



Center for Embedded Computer Systems  
University of California, Irvine

---

## **System-Level Modeling and Refinement of a Canny Edge Detector**

Xu Han ,Yasaman Samei and Rainer Dömer

Technical Report CECS-12-13  
November 7, 2012

Center for Embedded Computer Systems  
University of California, Irvine  
Irvine, CA 92697-3425, USA  
(949) 824-8059

{hanx, ysameisy, doemer}@uci.edu  
<http://www.cecs.uci.edu/>

---

# System-Level Modeling and Refinement of a Canny Edge Detector

Xu Han ,Yasaman Samei and Rainer Dömer

Technical Report CECS-12-13

November 7, 2012

Center for Embedded Computer Systems

University of California, Irvine

Irvine, CA 92697-3425, USA

(949) 824-8059

{hanx, ysameisy, doemer}@uci.edu

<http://www.cecs.uci.edu>

## Abstract

*Electronic System Level design methodology and supporting tools simplifies the design of complex embedded systems. In this paper, a case study with an application example of canny edge detector is presented. We recoded a C reference code of canny to an initial specification model in SpecC. In order to yield best design from automated system level design tools, we refined the specification model by exploiting data-level parallelism, pipelining and converting floating-point computation to fixed-point in the initial model. With the optimized specification model, we used System-on-Chip Environment (SCE) to explore the design space, find allocation and mapping scheme which can achieve real-time computing, and refinement the specification model to Transaction level Model.*

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                                   | <b>1</b>  |
| <b>2</b> | <b>Overview of Canny Edge Detector</b>                | <b>2</b>  |
| <b>3</b> | <b>System Level Modeling of Canny Edge Detector</b>   | <b>3</b>  |
| 3.1      | Initial Specification Model . . . . .                 | 3         |
| 3.2      | Model Refinements . . . . .                           | 3         |
| 3.2.1    | Parallelization of <i>gaussian_smooth</i> . . . . .   | 3         |
| 3.2.2    | Pipelining . . . . .                                  | 5         |
| 3.2.3    | Converting Floating-point to Fixed-point . . . . .    | 6         |
| <b>4</b> | <b>Model Refinements and Implementation using SCE</b> | <b>6</b>  |
| <b>5</b> | <b>Conclusion</b>                                     | <b>8</b>  |
|          | <b>References</b>                                     | <b>9</b>  |
| <b>A</b> | <b>Appendix</b>                                       | <b>10</b> |
| A.1      | Source Code of Canny Edge Detector in SpecC . . . . . | 10        |
| A.2      | Makefile for TLM generation in SCE . . . . .          | 30        |

## List of Figures

|    |   |   |
|----|---|---|
| 1  | Top-Down System-Level Design Flow . . . . .                                 | 2 |
| 2  | Canny Edge Detector Flowchart . . . . .                                     | 3 |
| 3  | Initial Specification Model . . . . .                                       | 4 |
| 4  | Canny Profile using SCE . . . . .   | 4 |
| 5  | Parallel <i>gaussian_smooth</i> . . . . .                                   | 5 |
| 6  | 5-stage Pipelined Canny Edge Detector [5] . . . . .                         | 5 |
| 7  | Canny Profile using SCE after <i>gaussian_smooth</i> parallelized . . . . . | 6 |
| 8  | PE Allocation . . . . .   | 6 |
| 9  | Behavior Mapping . . . . .  | 7 |
| 10 | Network Allocation . . . . .  | 8 |
| 11 | Network Connectivity . . . . .  | 8 |

## List of Tables

|   |                                    |   |
|---|------------------------------------|---|
| 1 | Timing of various Models . . . . . | 8 |
|---|------------------------------------|---|

# System-Level Modeling and Refinement of a Canny Edge Detector

Xu Han ,Yasaman Samei and Rainer Dömer

Center for Embedded Computer Systems  
University of California, Irvine  
Irvine, CA 92697-3425, USA

{hanx, ysameisy, doemer}@uci.edu  
<http://www.cecs.uci.edu>

## Abstract

*Electronic System Level design methodology and supporting tools simplifies the design of complex embedded systems. In this paper, a case study with an application example of canny edge detector is presented. We recoded a C reference code of canny to an initial specification model in SpecC. In order to yield best design from automated system level design tools, we refined the specification model by exploiting data-level parallelism, pipelining and converting floating-point computation to fixed-point in the initial model. With the optimized specification model, we used System-on-Chip Environment (SCE) to explore the design space, find allocation and mapping scheme which can achieve real-time computing, and refinement the specification model to Transaction level Model.*

## 1 Introduction

The growing market for embedded systems, typically portable electronic devices, demands products with better functionality and shorter time to market. The complexity of designing, debugging and verifying systems containing increasing number of HW/SW components presents great challenge to embedded design methodology. Electronic system level (ESL) design engages the problem using models of higher abstraction level. In ESL design, a system is firstly specified using a System Level Design Language (SLDL). The initial model is called a specification model. Then supporting tools can efficiently perform design space exploration, high-level synthesis, and software refinement to cycle-accurate level on the specification model.

Figure 1 [3] presents an ESL design methodology and tool sets SCE [2] using SpecC SLDL. Firstly, the product specification is captured using SpecC. Then, architecture refinement is performed, which involves allocation of processing elements (PE), mapping behaviors, channels and

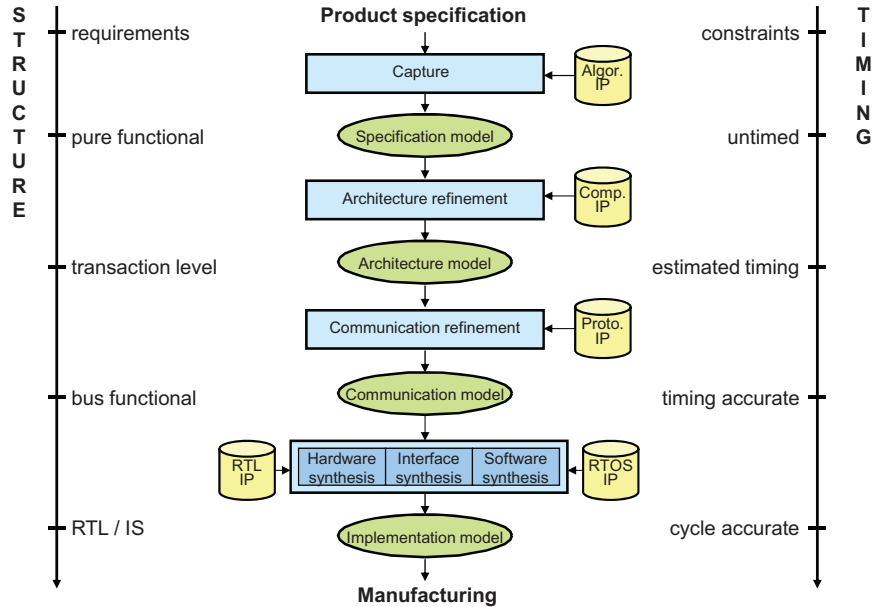


Figure 1: Top-Down System-Level Design Flow

variables to PEs. The result of architecture refinement is a system architecture model of concurrent PEs with abstract communication in channels. Next, scheduling refinement is performed to serialize the execution of behaviors on each PE according to either static or dynamic scheduling (RTOS support) chosen by designer. The result of this step is a system model with abstract RTOS inserted in each PE. Next, communication refinement implements the abstract communication channels between PEs. By allocating system busses and map the channels to them in SCE, we can then generate a bus-functional model of the system. Finally, the SW and HW components of bus-functional model is synthesized.

