiting narrow width operands to improve processor power and performance.

Hardware Subutilization

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

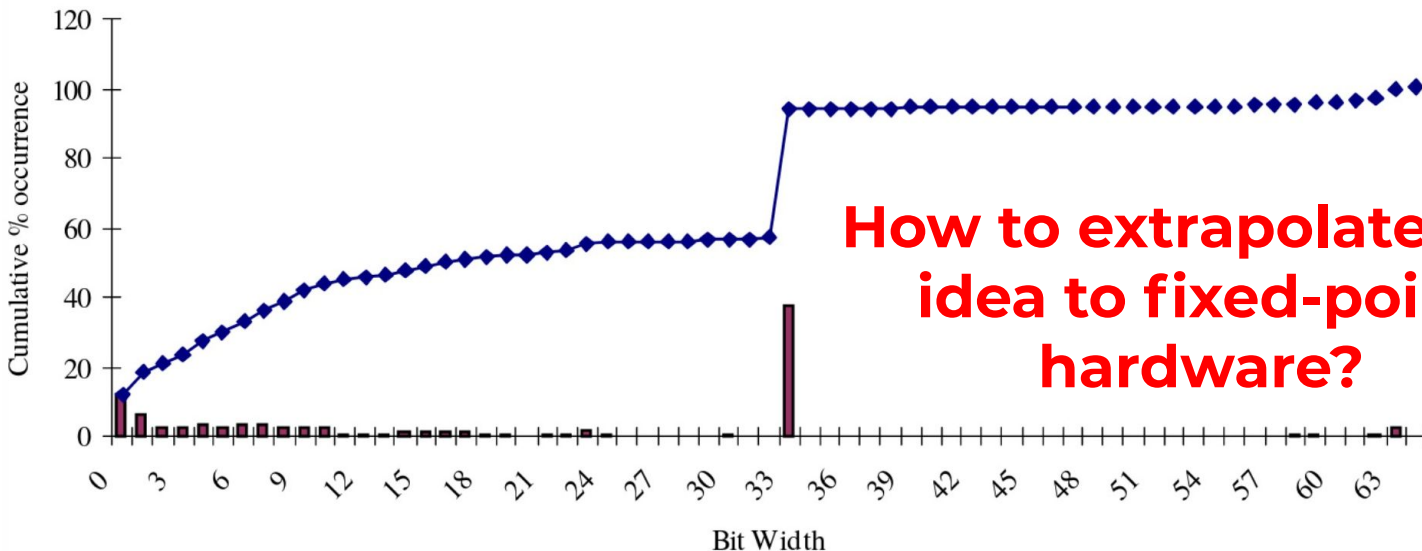


(David Brooks and Margaret Martonosi)

Dynamically exploiting narrow width operands to improve processor power and performance.

Hardware Subutilization

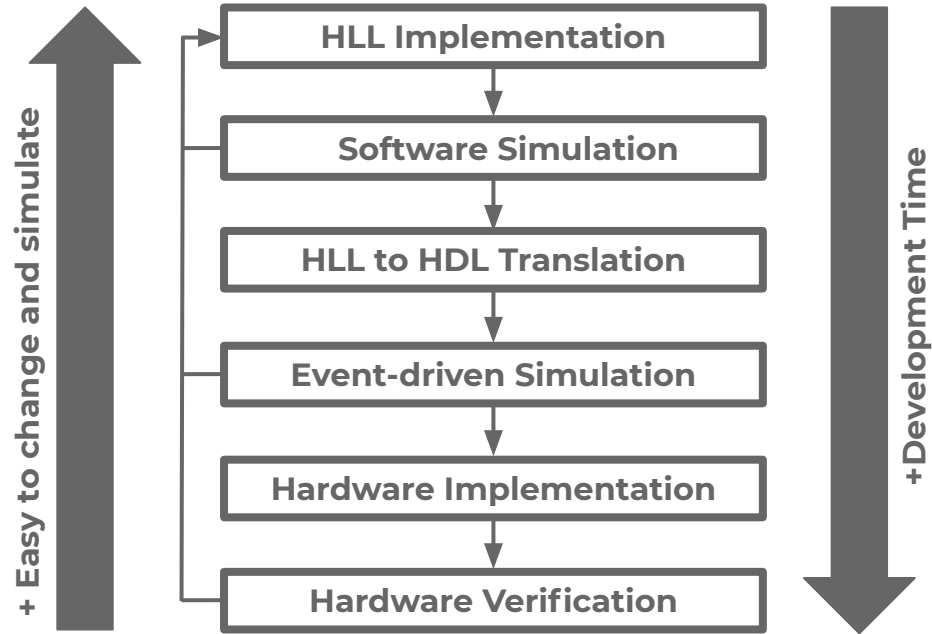
Several publications have shown indications that arithmetic units are usually underutilized.



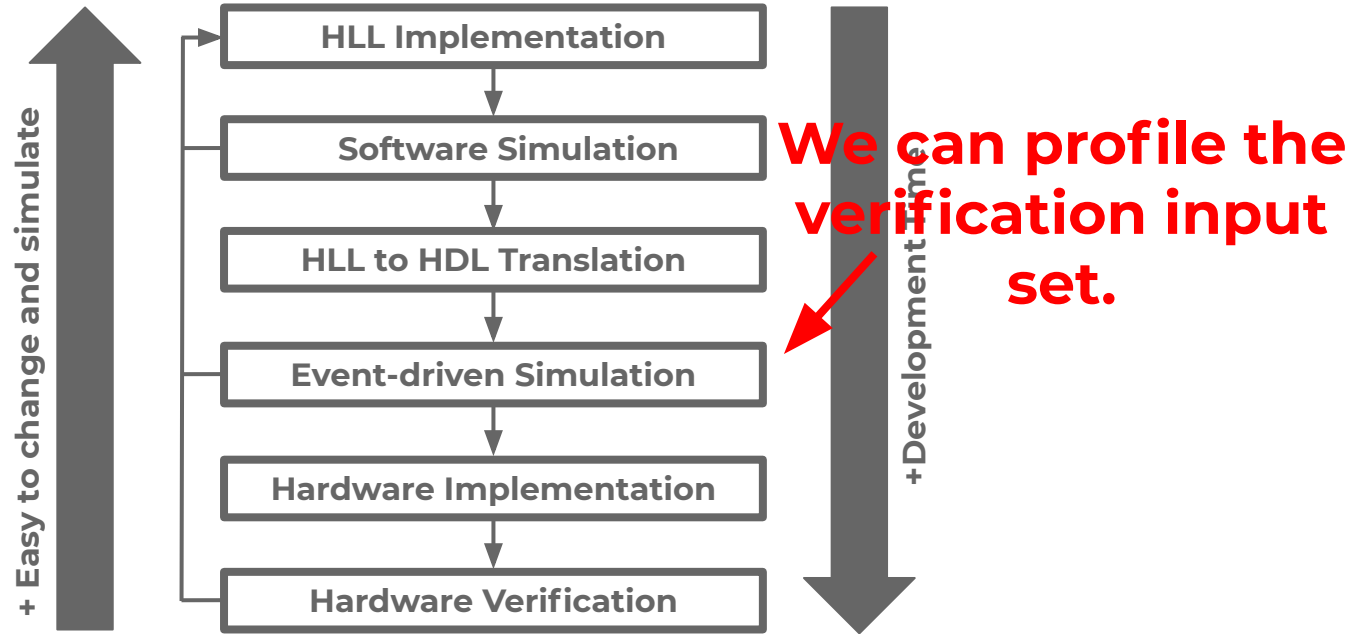
(David Brooks and Margaret Martonosi)

Dynamically exploiting narrow width operands to improve processor power and performance.

