

Space Codesign *Exploration, Design and Development of
Hardware & Software Multicore/MultiProcessor*

System-Level Exploration and Synthesis through Interoperability with HSL tools

Guy Bois

Space Codesign Systems
Polytechnique Montreal
guy.bois@spacecodesign.com

Space Codesign Systems
450 rue Saint-Pierre, Suite 1010
Montréal, Québec
H2Y 2M9 CANADA
<http://www.spacecodesign.com>

MPSOC 2015
July 14, 2015, Ventura

Agenda

**Space
Codesign**

- Why HLS?
- Limitation of HLS
- System Level Exploration and Synthesis
 - SpaceStudio + Vivado HLS
- HW/SW Refinement
- Example

- HLS is a viable alternative to HW design using HDL
- HLS produces better results than previous tools
- Time to market
 - Rapid exploration of HW solution space
- FPGAs have become large enough to justify the need of HLS
- Easier modifiability/maintainability

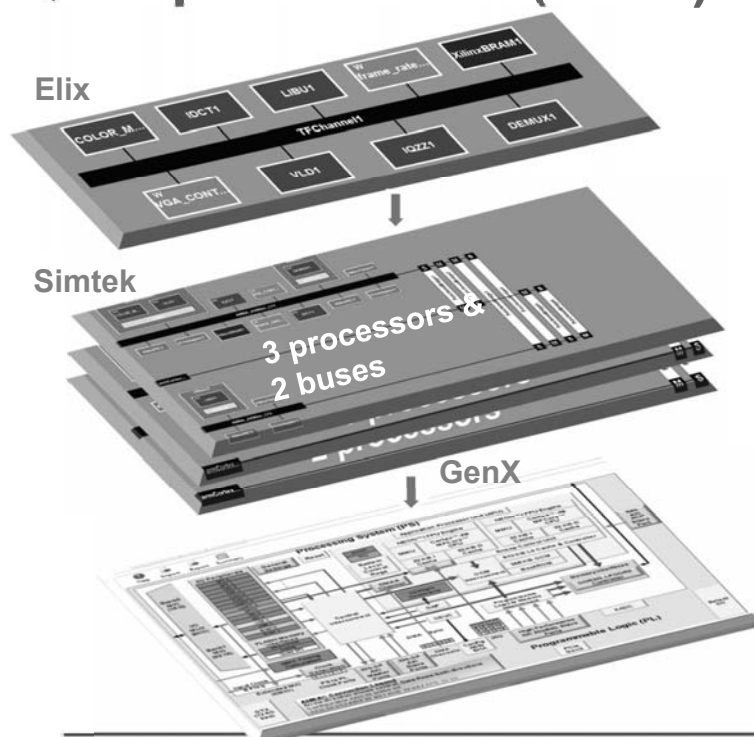
1. J. H. Anderson, Keynote at AHS 2015

- What about the SW interface (Firmware)?
- One must be able to:
 - Profile the application
 - Alter SW binary to call HW accelerators (e.g. C/C++ functions)
- Existing solutions:
 - SDSoC from Xilinx
 - LegUp from University of Toronto (legup.eecg.toronto.edu)

Limitation of HLS (2)

- What about the HW/SW solution space?
- One must be able to perform:
 - Functional/non-functional specification
 - System performance prediction
 - Non intrusive monitoring
 - HW/SW partitioning and (Micro)Architectural exploration
 - HW/SW codebuging (Visualization)
 - Loop acceleration (not only limited to C/C++ functions)
- Existing solution:
 - SpaceStudio from Space Codesign

SpaceStudio (Flow)



