S2CBench : Synthesizable SystemC Benchmark Suite for High-Level Synthesis

Benjamin Carrion Schafer[1], Ansuhree Mahapatra[2]
The Hong Kong Polytechnic University
Department of Electronic and Information Engineering
b.carrionschafer@polyu.edu.hk[1], anushree.mahapatra@connect.polyu.hk[2]

Design Automation and Reconfigurable Computing LAB
DARC
@DARC_lab

The Hong Kong Polytechnic University
香港理工大學

Outline

- Motivation for a Synthesizable SystemC Benchmark Suite
- S2CBench overview
  - 12 synthesizable design
  - 1 non synthesizable (tests floating point and trigonometric functions)
- Benchmark composition overview
- Detail benchmark characteristics
  - Size
  - Complexity
  - Arithmetic operations
- How to download
- Conclusions

Motivation for S2CBench (I)

- HLS tools evaluation cycles is typically very long (multiple tools are evaluated using multiple designs)
  - Companies don't have the expertise in HLS
  - Companies don't have ANSI-C, C++ or SystemC models for their RTL designs in order to compare the QoR of the HLS tools
- C/C++ supported by most vendors include vendor specific constructs. E.g. data types, port declarations
- SystemC only true language supported by all major HLS vendors

Vendor	Tool Name	Supported Languages
Cadence (Forte)	Cynthesizer	SystemC
Cadence	C-to-Silicon	C, C++, SystemC
Calypto	CatapultC	C++, SystemC
NEC	CyberWorkBench	C, SystemC
Xilinx	Vivado HLS	C, C++, SystemC

Motivation for S2CBench (II)

- Dedicated HLS benchmarks available are based on ANSI-C, e.g. CHStone
- Typically Multimedia applications written in ANSI-C used, e.g. MiBench or MediaBench or need to create their own ones:
  - Need to be edited to be made synthesizable forstate of the art commercial HLS tools
  - Do not support fixed point data types
  - Do not test specific HLS features (are just a collection of C programs)

➔ SystemC Benchmark suite will enable the direct comparison of commercial HLS tools

S2CBench Overview

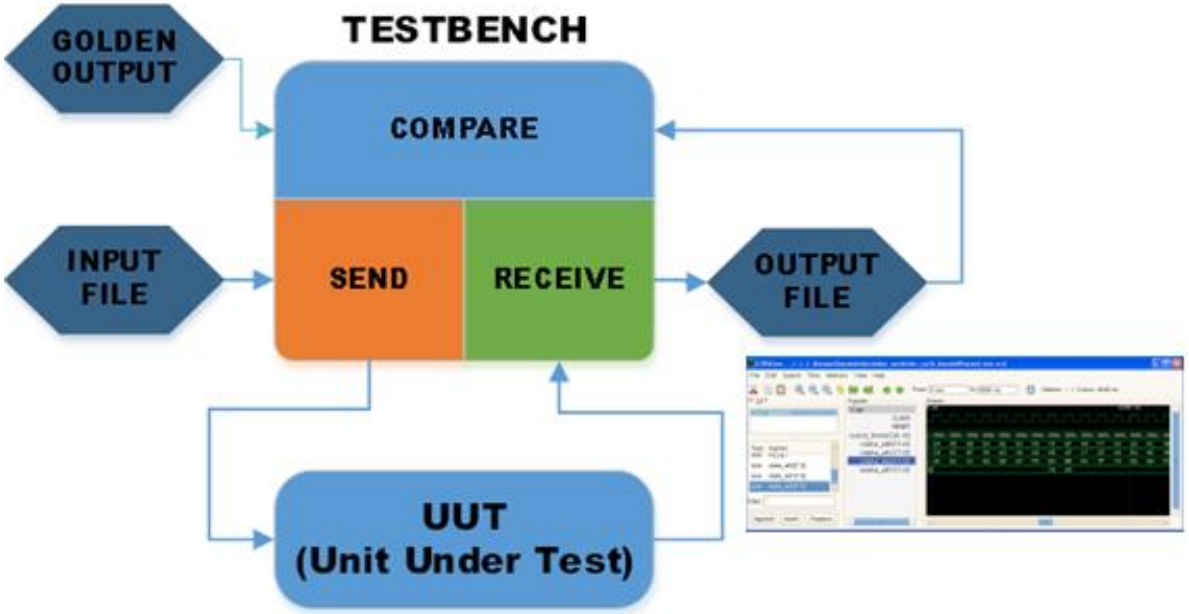
- 12+1 SystemC Benchmarks which comply with latest SystemC synthesizable subset draft (12 synthesizable+1 non synthesizable)
- From different domains
  - Automotive
  - Security
  - Telecommunication
  - Consumer
- Control dominated and Data dominant designs
- Each test unique HLS features
  1. Tool language support (e.g. templates, structures, fixed point data types)
  2. Synthesis optimizations (e.g. loop unrolling, pipelining, function inlining, array synthesis)
  3. Tool performance (e.g. running time, accuracy of synthesis report)