Typical DSP design workflow

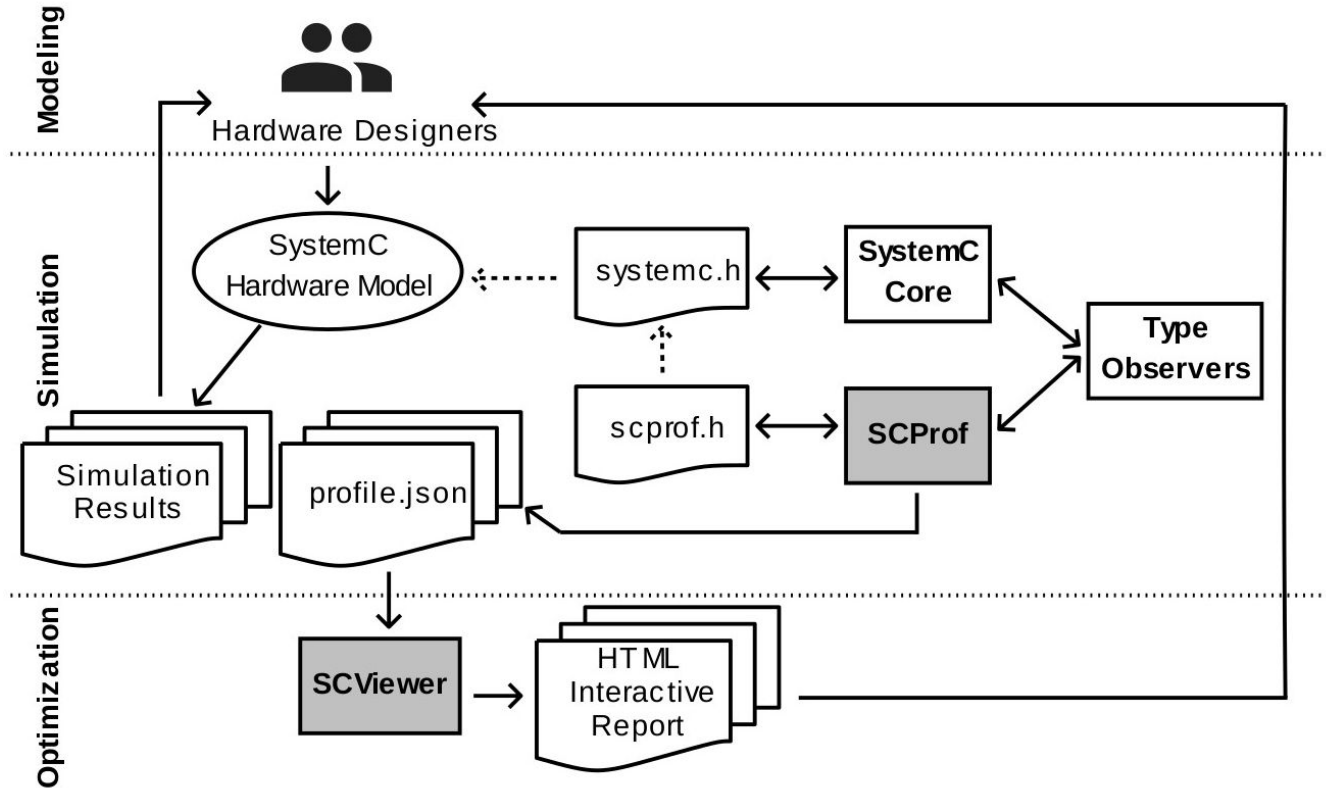


Typical DSP design workflow

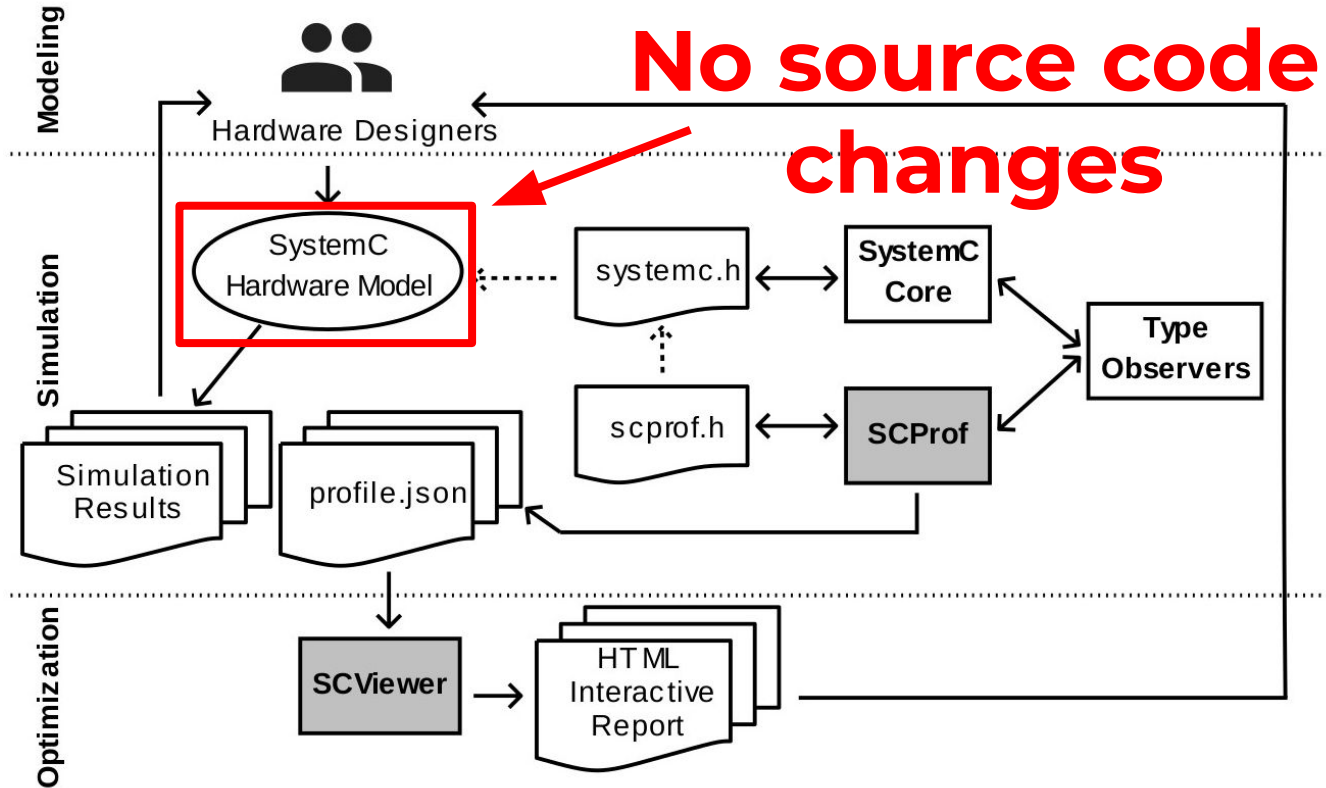


Profiling Infrastructure

Profiling Infrastructure

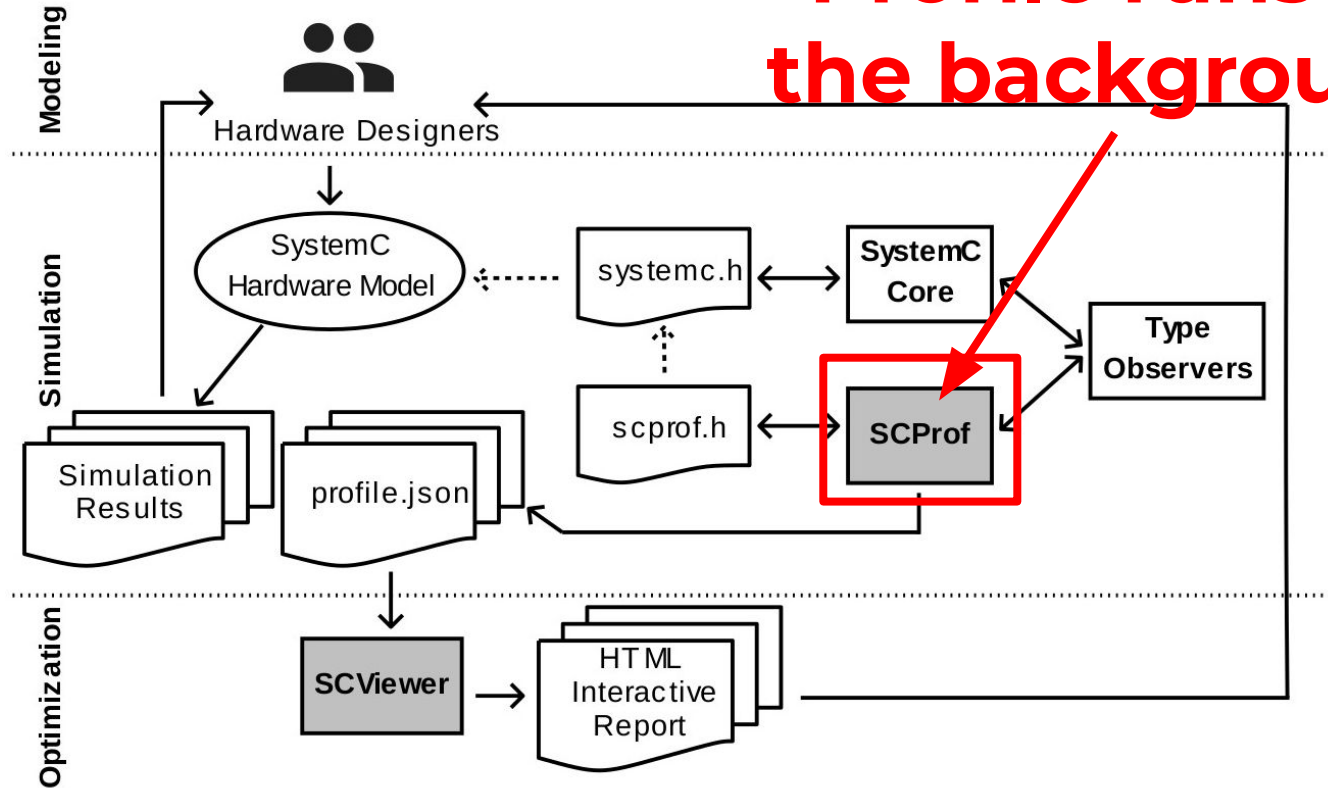


Profiling Infrastructure

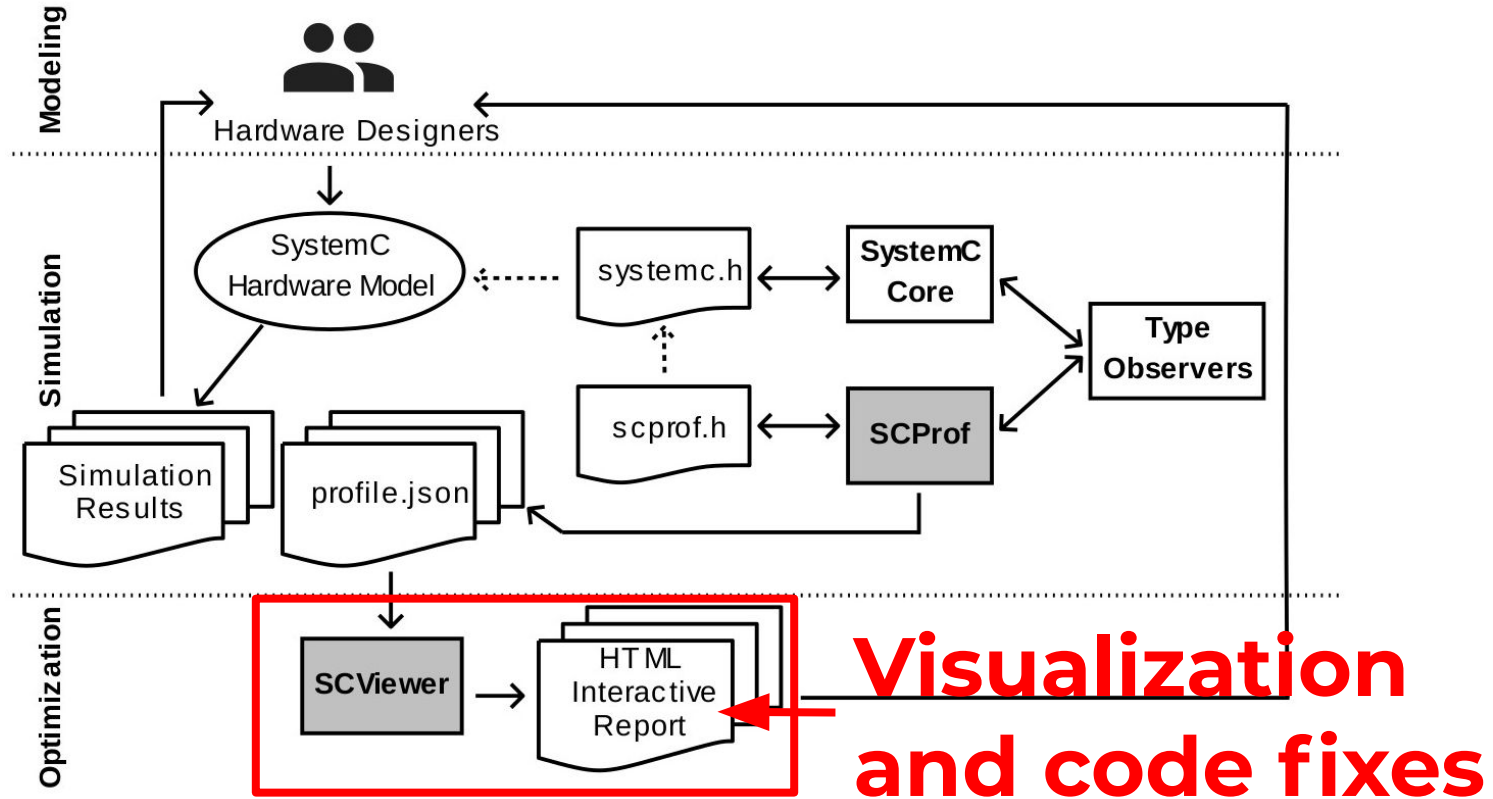


Profiling Infrastructure

Profile runs in the background



Profiling Infrastructure



SCView

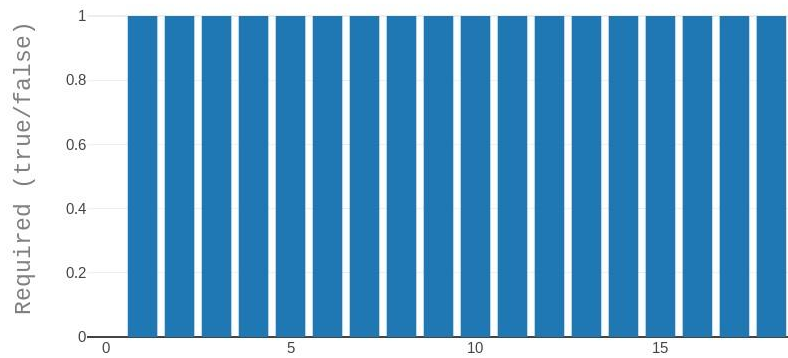
```
filt_decim.cpp
25  sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
26  bool stage_control_1 = false;
27
28  // signals for STAGE 2
29  sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
30  sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
31  sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
32  sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
33  bool stage_control_2 = false;
34
35  // signals for STAGE 3
36  sc_fixed<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  incoef2_var[x] = incoef2[x].read();
84
85  }
86  for (x = 0; x < TAPS_STAGE3; x++) {
87  incoef3_var[x] = incoef3[x].read();
88
89  }
90  for (x = 0; x < TAPS_STAGE4; x++) {
91  incoef4_var[x] = incoef4[x].read();
92
93  }
94  for (x = 0; x < TAPS_STAGE5; x++) {
95  incoef5_var[x] = incoef5[x].read();
96
97  }
98
99  // Main filter body
100  for (x = 0; x < TAPS_STAGE1; x++) {
101  in_stage_1[x] = incoef1_var[x] * stage_1[x];
102  }
103  for (x = 0; x < TAPS_STAGE2; x++) {
104  in_stage_2[x] = incoef2_var[x] * stage_2[x];
105  }
106  for (x = 0; x < TAPS_STAGE3; x++) {
107  in_stage_3[x] = incoef3_var[x] * stage_3[x];
108  }
109  for (x = 0; x < TAPS_STAGE4; x++) {
110  in_stage_4[x] = incoef4_var[x] * stage_4[x];
111  }
112  for (x = 0; x < TAPS_STAGE5; x++) {
113  in_stage_5[x] = incoef5_var[x] * stage_5[x];
114  }
115
116  // Main filter body
117  for (x = 0; x < TAPS_STAGE1; x++) {
118  buffer_1[x] = in_stage_1[x];
119  }
120  for (x = 0; x < TAPS_STAGE2; x++) {
121  buffer_2[x] = in_stage_2[x];
122  }
123  for (x = 0; x < TAPS_STAGE3; x++) {
124  buffer_3[x] = in_stage_3[x];
125  }
126  for (x = 0; x < TAPS_STAGE4; x++) {
127  buffer_4[x] = in_stage_4[x];
128  }
129  for (x = 0; x < TAPS_STAGE5; x++) {
130  buffer_5[x] = in_stage_5[x];
131  }
132
133  // Main filter body
134  for (x = 0; x < TAPS_STAGE1; x++) {
135  SoP1[x] = buffer_1[x];
136  }
137  for (x = 0; x < TAPS_STAGE2; x++) {
138  SoP2[x] = buffer_2[x];
139  }
140  for (x = 0; x < TAPS_STAGE3; x++) {
141  SoP3[x] = buffer_3[x];
142  }
143  for (x = 0; x < TAPS_STAGE4; x++) {
144  SoP4[x] = buffer_4[x];
145  }
146  for (x = 0; x < TAPS_STAGE5; x++) {
147  SoP5[x] = buffer_5[x];
148  }
149
150  // Main filter body
151  for (x = 0; x < TAPS_STAGE1; x++) {
152  stage_1[x] = SoP1[x];
153  }
154  for (x = 0; x < TAPS_STAGE2; x++) {
155  stage_2[x] = SoP2[x];
156  }
157  for (x = 0; x < TAPS_STAGE3; x++) {
158  stage_3[x] = SoP3[x];
159  }
160  for (x = 0; x < TAPS_STAGE4; x++) {
161  stage_4[x] = SoP4[x];
162  }
163  for (x = 0; x < TAPS_STAGE5; x++) {
164  stage_5[x] = SoP5[x];
165  }
166
167  // Main filter body
168  for (x = 0; x < TAPS_STAGE1; x++) {
169  stage_control_1 = true;
170  }
171  for (x = 0; x < TAPS_STAGE2; x++) {
172  stage_control_2 = true;
173  }
174  for (x = 0; x < TAPS_STAGE3; x++) {
175  stage_control_3 = true;
176  }
177  for (x = 0; x < TAPS_STAGE4; x++) {
178  stage_control_4 = true;
179  }
180  for (x = 0; x < TAPS_STAGE5; x++) {
181  stage_control_5 = true;
182  }
183
184  // Main filter body
185  for (x = 0; x < TAPS_STAGE1; x++) {
186  stage_1[x] = SoP1[x];
187  }
188  for (x = 0; x < TAPS_STAGE2; x++) {
189  stage_2[x] = SoP2[x];
190  }
191  for (x = 0; x < TAPS_STAGE3; x++) {
192  stage_3[x] = SoP3[x];
193  }
194  for (x = 0; x < TAPS_STAGE4; x++) {
195  stage_4[x] = SoP4[x];
196  }
197  for (x = 0; x < TAPS_STAGE5; x++) {
198  stage_5[x] = SoP5[x];
199  }
200
201  // Main filter body
202  for (x = 0; x < TAPS_STAGE1; x++) {
203  stage_control_1 = false;
204  }
205  for (x = 0; x < TAPS_STAGE2; x++) {
206  stage_control_2 = false;
207  }
208  for (x = 0; x < TAPS_STAGE3; x++) {
209  stage_control_3 = false;
210  }
211  for (x = 0; x < TAPS_STAGE4; x++) {
212  stage_control_4 = false;
213  }
214  for (x = 0; x < TAPS_STAGE5; x++) {
