Accelerating CCSDS121 in RISC-V with Custom Instructions

Martin Daněk, Roman Bartosiński
daiteq s.r.o., Czech Republic, {martin|roman}@daiteq.com

Overview

This work has been partially performed under an ESA contract 4000122242/17/NL/LF.

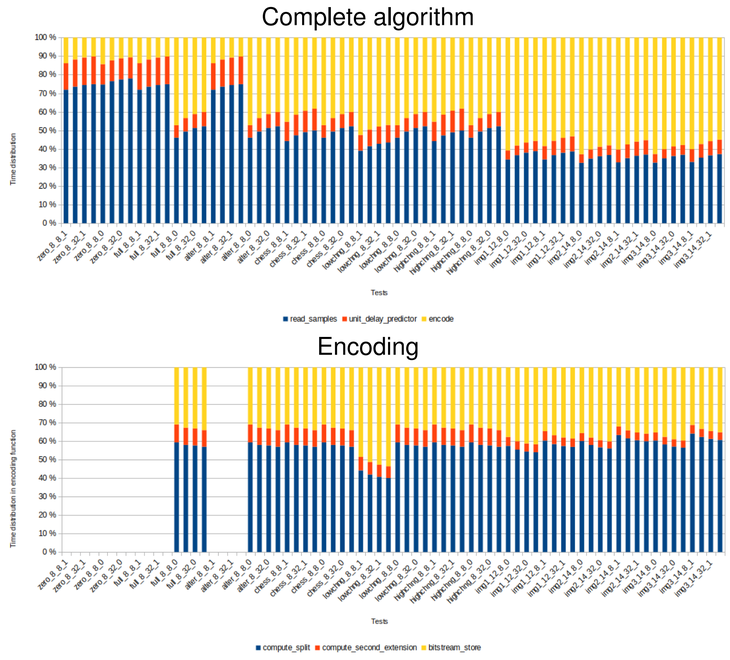
SWAR = SIMD-Within-A-Register
- reuses existing processor datapaths
- increases performance by processing more data values in a single operation
- allows fusing simple operations, e.g. multiply-accumulate

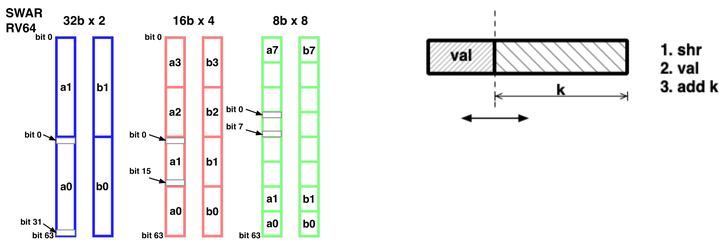
Custom instructions in RISC-V and SPARCv8:
- implemented in the SWAR unit
- general-purpose operations - +,-,*,shr
- domain-specific arithmetic operations
  - GNSS: convolution, demodulation
  - **CCSDS121: k-split**

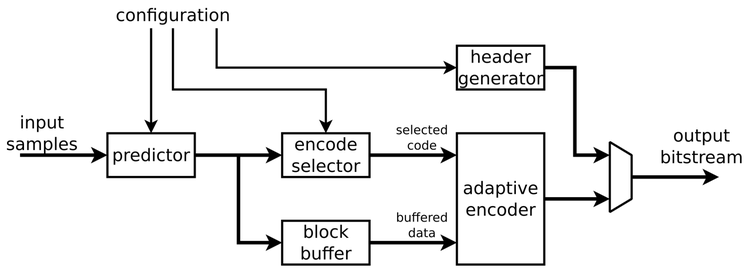
Four **SWAR operations** evaluated for CCSDS121 code selection:
- **right-shift**
- **k-split option (complex, simple)**
- encode selection without second extension
- full code selection

For k-split options the following schemes were evaluated for **readout of the optimal k value**:
- the best *k* value provided part of the SWAR operation,
- all accumulated bitstream lengths read, compared and the best *k* selected in software, accumulator address computed in software,
- like the previous point, but accumulator address auto-incremented on each CSR access,
- like the previous point, but the best value is detected and registered in hardware.