S2CBench : 12+1 designs

Design	Type	Domain	Optimizations Tested
qsort	dd	Auto/Ind	Loops, arrays, functions pointers
sobel	dd	Auto/Ind	Loops, functions, IO array expansion, multi-dimensional arrays expansion, fixed arrays (ROM, logic)
aes cipher	dd	Security	IO array expansion, multi-dimensional arrays expansion , large fixed arrays
kasumi	dd	Security	Multi-processes, delay report accuracy
md5c	dd	Security	#define macros, delay report accuracy
snow3G	dd	Security	Templates, delay report accuracy, function synthesis
adpcm	Cd	Telecom	Structure synthesis
FFT	dd	Telecom	Floating point, trigonometric functions
FIR	dd	Consu	IO array expansion, arrays, loops, functions, sum of products
Decimation	dd	Consum	Resource sharing across loops, fixed point data types
Interp	dd	Consum	Polynomial decomposition, fixed point data types, sum or products
IDCT	dd	Consum	#include statement to initialize arrays, loops, functions,
Disparity	cd/dd	Consum	Hierarchical design, multi-dimensional array expansion, synthesis running time

Benchmark Block Diagram

- TB sends stimuli data stored in files (editable) to UUT
- TB receives the data and compares it against golden output (stored in file)
- TB reports if results match or not
- Option to dump VCD file



Detail Benchmark Contents for each Design

- Makefile
  - Make : generates executable binary (default option)
  - Make wave : Generates binary which dumps a VCD file
  - Make debug: generates debug version (e.g. with gdb)
  - Make clean: cleans object file
- SystemC files
  - Main.cpp : top module includes UUT and TB
  - <benchmark>.cpp/.h : Main design description
  - tb_<benchmark>.cpp/.h : Testbench, sends receive and compares results against golden output
- Stimuli :
  - Inputs.txt : test vectors
  - Outputs_golden.txt : golden outputs
  - BMP : inputs for Sobel and disparity estimator

Quick – Quick Sort

- Description
  - sort design sorts data in ascending order using the well-known quick sort algorithm
- Main options to be tested
  - loop unrolling
  - array synthesis (register or memory)
  - function synthesis with pointer argument support

Sobel

- Description
  - edge-detection algorithm that takes a bitmap image directly as the input and returns a new bitmap image solely consisting of the edges of the original image.
- Main options to be tested
  - nested loop unrolling and pipelining optimizations
  - I/O ports expansion (expand inputs specified as arrays to individual ports)
  - multi-dimensional arrays expansion
  - fixed arrays synthesized as logic or ROMs
  - pointer arguments to functions

AES - Advanced Encryption Standard Cipher

- Description
  - Advanced Encryption Standard Cipher encryption algorithm performs AES encryption
- Main options to be tested
  - contains a large number of small *for* loops having inter-loop data dependencies.
  - input port expansion
  - array synthesis (memory or registers)
  - function synthesis (inline, goto)
  - large fixed arrays synthesized as logic or ROMs.