This report focuses on a case study of the ESL design methodology using an application example of canny edge detector (canny). Canny is modeled in SpecC, optimized for design and refined to pin-accurate level using SCE.

## 2 Overview of Canny Edge Detector

The canny edge detector is developed by Prof. John F. Canny in 1986 and our work is based on a reference implementation [1]. The algorithm applies five functions in sequential to an input image and detects all the edges in it (Figure 2).

The five functions are:

- **gaussian\_smooth** creates a gaussian kernel based on input parameter *SIGMA* (the standard deviation of the gaussian smoothing filter), and then used the kernel to filter or blur each pixel of the image to reduce the noise. The blurring occurs first horizontally and then vertically.

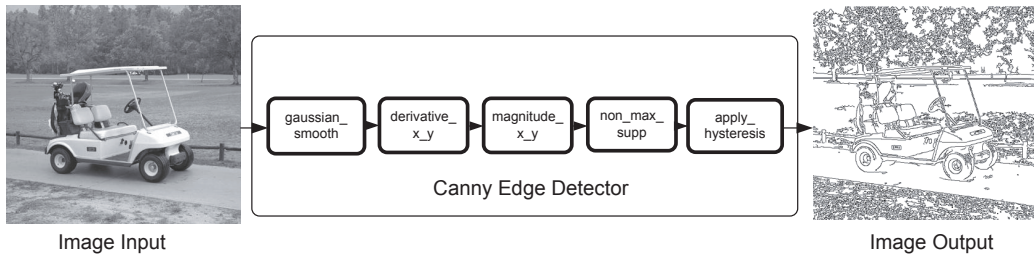


Figure 2: Canny Edge Detector Flowchart

- **derivative\_x\_y** computes the first derivative of the image in both the x any y directions.
- **magnitude\_x\_y** computes the magnitude of the gradient - the square root of the sum of the squared derivative values.
- **non\_max\_supp** applies non-maximal suppression to the magnitude of the gradient image. The pixels which are not part of local maxima are set as non-edges.
- **apply\_hysteresis** finds edges that are above some high threshold or are connected to a high pixel by a path of pixels greater than a low threshold. Parameter *TLOW* and *THIGH* specifies these two thresholds.

## 3 System Level Modeling of Canny Edge Detector

### 3.1 Initial Specification Model

The first step of ESL design on canny is to create a specification model. We recoded the unstructured and sequential C reference code into SpecC model by works including encapsulating functions into behaviors, creating channels and hierarchy, and creating a testbench. The resulting specification model is described as Figure 3(a) where *stimulus* sends incoming images to *platform*, I/O units(*din* and *dout*) send input to DUT *canny* and send output to *monitor*. DUT *canny* consists of 5 behaviors (Figure 3(b)) performing the 5 functions of canny algorithm.

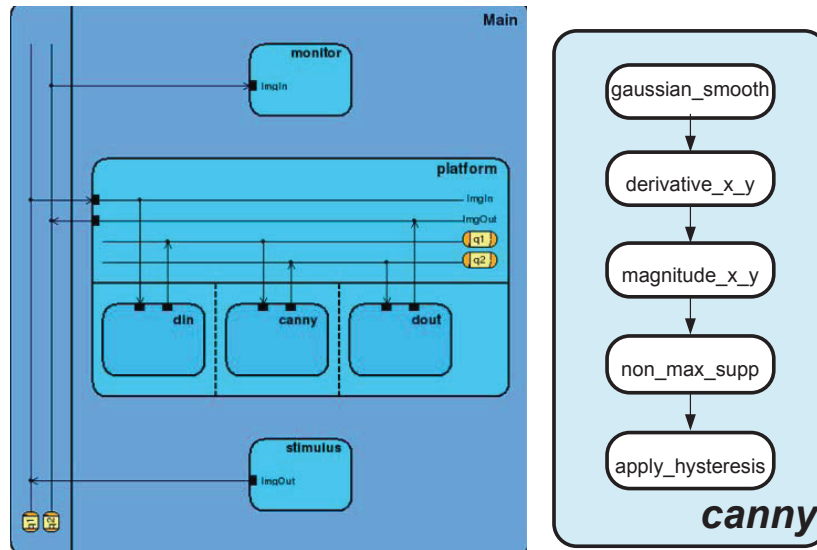
### 3.2 Model Refinements

We profiled the initial model using SCE profiler. The results (Figure 4) show that when ARM7 CPU is allocated, Behavior takes more than 50% of total computation. In order to yield better design, we considered to optimize *gaussian\_smooth*.

#### 3.2.1 Parallelization of *gaussian\_smooth*

Parallelization is one desired approach to optimize the heaviest function *gaussian\_smooth*. We observed that *gaussian\_smooth* firstly creates a Gaussian kernel used to blur the image (we call this