Functional validation

- Application Optimisation

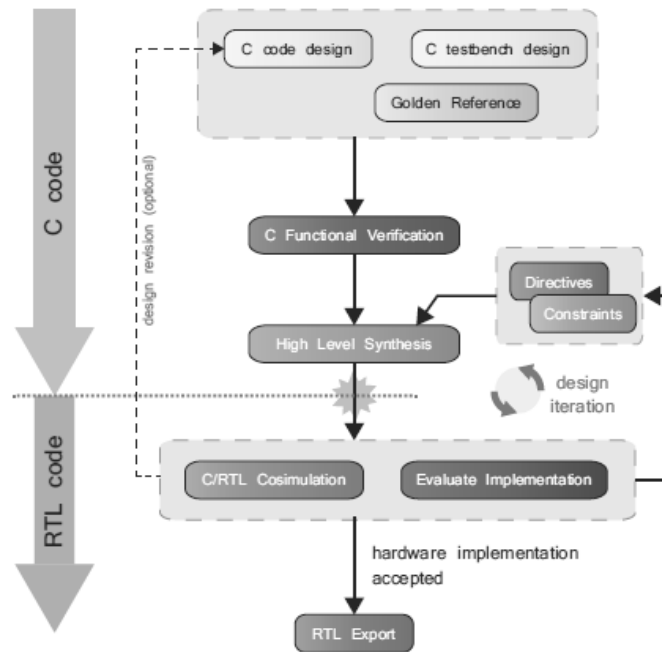
Architectural validation

- Exploration loop
- Architecture parameters evaluation:
 - Clocks frequency
 - FIFO sizes
 - Bus latency
 - Cache configuration
 - Etc.

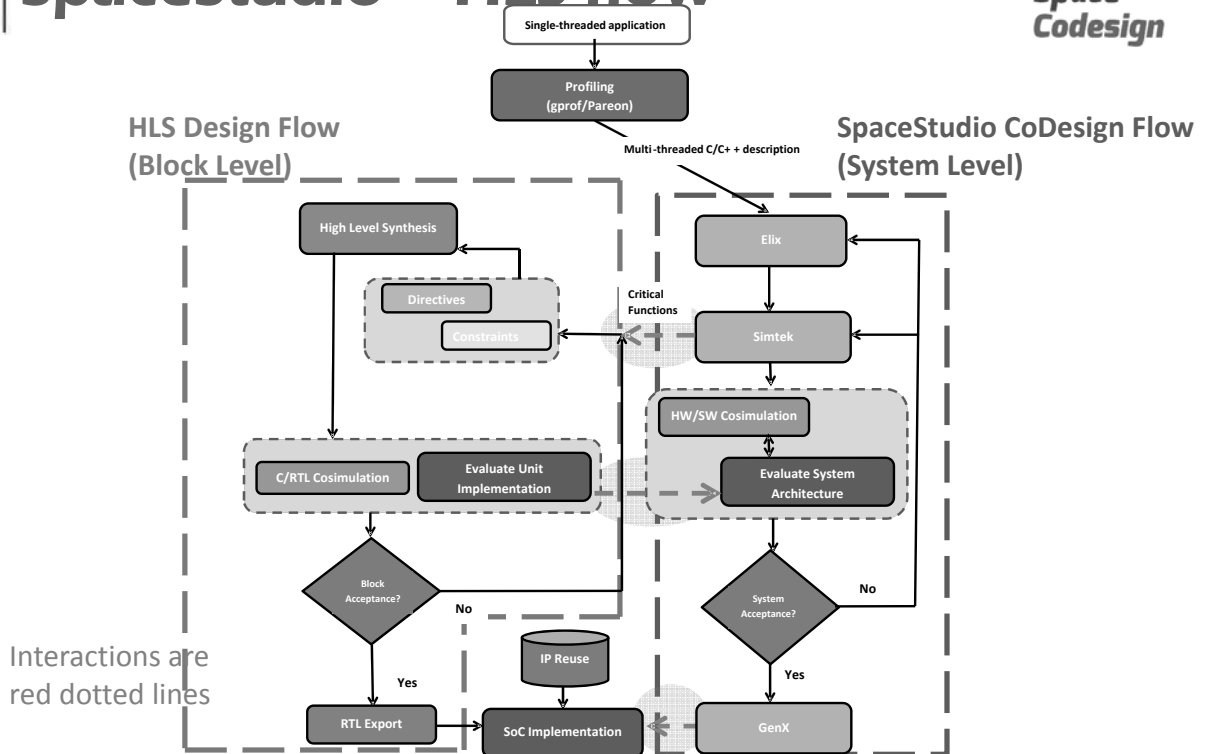
Implementation

- RTL project generation
 - Xilinx
 - Altera
 - Etc.