Kasumi

- Description
  - block cipher algorithm used in mobile communication systems
  - Composed of two sc_threads and multiple functions
- Main options to be tested
  - Contains large amount of logic operations (e.g. and, or, xor). HLS tools are notably not efficient, for accurately estimating the critical path of these applications, because the discrete delay of all the operations are simply added, thus overestimating the critical path
  - Multi-process systems verification

MD5C - Message Digest Algorithm

- Description
  - generates hash functions and check data integrity.
- Main options to be tested
  - functions synthesis
  - arrays of different bit widths
  - different levels of loop nesting
  - extensive use of *define* macros (language support)

Snow3G

- Description
  - stream cipher that produces a key stream that consists of 32-bit blocks using a 128-bit key
- Main options to be tested
  - Support of templates. A variable length multiplication operation is performed in this algorithm, which may be easily simplified using templates
  - Loops, functions and array synthesis

ADPCM -Adaptive Differential Pulse-Code modulation (encoder part only)

- Description
  - accepts 16-bit Pulse Code Modulation (PCM) samples as input and converts them into 4-bit samples
- Main options to be tested
  - loop unrolling, function synthesis, array synthesis
  - support for structures synthesis

FFT – Fast Fourier Transform (not synthesizable)

- Description
  - converts time/space to frequency and vice versa
- Main options to be tested
  - floating point operations and trigonometric functions
    → not synthesizable as per latest synthesis draft

➔Included because most commercial HLS provide tools to deal with floating points and trigonometric functions

➔Helps evaluation engineers understand how these operations are supported

FIR – Finite Impulse Response Filter

- Description
  - 10- tap FIR filter algorithm designed for 8- bit integer operations.
- Main options to be tested
  - loop unrolling and pipelining
  - automatic array expansion of the I/O ports
  - pointers to functions

Decimation Filter

- Description
  - 5-stage decimation filter. Consists of 5 FIR filters cascaded together where the output of one stage is the input to the next stage.
- Main options to be tested
  - resource sharing of the Multiply Accumulate (MAC) operations across loops
  - generated RTL is able to preserve the sum of product (SoP) construct → logic synthesis tool can optimize the construct further
  - fixed-point data types and its different rounding and saturation modes.

Interpolation Filter

- Description
  - 4-stage interpolation filter
- Main options to be tested
  - automatic polynomial decompositions. Significant area reduction can be obtained if polynomials can be decomposed into terms, so that the total number of arithmetic operations required is reduced → Mathematical optimization of HLS tool
  - fixed- point data types and its different rounding and saturation modes

IDCT - Inverse Discrete Cosine Transform

- Description
  - expresses a finite sequence of data points in terms of, a sum of cosine functions of different frequencies
- Main options to be tested
  - initialization of an array using *#include* statement
  - loops, functions, array synthesis

Disparity – Stereoscopic Disparity Estimator

- Description
  - estimates the disparity in a stereoscopic image.
  - It is the largest of all the designs and consists of 4 processes executed in parallel
- Main options to be tested
  - Almost all previously mentioned optimizations
  - Synthesis running time of the HLS tool (main thread contains a large number of loops leading to extreme long synthesis run times)
  - Verification of Multi-process (threads) systems