(a) Stimulus, Platform and Monitor (b) DUT: canny

Figure 3: Initial Specification Model

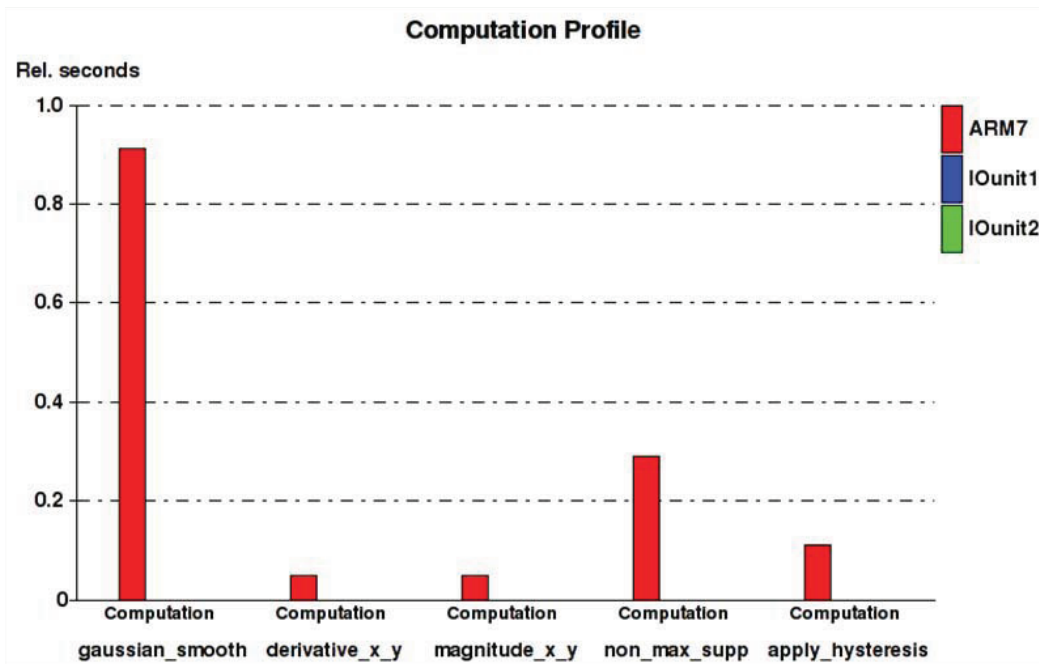


Figure 4: Canny Profile using SCE

task 'Prep'), then blurs the image in horizontally by filtering each pixel using its neighbors in X-direction (BlurX), and finally blurs the image in vertically by filtering each pixel using its neighbors

in Y-direction (BlurY). One parallelization strategy is to run BlurX on horizontal slices of the image in parallel and BlurY on vertical slices. In this way, the parallelization can be dependency free.

To model parallel tasks, we partitioned *gaussian\_smooth* into 3 behaviors, namely *Prep*, *BlurX* and *BlurY*. Assuming 4 available processing elements, we created the new hierarchy for *gaussian\_smooth* as Figure 3.2.1.

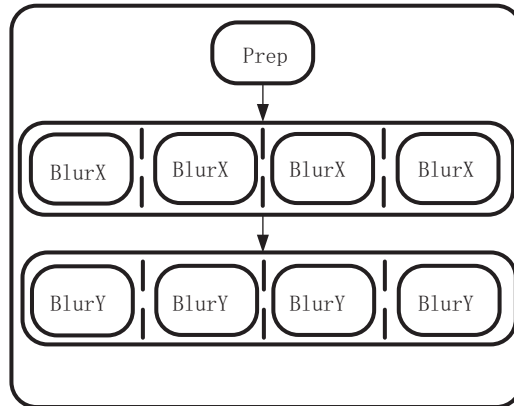


Figure 5: Parallel *gaussian\_smooth*

### 3.2.2 Pipelining

Though the reference C code only processes a single image, we adapted the specification model to process a sequence of images with the number of images specified by the user. We have pipelined the 5 functional blocks (Figure 6) so that the model is able to process multiple incoming images in parallel.

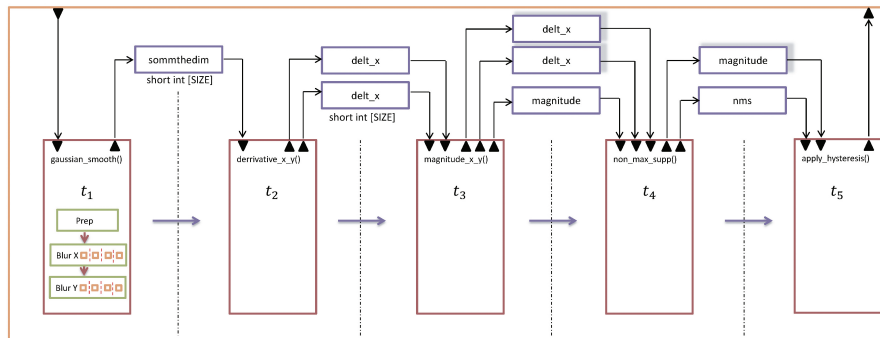


Figure 6: 5-stage Pipelined Canny Edge Detector [5]

### 3.2.3 Converting Floating-point to Fixed-point

Figure 7 shows updated profiling results when *gaussian\_smooth* is parallelized using 4 processing unit and tentatively mapped onto two hardware unit using SCE. Now *non\_max\_supp* contains the largest amount of computation in all behaviors and becomes the bottleneck in the pipeline.

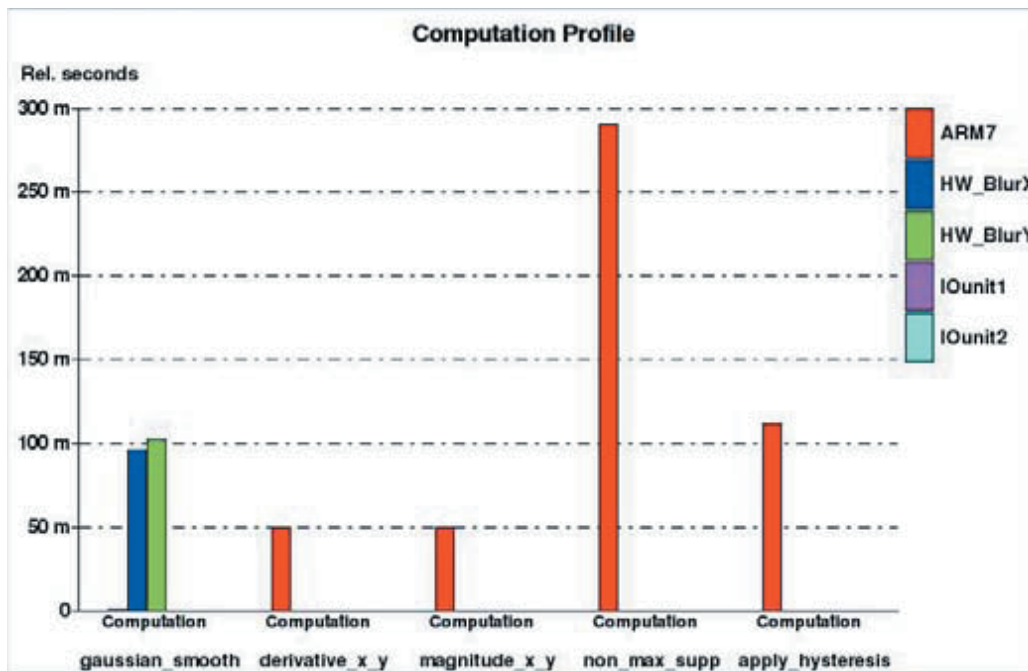


Figure 7: Canny Profile using SCE after *gaussian\_smooth* parallelized

From SCE profiling statistics, 95.6% of the operations in *non\_max\_supp* are floating-point. In order to balance the pipeline stages, we converted floating-point computation to fixed-point without loss of accuracy [4].