Vivado HLS Design Flow

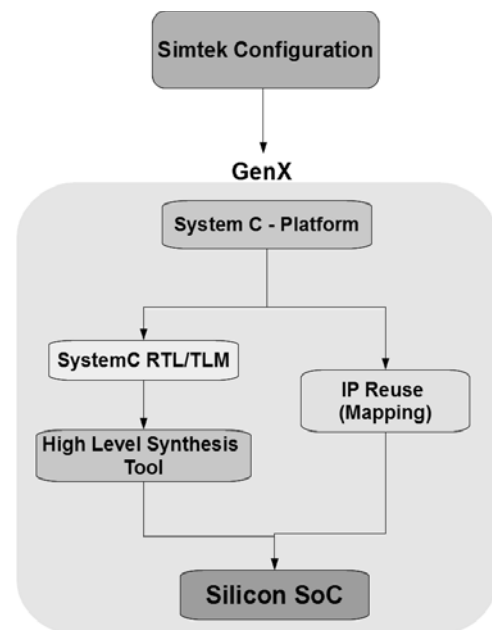


SpaceStudio + HLS flow



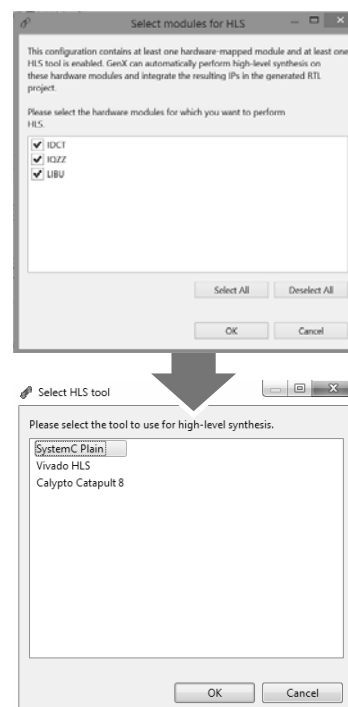
HW Synthesis

- Support for HW synthesis
 - HLS tools support
 - Automatic generation
- HW resource estimation
 - Retrieve HW estimates from HLS tool
 - Complete the Design Exploration Loop
- Refinement of Architectures (from Simtek) 
 - HW/SW Co-Synthesis



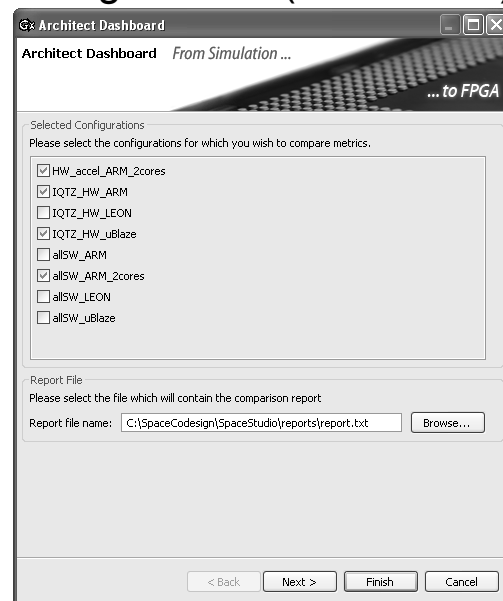
Behavioral synthesis (HLS)

- Support HLS tool
 - Vivado HLS
 - Calypto CatapultC
 - Cynthesizer
- Plain SystemC RTL
 - Explicit protocol
- Pragmas are sent to HLS
- Fine tuning in HLS tool



Architect's Dashboard

- Architect's Dashboard organizes simulation and analysis of multiple design architecture configurations (candidates)
- Select Candidates
- Simulate (if no recent results already exist)
- Comparison by criteria (e.g., Performance, HW resources)