Detailed Benchmark Characteristics

- Size, complexity, arithmetic operations

Program Characteristics	qsort	sobel	aes cipher	kasumi	md5c	snow 3G	adpcm	fft	fir	decm	interp	idct	disparity
Lines of code	204	269	429	415	467	522	270	334	176	422	231	450	634
Processes	1	1	1	2	1	1	1	1	1	1	1	2	4
Function	1	2	11	5	7	11	3	1	2	1	1	2	4
Number of arrays	2	3	7	13	5	5	1	2	2	10	5	2	6
if statement	1	8	3	2	3	1	12	0	1	19	0	2	16
for statement	5	8	20	12	8	4	1	2	2	15	5	3	11
while statement	1	1	1	2	2	2	1	10	1	1	1	1	9
Test vector	.txt	.bmp	.txt	.txt	.txt	.txt	.txt	.txt	.txt	.txt	.txt	.txt	.bmp
Operations													
Addition/subtract	8	26	65	44	284	11	15	17	7	50	14	123	33
Multiplications	0	2	16	0	4	0	2	5	1	5	10	33	2
Divisions	0	0	3	0	0	0	0	2	0	0	0	0	1
Logic operations	7	0	22	39	274	67	9	0	5	4	2	33	13
Comparisons	0	17	29	22	16	10	16	10	0	43	8	36	42

Publicly Available

www.s2cbench.org

http://sourceforge.net/projects/s2cbench/

S2CBENCH

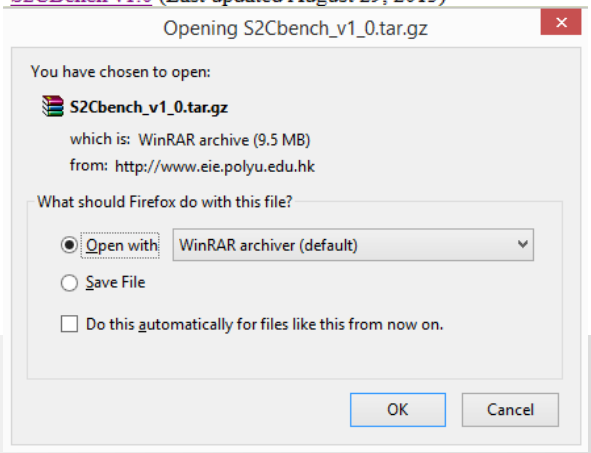
Synthesizable SystemC Benchmark Suite

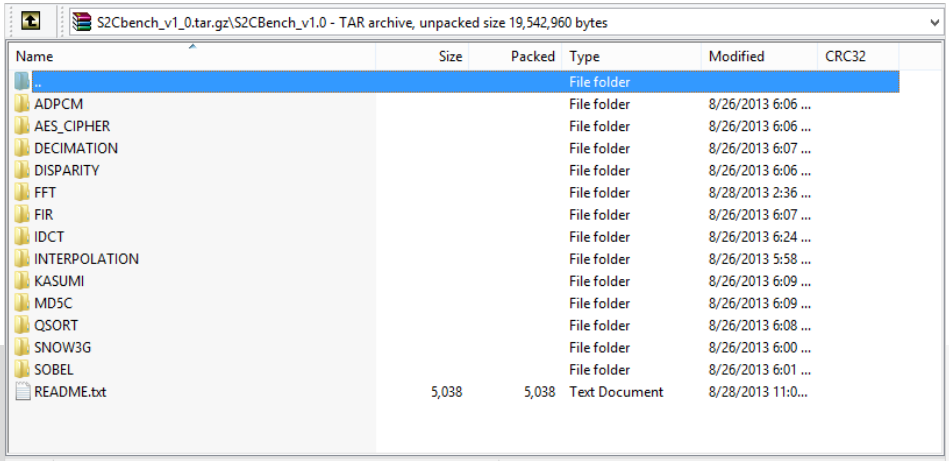
Home Download Contact

Download:

The S2CBench provides 12 programs written in synthesizable SystemC language. Each benchmark is designed for specific domains such as multimedia, digital signal processing, security, image processing, etc. The programs are provided with the objective to enable researchers analyze their innovative algorithms and techniques and help users compare the quality of results of state of the art commercial High Level Synthesis tools available in industry.

S2CBench v1.0 (Last updated August 29, 2013)





Summary and Conclusions

- A benchmark suite in a common language supported by all major HLS vendors
- Each benchmark tests unique HLS features
    1. Tool language support
    2. Synthesis optimizations
    3. Tool performance
- Benchmarks include testbench with inputs, golden outputs and option to generate VCD file
- Publicly available at www.s2cbench.org or sourceforge.net