The source code of the refined specification model can be found in Appendix A.1.

## 4 Model Refinements and Implementation using SCE

| PE Allocation  |                         |
|----------------|-------------------------|
| Name ▾         | Type                    |
| CPU            | ARM_7TDMI_10000_20000_0 |
| Hardware_BlurX | HW_Standard             |
| Hardware_BlurY | HW_Standard             |
| IO_IN          | HW_Virtual              |
| IO_OUT         | HW_Virtual              |

Figure 8: PE Allocation

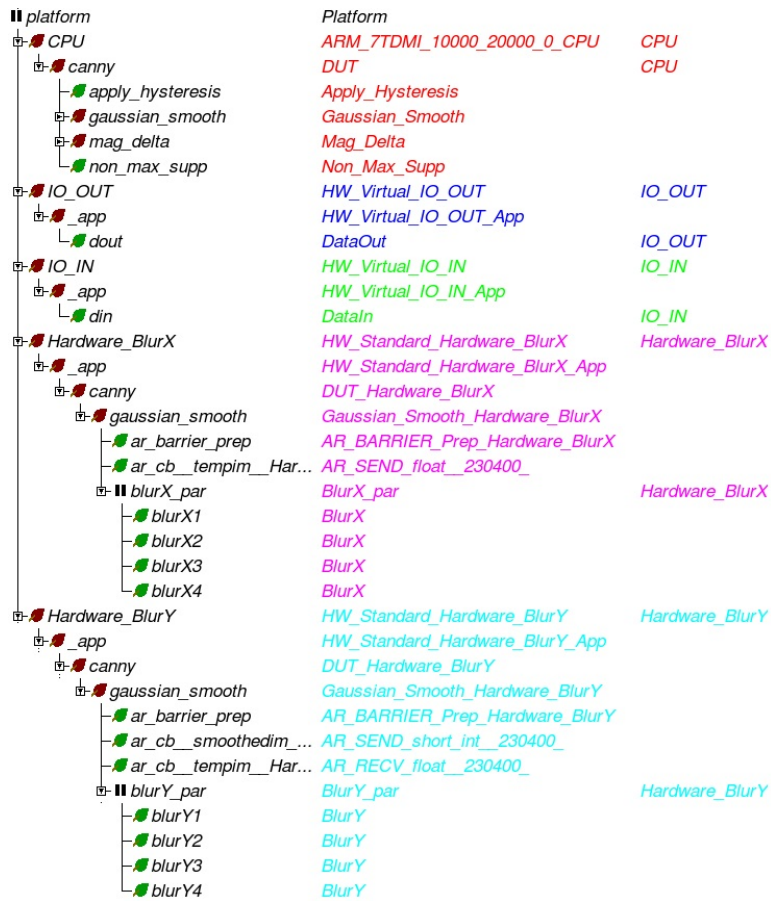


Figure 9: Behavior Mapping

In this section different steps of design space exploration shown in Figure 1, are performed over Canny example. The SCE tool is used to perform a step by step implementation. The refinement steps start with architecture refinement. In this step processing elements(PE) are added to design and canny's behaviors are allocated to them. The added PEs and their corresponding behaviors are shown in the Figure 8 and Figure 9.

The canny algorithms are running on a ARM7TDMI target processor and as it is shown in Figure 9 two sub behavior of the *gaussian\_smooth* function are allocated to separate hardware units. Since the *gaussian\_smooth* function is computationally expensive, extra hardware units are added to improve the performance of the design.

In the next step of system level design process different behaviors of the design are scheduled. In the canny example no dynamic scheduling algorithm is applied and behaviors execution order is static and same as the original specification of the design. After scheduling the behaviors, decisions on communication between system components are made. Two system BUSES are selected in canny example; AMBA BUS belong to allocated ARM CPU and a Hardware bus. The allocated BUSES

| Network Allocation |                     |      |     |      |       |          |       |              |  |
|--------------------|---------------------|------|-----|------|-------|----------|-------|--------------|--|
| Busses             |                     |      |     |      |       |          |       |              |  |
| CEs                |                     |      |     |      |       |          |       |              |  |
| Connectivity       |                     |      |     |      |       |          |       |              |  |
| Name               | Type                | Link | Mem | Intr | Queue | Clock    | Duty  | Speed        |  |
| CPU_Bus            | AMBA_AHB_20000_0_32 | ✓    | ✓   | ✓    |       | 20000 ps | 0.5 % | 800.0 Mbit/s |  |
| HW_Bus             | HardwareBus_16_32_1 | ✓    |     | ✓    |       | 0 ps     | 0.5 % | 10.0 Mbit/s  |  |

Figure 10: Network Allocation

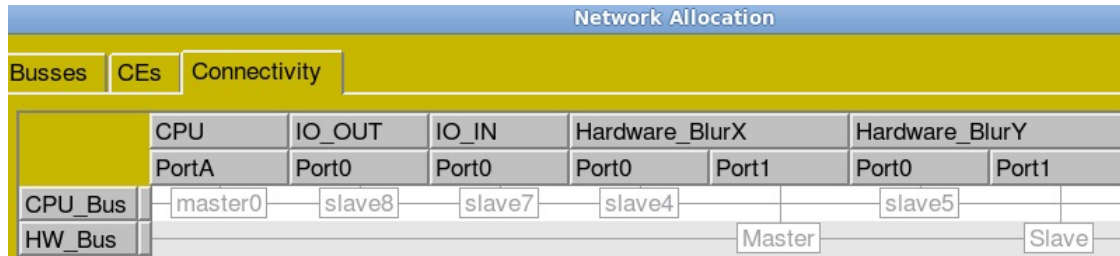


Figure 11: Network Connectivity

are illustrated in Figure 10. Following the BUS allocation, components connectivity, Masters and Slaves are defined. In Figure 11 component's connections over the BUS are clarified.

Finally the implementation is brought down to Transaction level Model(TLM). Table 1 reflects the simulation results in each step. The detailed configuration of the TLM generation is described in the Makefile in Appendix .