215  stage_control_5 = false;
216  }
217
218  // Main filter body
219  for (x = 0; x < TAPS_STAGE1; x++) {
220  stage_1[x] = SoP1[x];
221  }
222  for (x = 0; x < TAPS_STAGE2; x++) {
223  stage_2[x] = SoP2[x];
224  }
225  for (x = 0; x < TAPS_STAGE3; x++) {
226  stage_3[x] = SoP3[x];
227  }
228  for (x = 0; x < TAPS_STAGE4; x++) {
229  stage_4[x] = SoP4[x];
230  }
231  for (x = 0; x < TAPS_STAGE5; x++) {
232  stage_5[x] = SoP5[x];
233  }
234
235  // Main filter body
236  for (x = 0; x < TAPS_STAGE1; x++) {
237  stage_control_1 = true;
238  }
239  for (x = 0; x < TAPS_STAGE2; x++) {
240  stage_control_2 = true;
241  }
242  for (x = 0; x < TAPS_STAGE3; x++) {
243  stage_control_3 = true;
244  }
245  for (x = 0; x < TAPS_STAGE4; x++) {
246  stage_control_4 = true;
247  }
248  for (x = 0; x < TAPS_STAGE5; x++) {
249  stage_control_5 = true;
250  }
251
252  // Main filter body
253  for (x = 0; x < TAPS_STAGE1; x++) {
254  stage_1[x] = SoP1[x];
255  }
256  for (x = 0; x < TAPS_STAGE2; x++) {
257  stage_2[x] = SoP2[x];
258  }
259  for (x = 0; x < TAPS_STAGE3; x++) {
260  stage_3[x] = SoP3[x];
261  }
262  for (x = 0; x < TAPS_STAGE4; x++) {
263  stage_4[x] = SoP4[x];
264  }
265  for (x = 0; x < TAPS_STAGE5; x++) {
266  stage_5[x] = SoP5[x];
267  }
268
269  // Main filter body
270  for (x = 0; x < TAPS_STAGE1; x++) {
271  stage_control_1 = false;
272  }
273  for (x = 0; x < TAPS_STAGE2; x++) {
274  stage_control_2 = false;
275  }
276  for (x = 0; x < TAPS_STAGE3; x++) {
277  stage_control_3 = false;
278  }
279  for (x = 0; x < TAPS_STAGE4; x++) {
280  stage_control_4 = false;
281  }
282  for (x = 0; x < TAPS_STAGE5; x++) {
283  stage_control_5 = false;
284  }
285
286  // Main filter body
287  for (x = 0; x < TAPS_STAGE1; x++) {
288  stage_1[x] = SoP1[x];
289  }
290  for (x = 0; x < TAPS_STAGE2; x++) {
291  stage_2[x] = SoP2[x];
292  }
293  for (x = 0; x < TAPS_STAGE3; x++) {
294  stage_3[x] = SoP3[x];
295  }
296  for (x = 0; x < TAPS_STAGE4; x++) {
297  stage_4[x] = SoP4[x];
298  }
299  for (x = 0; x < TAPS_STAGE5; x++) {
300  stage_5[x] = SoP5[x];
301  }
302
303  // Main filter body
304  for (x = 0; x < TAPS_STAGE1; x++) {
305  stage_control_1 = true;
306  }
307  for (x = 0; x < TAPS_STAGE2; x++) {
308  stage_control_2 = true;
309  }
310  for (x = 0; x < TAPS_STAGE3; x++) {
311  stage_control_3 = true;
312  }
313  for (x = 0; x < TAPS_STAGE4; x++) {
314  stage_control_4 = true;
315  }
316  for (x = 0; x < TAPS_STAGE5; x++) {
317  stage_control_5 = true;
318  }
319
320  // Main filter body
321  for (x = 0; x < TAPS_STAGE1; x++) {
322  stage_1[x] = SoP1[x];
323  }
324  for (x = 0; x < TAPS_STAGE2; x++) {
325  stage_2[x] = SoP2[x];
326  }
327  for (x = 0; x < TAPS_STAGE3; x++) {
328  stage_3[x] = SoP3[x];
329  }
330  for (x = 0; x < TAPS_STAGE4; x++) {
331  stage_4[x] = SoP4[x];
332  }
333  for (x = 0; x < TAPS_STAGE5; x++) {
334  stage_5[x] = SoP5[x];
335  }
336
337  // Main filter body
338  for (x = 0; x < TAPS_STAGE1; x++) {
339  stage_control_1 = false;
340  }
341  for (x = 0; x < TAPS_STAGE2; x++) {
342  stage_control_2 = false;
343  }
344  for (x = 0; x < TAPS_STAGE3; x++) {
345  stage_control_3 = false;
346  }
347  for (x = 0; x < TAPS_STAGE4; x++) {
348  stage_control_4 = false;
349  }
350  for (x = 0; x < TAPS_STAGE5; x++) {
351  stage_control_5 = false;
352  }
353
354  // Main filter body
355  for (x = 0; x < TAPS_STAGE1; x++) {
356  stage_1[x] = SoP1[x];
357  }
358  for (x = 0; x < TAPS_STAGE2; x++) {
359  stage_2[x] = SoP2[x];
360  }
361  for (x = 0; x < TAPS_STAGE3; x++) {
362  stage_3[x] = SoP3[x];
363  }
364  for (x = 0; x < TAPS_STAGE4; x++) {
365  stage_4[x] = SoP4[x];
366  }
367  for (x = 0; x < TAPS_STAGE5; x++) {
368  stage_5[x] = SoP5[x];
369  }
370
371  // Main filter body
372  for (x = 0; x < TAPS_STAGE1; x++) {
373  stage_control_1 = true;
374  }
375  for (x = 0; x < TAPS_STAGE2; x++) {
376  stage_control_2 = true;
377  }
378  for (x = 0; x < TAPS_STAGE3; x++) {
379  stage_control_3 = true;
380  }
381  for (x = 0; x < TAPS_STAGE4; x++) {
382  stage_control_4 = true;
383  }
384  for (x = 0; x < TAPS_STAGE5; x++) {
385  stage_control_5 = true;
386  }
387
388  // Main filter body
389  for (x = 0; x < TAPS_STAGE1; x++) {