Complete algorithm







Compression with SWAR SHR

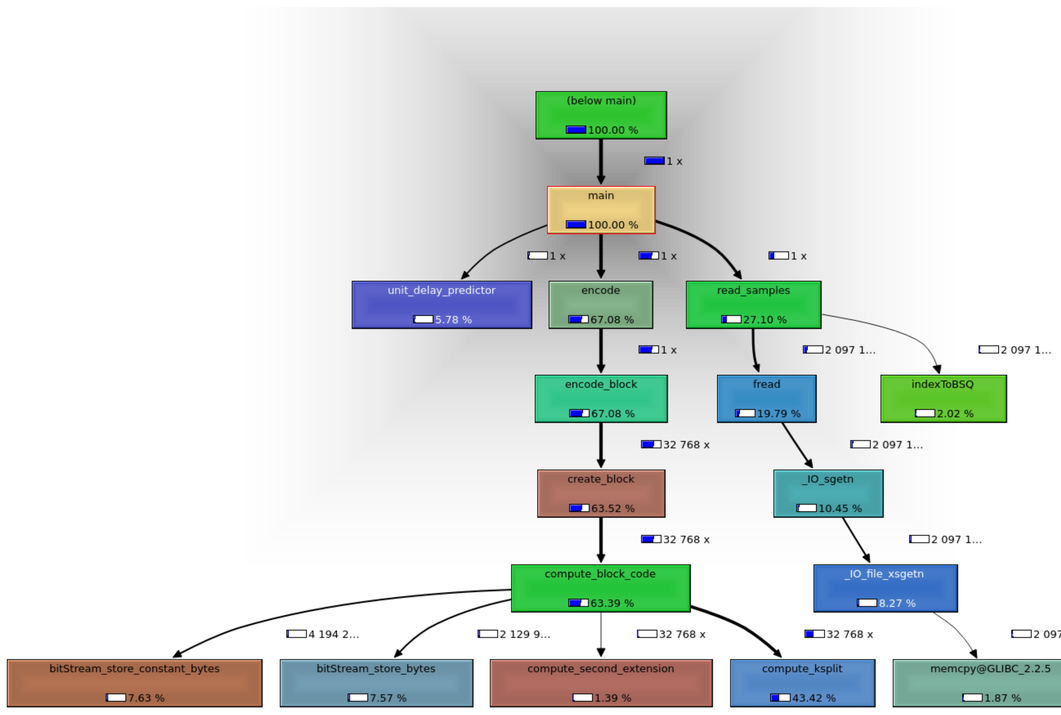
```
for (unsigned k=1;k<=kmax;++k) {
  op_set_ctrl(SW_OP_SHR);                  /* init SWAR to compute SHR */
  for (unsigned s=sfrom;s<blksize;++s) {  /* for all samples in a block (without reference) */
    op_swar(buffer[s], k);
  }
  /* read accumulated value */
  op_set_accum(0x0);
  volatile unsigned long long acc = op_get_accum();
  op_set_accum(0x1);
  acc |= ((unsigned long long)op_get_accum())<<32;
  sel_size[k] += acc + ((unsigned long long)(blksize-sfrom))*(1+k);
}
```