Table 1: Timing of various Models

| Model                        | Simulation Time(micro second) |
|------------------------------|-------------------------------|
| Specification Model          | 0                             |
| Architecture Model           | 1982972                       |
| Scheduled Architecture Model | 1982972                       |
| Network Model                | 1982972                       |
| TLM Model                    | 2012235                       |

## 5 Conclusion

In this report, we have created a system-level model of a canny edge detector based on its C reference code. We refined the model by exploiting parallelism, pipelining and converting floating-point to fixed-point for the bottleneck pipelining stage. We then use SCE to find an allocation and mapping scheme and refine the model automatically to Transaction level Model. The model is verified by simulation.

## References

- [1] Canny Edge Detector. [ftp://figment.csee.usf.edu/pub/Edge\\_Comparison/source\\_code/canny.src](ftp://figment.csee.usf.edu/pub/Edge_Comparison/source_code/canny.src).
- [2] Rainer Dömer, Andreas Gerstlauer, Junyu Peng, Dongwan Shin, Lukai Cai, Haobo Yu, Samar Abdi, and Daniel Gajski. System-on-Chip Environment: A SpecC-based Framework for Heterogeneous MPSoC Design. *EURASIP Journal on Embedded Systems*, 2008(647953):13 pages, 2008.
- [3] Andreas Gerstlauer, Rainer Dömer, Junyu Peng, and Daniel D. Gajski. *System Design: A Practical Guide with SpecC*. Kluwer, 2001.
- [4] Jiang Wan. Modeling of a canny edge detector system-on-chip for a digital camera. EECS222A 2012 Spring Course Project Report.
- [5] Ching-Yao Wang. Modeling of a canny edge detector system-on-chip for a digital camera. EECS222A 2012 Spring Course Project Report.

## A Appendix

### A.1 Source Code of Canny Edge Detector in SpecC

```
1 /* Canny Edge Detector for image stream
2    X.Han, Rainer Doemer, Oct.2012 */
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <string.h>
8
9 #include <sim.sh>
10 #include <c_typed_queue.sh>    /* make the templates available */
11 #include <c_typed_double_handshake.sh>
12
13 #define VERBOSE 0
14 #define TIME_BASE      (1 MICRO_SEC)    // print time in units of micro-seconds
15
16 #define NOEDGE 255
17 #define POSSIBLE_EDGE 128
18 #define EDGE 0
19 #define BOOSTBLURFACTOR 90.0
20
21 #define COLS 640
22 #define ROWS 360
23 #define SIZE COLS*ROWS
24
25 //do not include picture number and ".pgm" of the filename
26 #define FILENAME "image"
27
28 #define SIGMA 0.6
29 #define TLOW 0.3
30 #define THIGH 0.8
31
32 #define IMGNUM 4 //it can be any number
33 #define AVAIL_IMG 3
34 #define SHIFT_BIT 16
35
36 /* an upper bound for removing dynamic calloc
37 * SIGMA must be less than 4
38 * check for 'window size' below
39 */
40 #define WINSIZE 21
41
42 // patch for the non_max_supp function
43 #define SMP_PATCH
44
45 // fixed point computing for non_max_supp() function
46 #define FIXED_POINT
47
48 typedef unsigned char img[SIZE];    /* define our communication data types */
```

```

49 typedef short imgs[SIZE];
50
51 DEFINE_I_TYPED_SENDER(img, img)           // creates interface i_img_sender
52 DEFINE_I_TYPED_RECEIVER(img, img)        // creates interface i_img_receiver
53 DEFINE_I_TYPED_TRANCEIVER(img, img)      // creates interface i_img_tranceiver
54
55 DEFINE_C_TYPED_QUEUE(img, img)           // creates channel c_img_queue
56
57
58 behavior Monitor(i_img_receiver ImgIn, in sim_time StartTime[IMGNUM])
59 {
60     unsigned char edge[SIZE];
61
62     /******
63     * Function: write_pgm_image
64     * Purpose: This function writes an image in PGM format. The file is either
65     * written to the file specified by outfile_name or to standard output if
66     * outfile_name = NULL. A comment can be written to the header if coment != NULL.
67     *****/
68     int write_pgm_image(char *outfile_name, unsigned char *image, int rows,
69                       int cols, const char *comment, int maxval)
70     {
71         FILE *fp;
72
73         /******
74         * Open the output image file for writing if a filename was given. If no
75         * filename was provided, set fp to write to standard output.
76         *****/
77         if(outfile_name == NULL) fp = stdout;
78         else{
79             if((fp = fopen(outfile_name, "w")) == NULL){
80                 fprintf(stderr, "Error_writing_the_file_%s_in_write_pgm_image().\n",
81                          outfile_name);
82                 return(0);
83             }
84         }
85
86         /******
87         * Write the header information to the PGM file.
88         *****/
89         fprintf(fp, "P5\n%d\n", cols, rows);
90         if(comment != NULL)
91             if(strlen(comment) <= 70) fprintf(fp, "#%s\n", comment);
92         fprintf(fp, "%d\n", maxval);
93
94         /******
95         * Write the image data to the file.
96         *****/
97         if((unsigned)rows != fwrite(image, cols, rows, fp)){
98             fprintf(stderr, "Error_writing_the_image_data_in_write_pgm_image().\n");
99             if(fp != stdout) fclose(fp);
100            return(0);

```



```

101     }
102
103     if(fp != stdout) fclose(fp);
104     return(1);
105 }
106
107 void main()
108 {
109     char outfilename[128];    /* Name of the output "edge" image */
110     sim_time t, t2;
111     sim_time_string buf, buf2;
112     int i,n;
113
114     for(i=0; i<IMGNUM;i++)
115     {
116         ImgIn.receive(&edge);
117
118         t = now();
119         printf("%8s: \u2013Monitor\u2013received\u2013image%d.\n", time2str(buf, t/TIME_BASE), i);
120         t2 = t - StartTime[i];
121         printf("%8s: \u2013Image\u2013processing\u2013took\u2013%8s\u2013micro\u2013seconds.\n",
122             time2str(buf, t/TIME_BASE), time2str(buf2, t2/TIME_BASE));
123
124
125         /******
126         * Write out the edge image to a file.
127         *****/
128
129         n=i%AVAIL_IMG;
130
131         sprintf(outfilename, "%s_s_%3.2f_l_%3.2f_h_%3.2f%d.pgm", FILENAME, SIGMA, TLOW, THIGH, i);
132         if(VERBOSE) printf("Writing the edge image in the file %s.\n", outfilename);
133         if(write_pgm_image(outfilename, edge, ROWS, COLS, "", 255) == 0){
134             fprintf(stderr, "Error writing the edge image, %s.\n", outfilename);
135             exit(1);
136         }
137     } //for
138
139
140     exit(0);    // done testing, quit the simulation
141 }
142 };
143
144 behavior Stimulus(i_img_sender ImgOut, out sim_time StartTime[IMGNUM])
145 {
146     unsigned char image[SIZE];
147
148     /******
149     * Function: read_pgm_image
150     * Purpose: This function reads in an image in PGM format. The image can be
151     * read in from either a file or from standard input. The image is only read
152     * from standard input when infilename = NULL. Because the PGM format includes