390  stage_1[x] = SoP1[x];
391  }
392  for (x = 0; x < TAPS_STAGE2; x++) {
393  stage_2[x] = SoP2[x];
394  }
395  for (x = 0; x < TAPS_STAGE3; x++) {
396  stage_3[x] = SoP3[x];
397  }
398  for (x = 0; x < TAPS_STAGE4; x++) {
399  stage_4[x] = SoP4[x];
400  }
401  for (x = 0; x < TAPS_STAGE5; x++) {
402  stage_5[x] = SoP5[x];
403  }
404
405  // Main filter body
406  for (x = 0; x < TAPS_STAGE1; x++) {
407  stage_control_1 = false;
408  }
409  for (x = 0; x < TAPS_STAGE2; x++) {
410  stage_control_2 = false;
411  }
412  for (x = 0; x < TAPS_STAGE3; x++) {
413  stage_control_3 = false;
414  }
415  for (x = 0; x < TAPS_STAGE4; x++) {
416  stage_control_4 = false;
417  }
418  for (x = 0; x < TAPS_STAGE5; x++) {
419  stage_control_5 = false;
420  }
421
422  // Main filter body
423  for (x = 0; x < TAPS_STAGE1; x++) {
424  stage_1[x] = SoP1[x];
425  }
426  for (x = 0; x < TAPS_STAGE2; x++) {
427  stage_2[x] = SoP2[x];
428  }
429  for (x = 0; x < TAPS_STAGE3; x++) {
430  stage_3[x] = SoP3[x];
431  }
432  for (x = 0; x < TAPS_STAGE4; x++) {
433  stage_4[x] = SoP4[x];
434  }
435  for (x = 0; x < TAPS_STAGE5; x++) {
436  stage_5[x] = SoP5[x];
437  }
438
439  // Main filter body
440  for (x = 0; x < TAPS_STAGE1; x++) {
441  stage_control_1 = true;
442  }
443  for (x = 0; x < TAPS_STAGE2; x++) {
444  stage_control_2 = true;
445  }
446  for (x = 0; x < TAPS_STAGE3; x++) {
447  stage_control_3 = true;
448  }
449  for (x = 0; x < TAPS_STAGE4; x++) {
450  stage_control_4 = true;
451  }
452  for (x = 0; x < TAPS_STAGE5; x++) {
453  stage_control_5 = true;
454  }
455
456  // Main filter body
457  for (x = 0; x < TAPS_STAGE1; x++) {
458  stage_1[x] = SoP1[x];
459  }
460  for (x = 0; x < TAPS_STAGE2; x++) {
461  stage_2[x] = SoP2[x];
462  }
463  for (x = 0; x < TAPS_STAGE3; x++) {
464  stage_3[x] = SoP3[x];
465  }
466  for (x = 0; x < TAPS_STAGE4; x++) {
467  stage_4[x] = SoP4[x];
468  }
469  for (x = 0; x < TAPS_STAGE5; x++) {
470  stage_5[x] = SoP5[x];
471  }
472
473  // Main filter body
474  for (x = 0; x < TAPS_STAGE1; x++) {
475  stage_control_1 = false;
476  }
477  for (x = 0; x < TAPS_STAGE2; x++) {
478  stage_control_2 = false;
479  }
480  for (x = 0; x < TAPS_STAGE3; x++) {
481  stage_control_3 = false;
482  }
483  for (x = 0; x < TAPS_STAGE4; x++) {
484  stage_control_4 = false;
485  }
486  for (x = 0; x < TAPS_STAGE5; x++) {
487  stage_control_5 = false;
488  }
489
490  // Main filter body
491  for (x = 0; x < TAPS_STAGE1; x++) {
492  stage_1[x] = SoP1[x];
493  }
494  for (x = 0; x < TAPS_STAGE2; x++) {
495  stage_2[x] = SoP2[x];
496  }
497  for (x = 0; x < TAPS_STAGE3; x++) {
498  stage_3[x] = SoP3[x];
499  }
500  for (x = 0; x < TAPS_STAGE4; x++) {
501  stage_4[x] = SoP4[x];
502  }
503  for (x = 0; x < TAPS_STAGE5; x++) {
504  stage_5[x] = SoP5[x];
505  }
506
507  // Main filter body
508  for (x = 0; x < TAPS_STAGE1; x++) {
509  stage_control_1 = true;
510  }
511  for (x = 0; x < TAPS_STAGE2; x++) {
512  stage_control_2 = true;
513  }
514  for (x = 0; x < TAPS_STAGE3; x++) {
515  stage_control_3 = true;
516  }
517  for (x = 0; x < TAPS_STAGE4; x++) {
518  stage_control_4 = true;
519  }
520  for (x = 0; x < TAPS_STAGE5; x++) {
521  stage_control_5 = true;
522  }
523
524  // Main filter body
525  for (x = 0; x < TAPS_STAGE1; x++) {
526  stage_1[x] = SoP1[x];
527  }
528  for (x = 0; x < TAPS_STAGE2; x++) {
529  stage_2[x] = SoP2[x];
530  }
531  for (x = 0; x < TAPS_STAGE3; x++) {
532  stage_3[x] = SoP3[x];
533  }
534  for (x = 0; x < TAPS_STAGE4; x++) {
535  stage_4[x] = SoP4[x];
536  }
537  for (x = 0; x < TAPS_STAGE5; x++) {
538  stage_5[x] = SoP5[x];
539  }
540
541  // Main filter body
542  for (x = 0; x < TAPS_STAGE1; x++) {
543  stage_control_1 = false;
544  }
545  for (x = 0; x < TAPS_STAGE2; x++) {
546  stage_control_2 = false;
547  }
548  for (x = 0; x < TAPS_STAGE3; x++) {
549  stage_control_3 = false;
550  }
551  for (x = 0; x < TAPS_STAGE4; x++) {
552  stage_control_4 = false;
553  }
554  for (x = 0; x < TAPS_STAGE5; x++) {
555  stage_control_5 = false;
556  }
557
558  // Main filter body
559  for (x = 0; x < TAPS_STAGE1; x++) {
560  stage_1[x] = SoP1[x];
561  }
562  for (x = 0; x < TAPS_STAGE2; x++) {
563  stage_2[x] = SoP2[x];