Compression with SWAR k-split

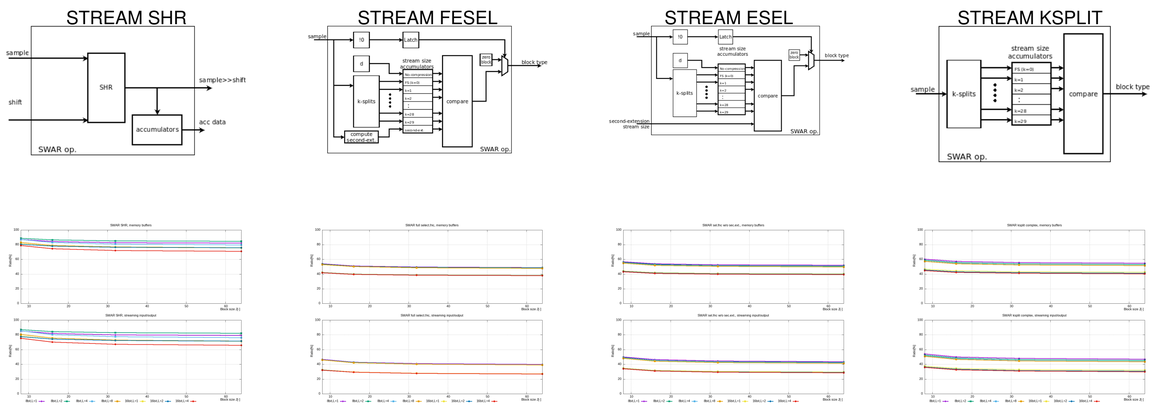
```
// init_block
op_set_ctrl(SW_OP_CCSDS_KSPLIT | is_ref_block | dyn_range);

// encode_selection(sample)
...
  // compute FS + k-split sum for all meaningful k
  bestk = op_swar(sample, 0);
...
  // compare sizes of output bitstream
...
  op_set_accum(0xC0 + bestk);
  unsigned long long ksum = op_get_accum();
  ksum |= (op_get_accum()<<32);
  if (ksum<opt_size) {
    opt_size = ksum;
    opt = bestk;
  }
...
```

CCSDS121 compression



Acceleration



Results

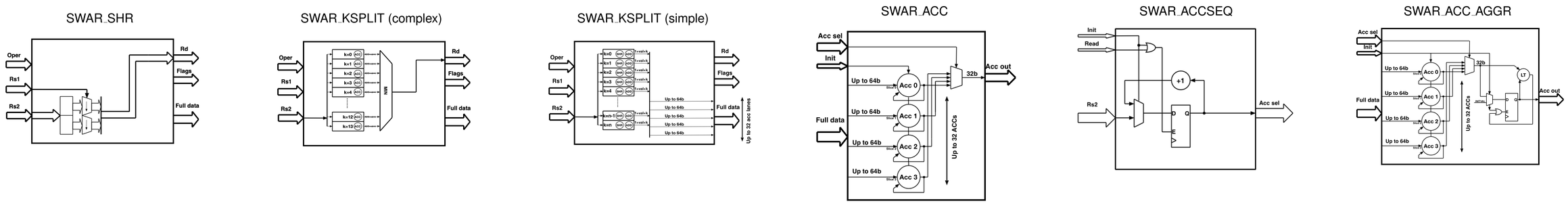
Table 1: Implementation Requirements in XCKU040

Operation	Total LUTs	DFFs	LUT incr [%]	DFF incr [%]
NOEL-V GPP SNGL ISS (NOEL-V GPP DUAL ISS)	46916 (56573)	22111 (24437)		
k-split 8x8b	333	302	0.71	1.37
k-split 4x16b	649	616	1.38	2.79
k-split 2x32b	2047	1246	4.36	5.64
SHyLoC 512x512x32 J=16	10989	2745	23.42	12.41

Table 2: CCSDS121 in Software: Throughput and Power Consumption

Version	CCSDS121 params		IPC	Power	Throughput	Energy per MB
	D	J	[1]	[W]	[Mb/s]	[J/Mb]
SW-only	12	64	0.9085	0.9406	3.23301	0.290936
k-split 4x16b	12	64	0.8884	1.00445	11.11158	0.090397
SWAR improvement				0.93643x	3.43691x	3.21844x

HW Implementation



www.daiteq.com