```

```

153  * the number of columns and the number of rows in the image, these are read
154  * from the file. Memory to store the image is allocated OUTSIDE this function.
155  * The found image size is checked against the expected rows and cols.
156  * All comments in the header are discarded in the process of reading the
157  * image. Upon failure, this function returns 0, upon success it returns 1.
158  *****/
159  int read_pgm_image(const char *infilename, unsigned char *image0, int rows, int cols)
160  {
161      FILE *fp;
162      char buf[71];
163      int r, c;
164
165      /******
166      * Open the input image file for reading if a filename was given. If no
167      * filename was provided, set fp to read from standard input.
168      *****/
169      if(infilename == NULL) fp = stdin;
170      else{
171          if((fp = fopen(infilename, "r")) == NULL){
172              fprintf(stderr, "Error reading the file %s in read_pgm_image().\n",
173                  infilename);
174              return(0);
175          }
176      }
177
178      /******
179      * Verify that the image is in PGM format, read in the number of columns
180      * and rows in the image and scan past all of the header information.
181      *****/
182      fgets(buf, 70, fp);
183      if(strncmp(buf, "P5", 2) != 0){
184          fprintf(stderr, "The file %s is not in PGM format in ", infilename);
185          fprintf(stderr, "read_pgm_image().\n");
186          if(fp != stdin) fclose(fp);
187          return(0);
188      }
189      do{ fgets(buf, 70, fp); }while(buf[0] == '#'); /* skip all comment lines */
190      sscanf(buf, "%d%d", &c, &r);
191      if(c != cols || r != rows){
192          fprintf(stderr, "The file %s is not a %d by %d image in ", infilename, cols, rows);
193          fprintf(stderr, "read_pgm_image().\n");
194          if(fp != stdin) fclose(fp);
195          return(0);
196      }
197      do{ fgets(buf, 70, fp); }while(buf[0] == '#'); /* skip all comment lines */
198
199      if((unsigned)rows != fread(image0, cols, rows, fp)){
200          fprintf(stderr, "Error reading the image data in read_pgm_image().\n");
201          if(fp != stdin) fclose(fp);
202          return(0);
203      }
204

```

```

205     if(fp != stdin) fclose(fp);
206     return(1);
207 }
208
209 void main()
210 {
211     sim_time t;
212     sim_time_string buf;
213     int i=0,n=0;
214     char infilename[40];
215
216
217
218
219     for(i=0;i < IMGNUM;i++)
220     {
221         n=i%AVAIL_IMG;
222
223         sprintf(infilename , "%s%d.pgm",FILENAME,n);
224
225         if(VERBOSE) printf("Reading the image %s.\n", infilename);
226         if(read_pgm_image(infilename , image, ROWS, COLS) == 0){
227             fprintf(stderr , "Error reading the input image, %s.\n", infilename);
228             exit(1);
229         }
230         t = now() / TIME_BASE;
231         printf("%8s: Stimulus sends image%d.\n", time2str(buf, t), i);
232         StartTime[i] = t;
233
234         ImgOut.send(image);
235
236     }
237
238
239 }
240 };
241
242
243 behavior Prep(i_img_receiver ImgIn, inout int center,
244              inout float kernel[WINSIZE], inout img image)
245 {
246     int window_size;          /* Dimension of the gaussian kernel. */
247
248     void make_gaussian_kernel(float sigma)
249     {
250         int i;
251         float x, fx, sum=0.0;
252
253         window_size = 1 + 2 * ceil(2.5 * sigma);
254         center = window_size / 2;
255
256         if(VERBOSE) printf("The kernel has %d elements.\n", window_size);

```

```

257
258     for(i=0;i<windowSize;i++){
259         x = (float)(i - center);
260         fx = pow(2.71828, -0.5*x*x/(sigma*sigma)) / (sigma * sqrt(6.2831853));
261         kernel[i] = fx;
262         sum += fx;
263     }
264
265     for(i=0;i<windowSize;i++) kernel[i] /= sum;
266
267     if(VERBOSE){
268         printf("The filter coefficients are:\n");
269         for(i=0;i<windowSize;i++)
270             printf("kernel[%d] = %f\n", i, kernel[i]);
271     }
272 }
273
274 void main()
275 {
276     ImgIn.receive(&image);
277
278     /******
279     * Create a 1-dimensional gaussian smoothing kernel.
280     *
281     *
282     *
283     *
284     */
285     if(VERBOSE) printf("Computing the gaussian smoothing kernel.\n");
286     make_gaussian_kernel(SIGMA);
287 }
288 };
289
290 behavior BlurX(in img image, in int center, in float kernel[WINSIZE],
291               inout float tempim[SIZE], in int rowStart, in int rowEnd)
292 {
293     void main()
294     {
295         int r, c, cc;    /* Counter variables. */
296         float dot,      /* Dot product summing variable. */
297             sum;        /* Sum of the kernel weights variable. */
298
299         /******
300         * Blur in the x - direction.
301         *
302         *
303         *
304         *
305         */
306         if(VERBOSE) printf("Blurring the image in the X-direction.\n");
307         for(r=rowStart;r<rowEnd;r++){
308             for(c=0;c<COLS;c++){
309                 dot = 0.0;
310                 sum = 0.0;
311                 for(cc=(-center);cc<=center;cc++){
312                     if((c+cc) >= 0 && ((c+cc) < COLS)){
313                         dot += (float)image[r*COLS+(c+cc)] * kernel[center+cc];
314                         sum += kernel[center+cc];