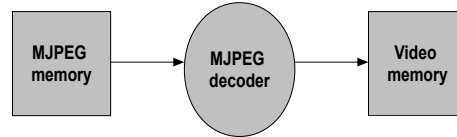
GenX

- Refinement of Selected Architecture
- Export RTL implementation of architecture
 - RTL hardware IPs and glue logic
 - Embedded firmware and software
 - Project and files for downstream tools
- Supported downstream technologies and tools
 - FPGA Platform Libraries
- Automated flow for application-specific accelerators
 - Support of HLS (high level synthesis) flows
 - RTL generation from C/C++ and SystemC

- Generated for each processor core
- Embedded software
 - SW-mapped modules as RTOS tasks
 - Communication drivers (kernel drivers)
 - Board support package (BSP)
 - RTOS configuration files
 - OS-specific (e.g., Embedded Linux – device trees)
 - Build files
- May also generate boot logic
 - Bootloader, boot ROM, etc.

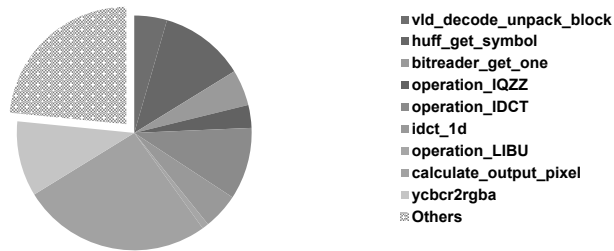
- MJPEG - SW acceleration with Hardware co-processor
- Sobel Filter:
 - See http://youtu.be/OKp_3jixZM

- SW acceleration with Hardware co-processor



- Step 1.1 Profiling the SW application

MJPEG functions execution time

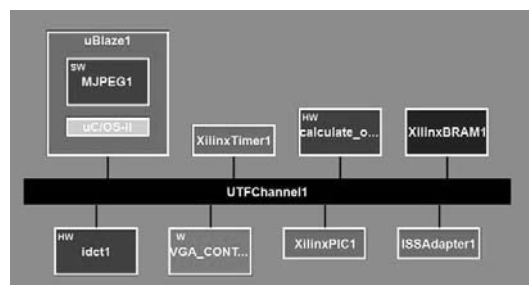


Function	Execution Time (%)
operation_LIBU	0.90%
operation_IQZZ	3.15%
vld_decode_unpack_block	4.50%
bitreader_get_one	4.95%
idct_1d	4.95%
operation_IDCT	9.91%
ycbcr2rgba	10.36%
huff_get_symbol	11.71%
calculate_output_pixel	26.13%
All other MJPEG decoder functions	26.42%