564  }
565  for (x = 0; x < TAPS_STAGE3; x++) {
566  stage_3[x] = SoP3[x];
567  }
568  for (x = 0; x < TAPS_STAGE4; x++) {
569  stage_4[x] = SoP4[x];
570  }
571  for (x = 0; x < TAPS_STAGE5; x++) {
572  stage_5[x] = SoP5[x];
573  }
574
575  // Main filter body
576  for (x = 0; x < TAPS_STAGE1; x++) {
577  stage_control_1 = true;
578  }
579  for (x = 0; x < TAPS_STAGE2; x++) {
580  stage_control_2 = true;
581  }
582  for (x = 0; x < TAPS_STAGE3; x++) {
583  stage_control_3 = true;
584  }
585  for (x = 0; x < TAPS_STAGE4; x++) {
586  stage_control_4 = true;
587  }
588  for (x = 0; x < TAPS_STAGE5; x++) {
589  stage_control_5 = true;
590  }
591
592  // Main filter body
593  for (x = 0; x < TAPS_STAGE1; x++) {
594  stage_1[x] = SoP1[x];
595  }
596  for (x = 0; x < TAPS_STAGE2; x++) {
597  stage_2[x] = SoP2[x];
598  }
599  for (x = 0; x < TAPS_STAGE3; x++) {
600  stage_3[x] = SoP3[x];
601  }
602  for (x = 0; x < TAPS_STAGE4; x++) {
603  stage_4[x] = SoP4[x];
604  }
605  for (x = 0; x < TAPS_STAGE5; x++) {
606  stage_5[x] = SoP5[x];
607  }
608
609  // Main filter body
610  for (x = 0; x < TAPS_STAGE1; x++) {
611  stage_control_1 = false;
612  }
613  for (x = 0; x < TAPS_STAGE2; x++) {
614  stage_control_2 = false;
615  }
616  for (x = 0; x < TAPS_STAGE3; x++) {
617  stage_control_3 = false;
618  }
619  for (x = 0; x < TAPS_STAGE4; x++) {
620  stage_control_4 = false;
621  }
622  for (x = 0; x < TAPS_STAGE5; x++) {
623  stage_control_5 = false;
624  }
625
626  // Main filter body
627  for (x = 0; x < TAPS_STAGE1; x++) {
628  stage_1[x] = SoP1[x];
629  }
630  for (x = 0; x < TAPS_STAGE2; x++) {
631  stage_2[x] = SoP2[x];
632  }
633  for (x = 0; x < TAPS_STAGE3; x++) {
634  stage_3[x] = SoP3[x];
635  }
636  for (x = 0; x < TAPS_STAGE4; x++) {
637  stage_4[x] = SoP4[x];
638  }
639  for (x = 0; x < TAPS_STAGE5; x++) {
640  stage_5[x] = SoP5[x];
641  }
642
643  // Main filter body
644  for (x = 0; x < TAPS_STAGE1; x++) {
645  stage_control_1 = true;
646  }
647  for (x = 0; x < TAPS_STAGE2; x++) {
648  stage_control_2 = true;
649  }
650  for (x = 0; x < TAPS_STAGE3; x++) {
651  stage_control_3 = true;
652  }
653  for (x = 0; x < TAPS_STAGE4; x++) {
654  stage_control_4 = true;
655  }
656  for (x = 0; x < TAPS_STAGE5; x++) {
657  stage_control_5 = true;
658  }
659
660  // Main filter body
661  for (x = 0; x < TAPS_STAGE1; x++) {
662  stage_1[x] = SoP1[x];
663  }
664  for (x = 0; x < TAPS_STAGE2; x++) {
665  stage_2[x] = SoP2[x];
666  }
667  for (x = 0; x < TAPS_STAGE3; x++) {
668  stage_3[x] = SoP3[x];
669  }
670  for (x = 0; x < TAPS_STAGE4; x++) {
671  stage_4[x] = SoP4[x];
672  }
673  for (x = 0; x < TAPS_STAGE5; x++) {
674  stage_5[x] = SoP5[x];
675  }
676
677  // Main filter body
678  for (x = 0; x < TAPS_STAGE1; x++) {
679  stage_control_1 = false;
680  }
681  for (x = 0; x < TAPS_STAGE2; x++) {
682  stage_control_2 = false;
683  }
684  for (x = 0; x < TAPS_STAGE3; x++) {
685  stage_control_3 = false;
686  }
687  for (x = 0; x < TAPS_STAGE4; x++) {
688  stage_control_4 = false;
689  }
690  for (x = 0; x < TAPS_STAGE5; x++) {
691  stage_control_5 = false;
692  }
693
694  // Main filter body
695  for (x = 0; x < TAPS_STAGE1; x++) {
696  stage_1[x] = SoP1[x];
697  }
698  for (x = 0; x < TAPS_STAGE2; x++) {
699  stage_2[x] = SoP2[x];
700  }
701  for (x = 0; x < TAPS_STAGE3; x++) {
702  stage_3[x] = SoP3[x];
703  }
704  for (x = 0; x < TAPS_STAGE4; x++) {
705  stage_4[x] = SoP4[x];
706  }
707  for (x = 0; x < TAPS_STAGE5; x++) {
708  stage_5[x] = SoP5[x];
709  }
710
711  // Main filter body
712  for (x = 0; x < TAPS_STAGE1; x++) {
713  stage_control_1 = true;
714  }
715  for (x = 0; x < TAPS_STAGE2; x++) {
716  stage_control_2 = true;
717  }
718  for (x = 0; x < TAPS_STAGE3; x++) {
719  stage_control_3 = true;
720  }
721  for (x = 0; x < TAPS_STAGE4; x++) {
722  stage_control_4 = true;
723  }
724  for (x = 0; x < TAPS_STAGE5; x++) {
725  stage_control_5 = true;
726  }
727
728  // Main filter body
729  for (x = 0; x < TAPS_STAGE1; x++) {
730  stage_1[x] = SoP1[x];
731  }
732  for (x = 0; x < TAPS_STAGE2; x++) {
733  stage_2[x] = SoP2[x];
734  }
735  for (x = 0; x < TAPS_STAGE3; x++) {
736  stage_3[x] = SoP3[x];
737  }
738  for (x = 0; x < TAPS_STAGE4; x++) {