```

```

309     }
310   }
311   tempim[r*COLS+c] = dot/sum;
312 }
313 }
314 }
315 };
316
317
318 behavior BlurY(out imgs smoothedim, in int center, in float kernel[WINSIZE],
319               in float tempim[SIZE], in int colStart, in int colEnd)
320 {
321
322 void main()
323 {
324   int r, c, rr;           /* Counter variables. */
325   float dot,             /* Dot product summing variable. */
326         sum;             /* Sum of the kernel weights variable. */
327
328   /******
329   * Blur in the y - direction.
330   * *****/
331   if(VERBOSE) printf("Blurring the image in the Y-direction.\n");
332   for(c=colStart;c<colEnd;c++){
333     for(r=0;r<ROWS;r++){
334       sum = 0.0;
335       dot = 0.0;
336       for(rr=(-center);rr<=center;rr++){
337         if((r+rr) >= 0 && ((r+rr) < ROWS)){
338           dot += tempim[(r+rr)*COLS+c] * kernel[center+rr];
339           sum += kernel[center+rr];
340         }
341       }
342       smoothedim[r*COLS+c] = (short int)(dot*BOOSTBLURFACTOR/sum + 0.5);
343     }
344   }
345 }
346 };
347
348
349 behavior BlurX_par(in img image, in int center, in float kernel[WINSIZE],
350                  inout float tempim[SIZE])
351 {
352   BlurX blurX1(image, center, kernel, tempim, ((ROWS/4)*0), ((ROWS/4)*1));
353   BlurX blurX2(image, center, kernel, tempim, ((ROWS/4)*1), ((ROWS/4)*2));
354   BlurX blurX3(image, center, kernel, tempim, ((ROWS/4)*2), ((ROWS/4)*3));
355   BlurX blurX4(image, center, kernel, tempim, ((ROWS/4)*3), ((ROWS/4)*4));
356
357   void main()
358   {
359     par{
360       blurX1.main();

```

```

361         blurX2.main();
362         blurX3.main();
363         blurX4.main();
364     }
365 }
366 };
367
368
369 behavior BlurY_par(out imgs smoothedim, in int center, in float kernel[WINSIZE],
370                 in float tempim[SIZE])
371 {
372     BlurY blurY1(smoothedim, center, kernel, tempim, ((COLS/4)*0), ((COLS/4)*1));
373     BlurY blurY2(smoothedim, center, kernel, tempim, ((COLS/4)*1), ((COLS/4)*2));
374     BlurY blurY3(smoothedim, center, kernel, tempim, ((COLS/4)*2), ((COLS/4)*3));
375     BlurY blurY4(smoothedim, center, kernel, tempim, ((COLS/4)*3), ((COLS/4)*4));
376
377     void main()
378     {
379         par{
380             blurY1.main();
381             blurY2.main();
382             blurY3.main();
383             blurY4.main();
384         }
385     }
386 };
387
388 behavior Blur_done( in imgs smoothedim, out imgs smoothedimg )
389 {
390     void main()
391     {
392         smoothedimg = smoothedim;
393     }
394 };
395
396 behavior Gaussian_Smooth (i_img_receiver ImgIn, out imgs smoothedimg)
397 {
398     img image;
399     int center;
400     float kernel[WINSIZE];
401     float tempim[SIZE];
402     imgs smoothedim;
403
404     Prep prep(ImgIn, center, kernel, image);
405     BlurX_par blurX_par(image, center, kernel, tempim);
406     BlurY_par blurY_par(smoothedim, center, kernel, tempim);
407     Blur_done blur_done(smoothedim, smoothedimg);
408
409     void main()
410     {
411         prep.main();
412         blurX_par.main();

```

```

413     blurY_par.main();
414     blur_done.main();
415 }
416 };
417
418
419 behavior Derivative_X-Y(in imgs smoothedim, out imgs delta_x, out imgs delta_y)
420 {
421     /******
422     * PROCEDURE: derrivative_x_y
423     * PURPOSE: Compute the first derivative of the image in both the x any y
424     * directions. The differential filters that are used are:
425     *
426     *            $dx = \begin{matrix} -1 & 0 & +1 \end{matrix}$     and     $dy = \begin{matrix} -1 \\ 0 \\ +1 \end{matrix}$ 
427     *
428     *
429     *
430     * NAME: Mike Heath
431     * DATE: 2/15/96
432     * *****/
433     void derivative_x_y(short int *smoothedimg, int rows, int cols,
434         short int *deltax, short int *deltay)
435     {
436         int r, c, pos;
437
438         /******
439         * Compute the x-derivative. Adjust the derivative at the borders to avoid
440         * losing pixels.
441         * *****/
442         if(VERBOSE) printf("Computing the X-direction derivative.\n");
443         for(r=0;r<rows;r++){
444             pos = r * cols;
445             deltax[pos] = smoothedimg[pos+1] - smoothedimg[pos];
446             pos++;
447             for(c=1;c<(cols-1);c++,pos++){
448                 deltax[pos] = smoothedimg[pos+1] - smoothedimg[pos-1];
449             }
450             deltax[pos] = smoothedimg[pos] - smoothedimg[pos-1];
451         }
452
453         /******
454         * Compute the y-derivative. Adjust the derivative at the borders to avoid
455         * losing pixels.
456         * *****/
457         if(VERBOSE) printf("Computing the Y-direction derivative.\n");
458         for(c=0;c<cols;c++){
459             pos = c;
460             deltay[pos] = smoothedimg[pos+cols] - smoothedimg[pos];
461             pos += cols;
462             for(r=1;r<(rows-1);r++,pos+=cols){
463                 deltay[pos] = smoothedimg[pos+cols] - smoothedimg[pos-cols];
464             }

```

```

465         deltax[pos] = smoothedimg[pos] - smoothedimg[pos-cols];
466     }
467 }
468
469 void main()
470 {
471     imgs deltax, deltax;
472
473     derivative_x_y(smoothedim, ROWS, COLS, deltax, deltax);
474
475     delta_y=deltax;
476     delta_x=deltax;
477 }
478 };
479
480 behavior Magnitude_X_Y(in imgs delta_x, in imgs delta_y, out imgs magnitude,
481                       out imgs delta_x_p1, out imgs delta_y_p1)
482 {
483     /******
484     * PROCEDURE: magnitude_x_y
485     * PURPOSE: Compute the magnitude of the gradient. This is the square root of
486     * the sum of the squared derivative values.
487     * NAME: Mike Heath
488     * DATE: 2/15/96
489     *****/
490     void magnitude_x_y(short int *deltax, short int *deltay, int rows, int cols,
491                      short int *mag)
492     {
493         int r, c, pos, sq1, sq2;
494
495         for(r=0,pos=0;r<rows;r++){
496             for(c=0;c<cols;c++,pos++){
497                 sq1 = (int)deltax[pos] * (int)deltax[pos];
498                 sq2 = (int)deltay[pos] * (int)deltay[pos];
499                 mag[pos] = (short)(0.5 + sqrt((float)sq1 + (float)sq2));
500             }
501         }
502     }
503
504     void main()
505     {
506         imgs mag;
507
508         magnitude_x_y(delta_x, delta_y, ROWS, COLS, mag);
509
510         magnitude = mag;
511         delta_x_p1 = delta_x;
512         delta_y_p1 = delta_y;
513     }
514 };
515
516 behavior Mag_Delta(in imgs smoothedim, out imgs delta_x, out imgs delta_y,