Two functions suggested as coprocessors

MJPEG functions execution time

- Step 1.2 From single-threaded to multithreaded



■ Step 1.2 From single-threaded to multithreaded

```

852     operation_IDCT(data);
853 }
854 }
855
856 //IDCT=====
857 void MJPEG::operation_IDCT(short dataInput[])
858 {
859     SPACE_ALIGNED unsigned char Idct[BLOCK_SIZE];
860
861     #ifdef IDCT_ID
862         // Send array
863         ModuleWrite(IDCT_ID,SPACE_BLOCKING, &dataInput);
864         // Receive array
865         ModuleRead(IDCT_ID,SPACE_BLOCKING, &Idct[0],BL
866     #else
867         int Y[BLOCK_SIZE];
868
869         for (int row = 0; row < BLOCK_HEIGHT; row++)
870         {
871             for (int column = 0; column < BLOCK_WIDTH;
872                 Y(row, column) = dataInput[row*BLOCK_W
873                 idct_id(&Y(row, 0));
874                 /* Result Y is scaled up by factor sqrt(8)
875         }
    
```

```

23 void idct::thread(void) {
24
25     idct_input_struct input;
26     int Y[BLOCK_SIZE];
27     SPACE_ALIGNED unsigned char Idct[BLOCK_SIZE];
28     short row, column;
29     computeFor();
30
31     while(1) {
32         ModuleRead(MJPEG_ID,SPACE_BLOCKING,&input);
33
34         A: for (row = 0; row < BLOCK_HEIGHT; row++)
35         {
36             B: for (column = 0; column < BLOCK_WIDTH; column++)
37                 Y(row, column) = input.dataInput[row*BLOCK_WIDTH+column] << 8_BITS;
38             IDCT_ID(&Y(row, 0));
39             /* Result Y is scaled up by factor sqrt(8)+2*8_BITS. */
40
41             C: for (column = 0; column < BLOCK_WIDTH; column++)
42             {
43                 int Yc[BLOCK_HEIGHT];
44                 D: for (row = 0; row < BLOCK_HEIGHT; row++)
45                     Yc[row] = Y(row, column);
46                 IDCT_ID(Yc);
47                 E: for (row = 0; row < BLOCK_HEIGHT; row++)
48                 {
49                     /* Result is once more scaled up by a factor sqrt(8). */
50                     int r = 128 + descale(Yc[row], 2*8_BITS);
51                     /* Clip to 8 bits unsigned */
52                     r = r >= 0 ? r < 255 ? r : 255 : 0;
53                     Idct_for(row, column) = r;
54                 }
55             }
56             ModuleWrite(MJPEG_ID,SPACE_BLOCKING, &Idct[0],BLOCK_SIZE);
57
58             //From HLS - HD optimization
59             computeFor(50);
60         }
61     }
    
```

■ Step 1.3 Architectural Exploration (1)

- Add the co-processor and compare the simulation time
- What is the speed-up?

	Execution time (seconds)	Performance (frame/sec)	Speed-up
MJPEG single thread	1.3559	1.4750	
MJPEG with IDCT HW	1.2081	1.6555	1.1223
MJPEG with calculate output pixel HW	0.4435	4.5096	3.0572
MJPEG with IDCT and calculate output pixel HW	0.2957	6.7636	4.5854

N.B. No optimization has been applied for HLS

- Step 1.3 Architectural Exploration (2)
 - How can we emulate the hardware co-processor execution's time?
 - Two possibilities:
 - 1. The hardware design team knows the IP latency
 - 2. Using a HLS tool to know the generated IP latency. Even if we don't use any pragma to optimize the synthesis, the result is useful as worst case.

Second possibility:

```

144 + A:
145 * Trip count: 8
146 * Latency: 240
147 + B:
148 * Trip count: 8
149 * Latency: 16
150 + C:
151 * Trip count: 8
152 * Latency: 440
153 + D:
154 * Trip count: 8
155 * Latency: 16
156 + E:
157 * Trip count: 8
158 * Latency: 24
159 + Loop 1.4:
160 * Trip count: 16
161 * Latency: ?
162 + Loop 1.4.1:
163 * Trip count: ?
164 * Latency: ?
165
22 void idct::thread(void) {
23
24 idct_input_struct input;
25 int Y(BLOCK_SIZE);
26 SPACE_ALIGNED unsigned char Idct(BLOCK_SIZE);
27 short row ,column;
28 computeFor();
29
30
31
32 while(1) {
33 ModuleRead(MJPEG_ID,SPACE_BLOCKING,4*input);
34
35 A: for (row = 0; row < BLOCK_HEIGHT; row++)
36 {
37 B: for (column = 0; column < BLOCK_WIDTH; column++)
38 Y(row, column) = input.data[input[row*BLOCK_WIDTH+column] << 2_BITS];
39 idct_idct(row, 0);
40 /* Result Y is scaled up by factor sqrt(8)*2^8_BITS. */
41 }
42
43 C: for (column = 0; column < BLOCK_WIDTH; column++)
44 {
45 int Ys[BLOCK_HEIGHT];
46
47 D: for (row = 0; row < BLOCK_HEIGHT; row++)
48 Ys[row] = Y(row, column);
49
50 idcm_id(Ys);
51
52 E: for (row = 0; row < BLOCK_HEIGHT; row++)
53 {
54 /* Result is once more scaled up by a factor sqrt(8). */
55 int z = 128 + descale(Ys[row], 7*8_BITS);
56 /* Clip to 8 bits unsigned. */
57 z = z > 0 ? (z < 255 ? z : 255) : 0;
58 Idct_fast(row, column) = z;
59 }
60
61
62 ModuleWrite(MJPEG_ID,SPACE_BLOCKING, 4*idct(),BLOCK_SIZE);
63
64
65 //From HLS - NO optimization
66 computeF(440);
67
68
69 }
70
    
```

Automatic annotation

- Step 1.3 Architectural Exploration (3)
 - What is the area estimation?