739  stage_4[x] = SoP4[x];
740  }
741  for (x = 0; x < TAPS_STAGE5; x++) {
742  stage_5[x] = SoP5[x];
743  }
744
745  // Main filter body
746  for (x = 0; x < TAPS_STAGE1; x++) {
747  stage_control_1 = false;
748  }
749  for (x = 0; x < TAPS_STAGE2; x++) {
750  stage_control_2 = false;
751  }
752  for (x = 0; x < TAPS_STAGE3; x++) {
753  stage_control_3 = false;
754  }
755  for (x = 0; x < TAPS_STAGE4; x++) {
756  stage_control_4 = false;
757  }
758  for (x = 0; x < TAPS_STAGE5; x++) {
759  stage_control_5 = false;
760  }
761
762  // Main filter body
763  for (x = 0; x < TAPS_STAGE1; x++) {
764  stage_1[x] = SoP1[x];
765  }
766  for (x = 0; x < TAPS_STAGE2; x++) {
767  stage_2[x] = SoP2[x];
768  }
769  for (x = 0; x < TAPS_STAGE3; x++) {
770  stage_3[x] = SoP3[x];
771  }
772  for (x = 0; x < TAPS_STAGE4; x++) {
773  stage_4[x] = SoP4[x];
774  }
775  for (x = 0; x < TAPS_STAGE5; x++) {
776  stage_5[x] = SoP5[x];
777  }
778
779  // Main filter body
780  for (x = 0; x < TAPS_STAGE1; x++) {
781  stage_control_1 = true;
782  }
783  for (x = 0; x < TAPS_STAGE2; x++) {
784  stage_control_2 = true;
785  }
786  for (x = 0; x < TAPS_STAGE3; x++) {
787  stage_control_3 = true;
788  }
789  for (x = 0; x < TAPS_STAGE4; x++) {
790  stage_control_4 = true;
791  }
792  for (x = 0; x < TAPS_STAGE5; x++) {
793  stage_control_5 = true;
794  }
795
796  // Main filter body
797  for (x = 0; x < TAPS_STAGE1; x++) {
798  stage_1[x] = SoP1[x];
799  }
800  for (x = 0; x < TAPS_STAGE2; x++) {
801  stage_2[x] = SoP2[x];
802  }
803  for (x = 0; x < TAPS_STAGE3; x++) {
804  stage_3[x] = SoP3[x];
805  }
806  for (x = 0; x < TAPS_STAGE4; x++) {
807  stage_4[x] = SoP4[x];
808  }
809  for (x = 0; x < TAPS_STAGE5; x++) {
810  stage_5[x] = SoP5[x];
811  }
812
813  // Main filter body
814  for (x = 0; x < TAPS_STAGE1; x++) {
815  stage_control_1 = false;
816  }
817  for (x = 0; x < TAPS_STAGE2; x++) {
818  stage_control_2 = false;
819  }
820  for (x = 0; x < TAPS_STAGE3; x++) {
821  stage_control_3 = false;
822  }
823  for (x = 0; x < TAPS_STAGE4; x++) {
824  stage_control_4 = false;
825  }
826  for (x = 0; x < TAPS_STAGE5; x++) {
827  stage_control_5 = false;
828  }
829
830  // Main filter body
831  for (x = 0; x < TAPS_STAGE1; x++) {
832  stage_1[x] = SoP1[x];
833  }
834  for (x = 0; x < TAPS_STAGE2; x++) {
835  stage_2[x] = SoP2[x];
836  }
837  for (x = 0; x < TAPS_STAGE3; x++) {
838  stage_3[x] = SoP3[x];
839  }
840  for (x = 0; x < TAPS_STAGE4; x++) {
841  stage_4[x] = SoP4[x];
842  }
843  for (x = 0; x < TAPS_STAGE5; x++) {
844  stage_5[x] = SoP5[x];
845  }
846
847  // Main filter body
848  for (x = 0; x < TAPS_STAGE1; x++) {
849  stage_control_1 = true;
850  }
851  for (x = 0; x < TAPS_STAGE2; x++) {
852  stage_control_2 = true;
853  }
854  for (x = 0; x < TAPS_STAGE3; x++) {
855  stage_control_3 = true;
856  }
857  for (x = 0; x < TAPS_STAGE4; x++) {
858  stage_control_4 = true;
859  }
860  for (x = 0; x < TAPS_STAGE5; x++) {
861  stage_control_5 = true;
862  }
863
864  // Main filter body
865  for (x = 0; x < TAPS_STAGE1; x++) {
866  stage_1[x] = SoP1[x];
867  }
868  for (x = 0; x < TAPS_STAGE2; x++) {
869  stage_2[x] = SoP2[x];
870  }
871  for (x = 0; x < TAPS_STAGE3; x++) {
872  stage_3[x] = SoP3[x];
873  }
874  for (x = 0; x < TAPS_STAGE4; x++) {
875  stage_4[x] = SoP4[x];
876  }
877  for (x = 0; x < TAPS_STAGE5; x++) {
878  stage_5[x] = SoP5[x];
879  }
880
881  // Main filter body
882  for (x = 0; x < TAPS_STAGE1; x++) {
883  stage_control_1 = false;
884  }
885  for (x = 0; x < TAPS_STAGE2; x++) {
886  stage_control_2 = false;
887  }
888  for (x = 0; x < TAPS_STAGE3; x++) {
889  stage_control_3 = false;
890  }
891  for (x = 0; x < TAPS_STAGE4; x++) {
892  stage_control_4 = false;
893  }
894  for (x = 0; x < TAPS_STAGE5; x++) {
895  stage_control_5 = false;
896  }
897
898  // Main filter body
899  for (x = 0; x < TAPS_STAGE1; x++) {
900  stage_1[x] = SoP1[x];
901  }
902  for (x = 0; x < TAPS_STAGE2; x++) {
903  stage_2[x] = SoP2[x];
904  }
905  for (x = 0; x < TAPS_STAGE3; x++) {
906  stage_3[x] = SoP3[x];
907  }
908  for (x = 0; x < TAPS_STAGE4; x++) {
909  stage_4[x] = SoP4[x];
910  }
911  for (x = 0; x < TAPS_STAGE5; x++) {
912  stage_5[x] = SoP5[x];
913  }

```

```
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  // signals for STAGE 4
42  sc_fixed<19,2,SC_RND,SC_SAT> in_stage_4;
43  sc_fixed<19,2,SC_RND,SC_SAT> buffer_4;
44  sc_fixed<36,4,SC_RND,SC_SAT> SoP4;
45  sc_fixed<19, 2, SC_RND, SC_SAT> stage_4;
46  bool stage_control_4 = false;
47
48  // signals for STAGE 5
49  sc_fixed<19,2,SC_RND,SC_SAT> in_stage_5;
50  sc_fixed<19,2,SC_RND,SC_SAT> buffer_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  incoef2_var[x] = incoef2[x].read();
84
85  }
86  for (x = 0; x < TAPS_STAGE3; x++) {
87  incoef3_var[x] = incoef3[x].read();
88
89  }
90  for (x = 0; x < TAPS_STAGE4; x++) {
91  incoef4_var[x] = incoef4[x].read();
92
93  }
94  for (x = 0; x < TAPS_STAGE5; x++) {
95  incoef5_var[x] = incoef5[x].read();
96
97  }
98  }
```

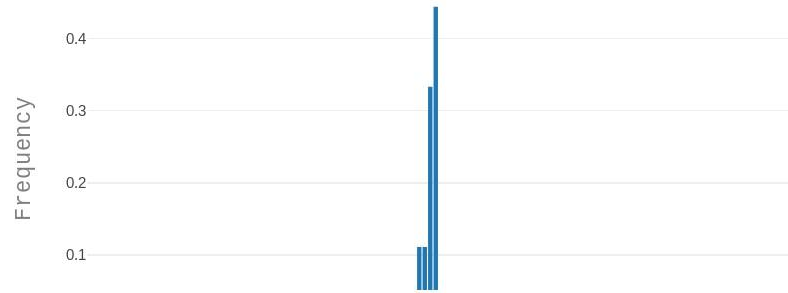
Some bits are never utilized. This may be a logic problem or the data used in your simulation is not representative. If this is not the case, change the precision to WL: 18 and IWL: 1 to improve hardware utilization.

Bit utilization

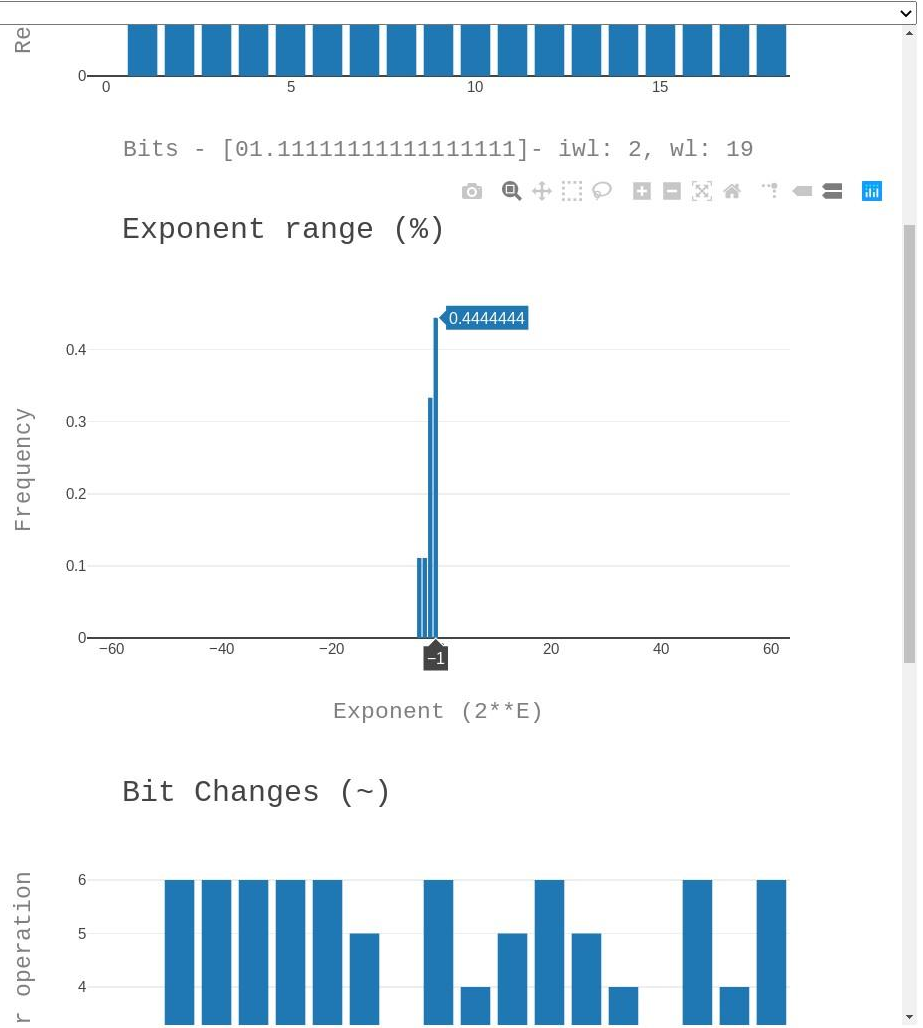


Bits - [01.1111111111111111]- iwl: 2, wl: 19

Exponent range (%)

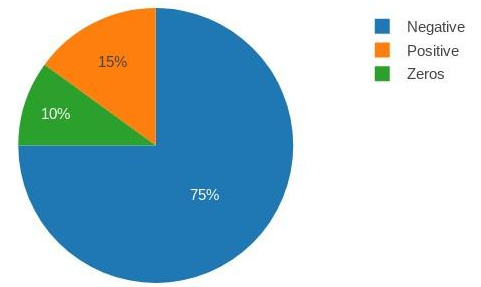
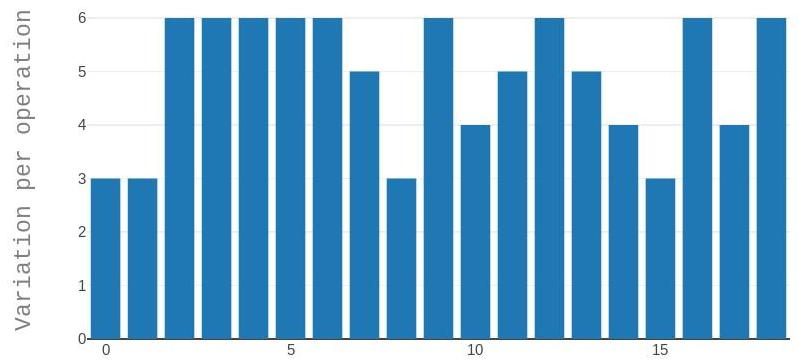


```
filt_decim.cpp
25  sc_fixed<19, 2, SC_RND, SC_SAT> stage_1;
26  bool stage_control_1 = false;
27
28  // signals for STAGE 2
29  sc_fixed<19,2,SC_RND,SC_SAT> in_stage_2;
30  sc_fixed<19,2,SC_RND,SC_SAT> buffer_2;
31  sc_fixed<36,4,SC_RND,SC_SAT> SoP2;
32  sc_fixed<19, 2, SC_RND, SC_SAT> stage_2;
33  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
```




```
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
```

Bit Changes (~)