Area Estimates **IDCT**

Summary:
(Target device: xc7k325tffg900-2)

ID	Name	BRAM_18K	DSP48E	FF	LUT	SLICE
0	Component	-	56	0	0	-
1	Expression	-	-	0	1738	-
2	FIFO	-	-	-	-	-
3	Memory	4	-	0	0	-
4	Multiplexer	-	-	-	1020	-
5	Register	-	-	1858	-	-
6	ShiftMemory	-	-	-	-	-
-	Total	4	56	1858	2758	0
-	Available	890	840	407600	203800	50950
-	Utilization (%)	-0	6	-0	1	0

Area Estimates **Calculate output pixel**

Summary:
(Target device: xc7k325tffg900-2)

ID	Name	BRAM_18K	DSP48E	FF	LUT	SLICE
0	Component	-	28	7073	7297	-
1	Expression	-	5	0	1166	-
2	FIFO	-	-	-	-	-
3	Memory	3	-	0	0	-
4	Multiplexer	-	-	-	3437	-
5	Register	-	-	3204	-	-
6	ShiftMemory	-	-	-	-	-
-	Total	3	33	10277	11900	0
-	Available	890	840	407600	203800	50950
-	Utilization (%)	-0	3	2	5	0

Coprocessors area estimation

FPGA Estimation Results

From Simulation ...

... to FPGA

User-Defined Components

Because the following hardware components are user-defined components, they could not be found in SpaceStudio's library of hardware components. Please enter estimates of hardware resource utilization for each of user-defined components.

If you have already entered estimates in a previous FPGA estimation, these estimates might have been saved by SpaceStudio. In that case, please verify and confirm those previous estimates.

Area Metrics

Compon...	FlipFlops	LUT Elem...	Block RA...	DSPs	Available ...	Global CL...	DCMs
VGA_CO...	0	0	0	0	0	0	0
calculate	10348	10277	3	33	0	0	0
idct_top	1858	2758	4	56	0	0	0

Area estimation of the complete system

FPGA Estimation Results

From Simulation ...

... to FPGA

Computer Aided Design (CAD) Tools

EDA: Xilinx - EDK 14.4

Board

Board: Kintex-7 KC705 Evaluation Platform
 Vendor: xilinx.com
 Family: kintex7
 Device: xc7k325t
 Package: ffg900
 Speed grade: -2

Platform Resource Usage

FlipFlops:	3.94 %	16 057 of 407 600 FlipFlops
LUT Elements:	9.25 %	18 859 of 203 800 LUT Elements
Block RAMs:	13.03 %	58 of 445 Block RAMs
DSPs:	10.95 %	92 of 840 DSPs
Available IOBs:	0.60 %	3 of 500 Available IOBs
Global Clock Buffers:	6.25 %	2 of 32 Global Clock Buffers
DCMs:	0.00 %	0 of 10 DCMs

- Step 1.3 Architectural Exploration (4)
 - Other questions that can be answered in minutes (rather than hours/days):

1. What would be the effect on the frame rate if we replace the uBlaze by a Cortex A9?
2. The parameter "read/write miss penalty" is currently set at 3 cycles for cache L1 and 20 cycles for cache L2. What will be the effect on the frame rate if this parameter is divided by 3 for cache L1 (1 cycle) and by 4 for cache L2 (5 cycles)? (*Keep modifications of previous requirements*)
3. The current size of all FIFOs used for communication is 64, but we must save hardware resources. What will be the impact on the performance if we decrease the size of all FIFOs to 16? (*Keep modifications of previous requirements*)

Etc.