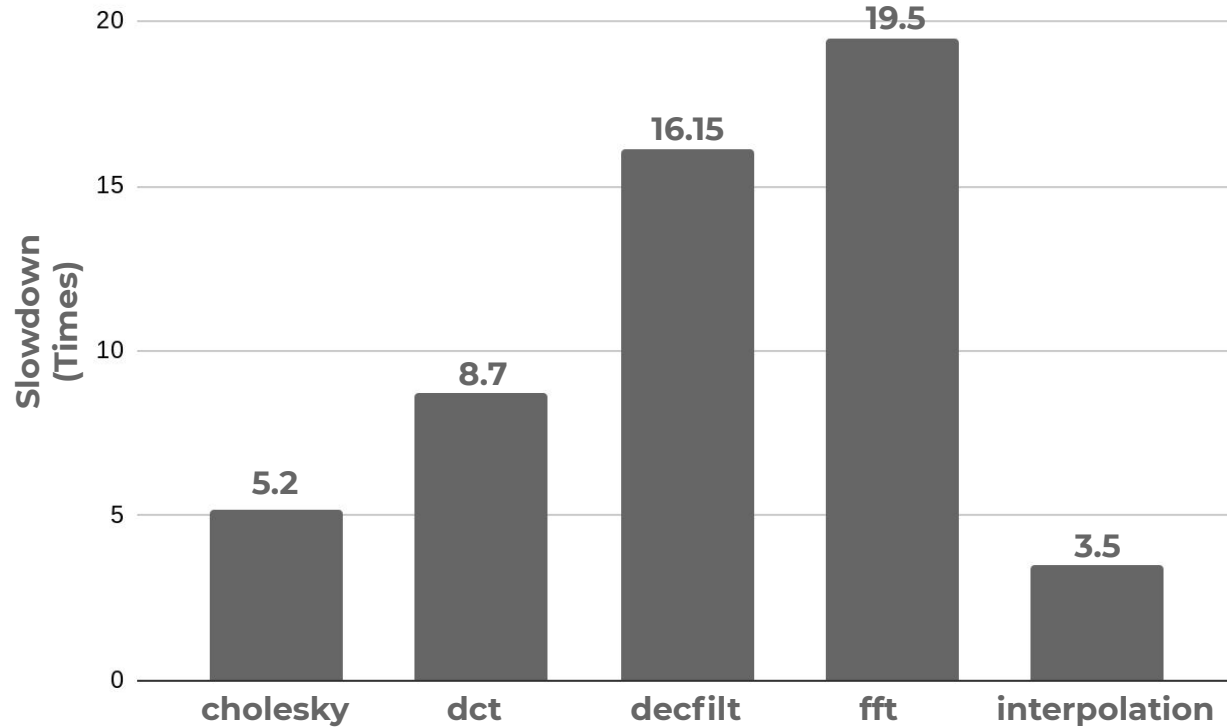
Results

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)

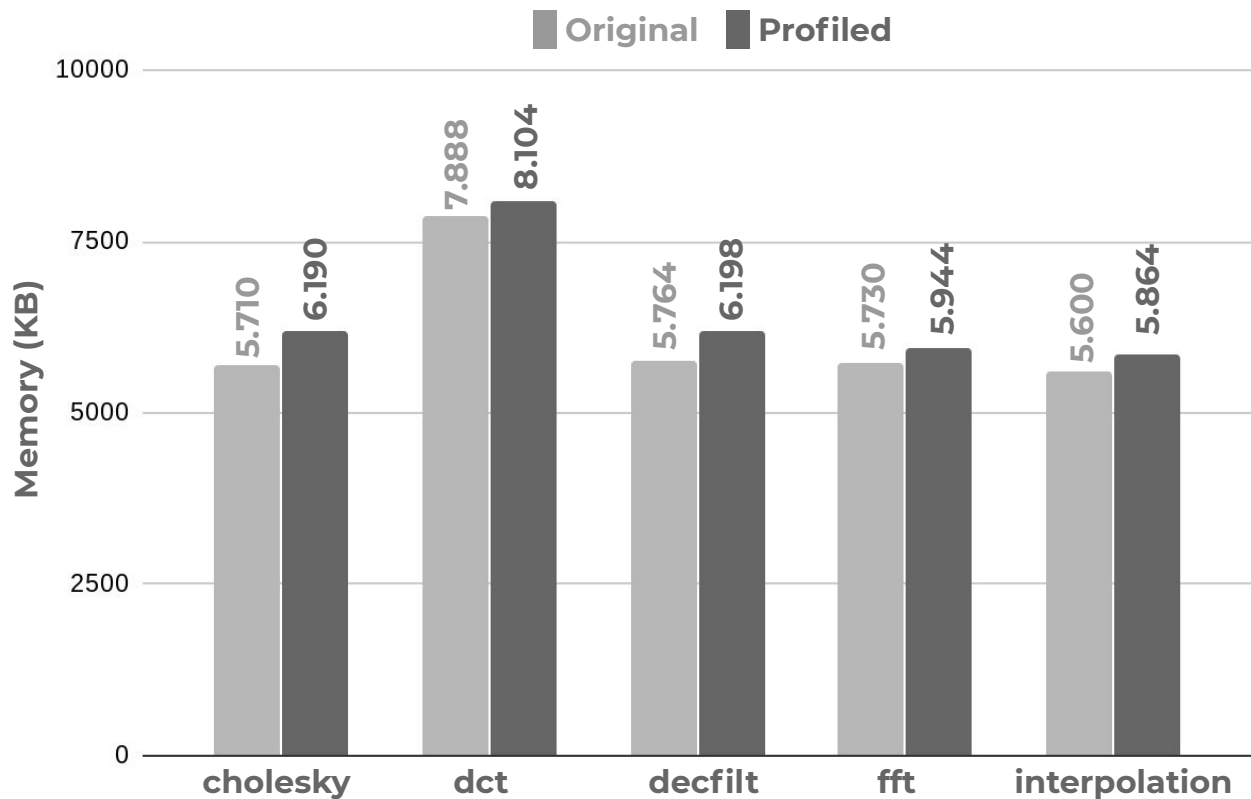
Simulation overhead



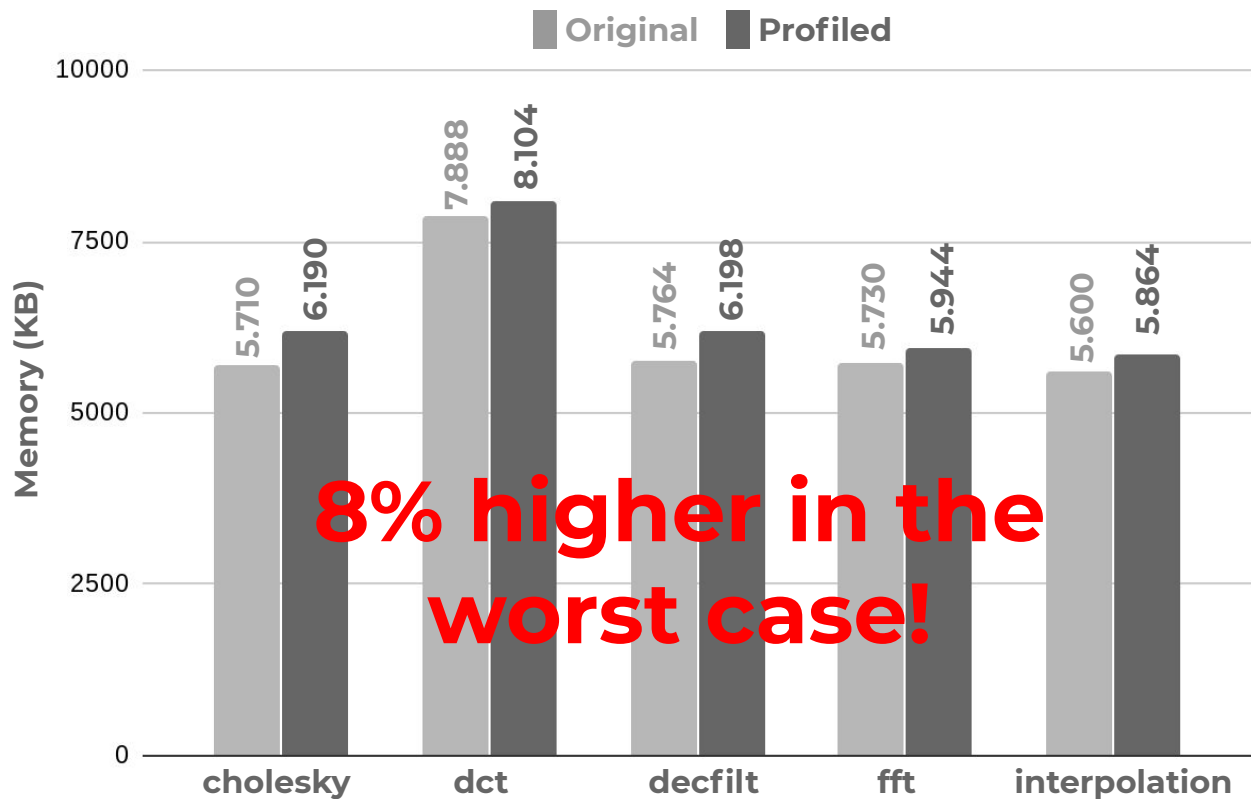
Simulation overhead



Memory usage



Memory usage



8% higher in the worst case!

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	5	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	-7	10	-2	9	-7	10	-2	10	-2	10	-2	9,09%
iqc_acc_a_s_t	16	-2	12	-6	13	-5	14	-4	15	-3	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	-1	5	-7	6	-5	7	-6	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