```



```

517         out imgs magnitude)
518 {
519     imgs delta_x_p1;
520     imgs delta_y_p1;
521
522     Derivative_X-Y derivative_x_y(smoothedim, delta_x_p1, delta_y_p1);
523     Magnitude_X-Y magnitude_x_y(delta_x_p1, delta_y_p1, magnitude,
524                                 delta_x, delta_y);
525
526     void main()
527     {
528         derivative_x_y.main();
529         magnitude_x_y.main();
530     }
531 };
532
533 behavior Non_Max_Supp(in imgs delta_x, in imgs delta_y, in imgs magnitude,
534                     out img nms, out imgs magnitude_p1)
535 {
536
537     /******
538     * PROCEDURE: non_max_supp
539     * PURPOSE: This routine applies non-maximal suppression to the magnitude of
540     * the gradient image.
541     * NAME: Mike Heath
542     * DATE: 2/15/96
543     *****/
544     non_max_supp(short *mag, short *gradx, short *grady, int nrows, int ncols,
545                unsigned char *result)
546     {
547         int rowcount, colcount, count;
548         short *magrowptr, *magptr;
549         short *gxrowptr, *gxptr;
550         short *gyrowptr, *gyptr, z1, z2;
551         int gx, gy;
552         short m00;
553     #ifdef FIXED_POINT
554         int mag1, mag2, xperp, yperp;
555     #else
556         float mag1, mag2, xperp, yperp;
557     #endif
558         unsigned char *resultrowptr, *resultptr;
559
560     /******
561     * Zero the edges of the result image.
562     *****/
563     for(count=0, resultrowptr=result, resultptr=result+ncols*(nrows-1);
564         count<ncols; resultptr++, resultrowptr++, count++){
565         *resultrowptr = *resultptr = (unsigned char) 0;
566     }
567
568     for(count=0, resultptr=result, resultrowptr=result+ncols-1;

```

```

569         count<nrows; count++,resultptr+=ncols,resultrowptr+=ncols){
570         *resultptr = *resultrowptr = (unsigned char) 0;
571     }
572
573     /******
574     * Suppress non-maximum points.
575     * *****/
576     for (rowcount=1,magrowptr=mag+ncols+1,gxrowptr=gradx+ncols+1,
577         gyrowptr=grady+ncols+1,resultrowptr=result+ncols+1;
578
579     #ifdef SMP_PATCH
580         rowcount<=nrows-2;
581     #else
582         rowcount<nrows-2;
583     #endif
584
585         rowcount++,magrowptr+=ncols,gyrowptr+=ncols,gxrowptr+=ncols,
586         resultrowptr+=ncols){
587     for (colcount=1,magptr=magrowptr,gxptr=gxrowptr,gyptr=gyrowptr,
588     #ifdef SMP_PATCH
589         resultptr=resultrowptr;colcount<=ncols-2;
590     #else
591         resultptr=resultrowptr;colcount<ncols-2;
592     #endif
593
594         colcount++,magptr++,gxptr++,gyptr++,resultptr++){
595     m00 = *magptr;
596     if (m00 == 0){
597         *resultptr = (unsigned char) NOEDGE;
598     }
599     else{
600     #ifdef FIXED_POINT
601         gx = *gxptr;
602         gy = *gyptr;;
603         xperp = -(gx<<SHIFT_BIT)/m00;
604         yperp = (gy<<SHIFT_BIT)/m00;
605     #else
606         xperp = -(gx = *gxptr)/((float)m00);
607         yperp = (gy = *gyptr)/((float)m00);
608     #endif
609
610         }
611
612     if (gx >= 0){
613         if (gy >= 0){
614             if (gx >= gy)
615             {
616                 /* III */
617                 /* Left point */
618                 z1 = *(magptr - 1);
619                 z2 = *(magptr - ncols - 1);
620
621                 mag1 = (m00 - z1)*xperp + (z2 - z1)*yperp;

```

```

621         /* Right point */
622         z1 = *(magptr + 1);
623         z2 = *(magptr + ncols + 1);
624
625         mag2 = (m00 - z1)*xperp + (z2 - z1)*yperp;
626     }
627     else
628     {
629         /* 110 */
630         /* Left point */
631         z1 = *(magptr - ncols);
632         z2 = *(magptr - ncols - 1);
633
634         mag1 = (z1 - z2)*xperp + (z1 - m00)*yperp;
635
636         /* Right point */
637         z1 = *(magptr + ncols);
638         z2 = *(magptr + ncols + 1);
639
640         mag2 = (z1 - z2)*xperp + (z1 - m00)*yperp;
641     }
642 }
643 else
644 {
645     if (gx >= -gy)
646     {
647         /* 101 */
648         /* Left point */
649         z1 = *(magptr - 1);
650         z2 = *(magptr + ncols - 1);
651
652         mag1 = (m00 - z1)*xperp + (z1 - z2)*yperp;
653
654         /* Right point */
655         z1 = *(magptr + 1);
656         z2 = *(magptr - ncols + 1);
657
658         mag2 = (m00 - z1)*xperp + (z1 - z2)*yperp;
659     }
660     else
661     {
662         /* 100 */
663         /* Left point */
664         z1 = *(magptr + ncols);
665         z2 = *(magptr + ncols - 1);
666
667         mag1 = (z1 - z2)*xperp + (m00 - z1)*yperp;
668
669         /* Right point */
670         z1 = *(magptr - ncols);
671         z2 = *(magptr - ncols + 1);
672

```

```

673         mag2 = (z1 - z2)*xperp + (m00 - z1)*yperp;
674     }
675 }
676 }
677 else
678 {
679     if ((gy = *gyptr) >= 0)
680     {
681         if (-gx >= gy)
682         {
683             /* 011 */
684             /* Left point */
685             z1 = *(magptr + 1);
686             z2 = *(magptr - ncols + 1);
687
688             mag1 = (z1 - m00)*xperp + (z2 - z1)*yperp;
689
690             /* Right point */
691             z1 = *(magptr - 1);
692             z2 = *(magptr + ncols - 1);
693
694             mag2 = (z1 - m00)*xperp + (z2 - z1)*yperp;
695         }
696         else
697         {
698             /* 010 */
699             /* Left point */
700             z1 = *(magptr - ncols);
701             z2 = *(magptr - ncols + 1);
702
703             mag1 = (z2 - z1)*xperp + (z1 - m00)*yperp;
704
705             /* Right point */
706             z1 = *(magptr + ncols);
707             z2 = *(magptr + ncols - 1);
708
709             mag2 = (z2 - z1)*xperp + (z1 - m00)*yperp;
710         }
711     }
712     else
713     {
714         if (-gx > -gy)
715         {
716             /* 001 */
717             /* Left point */
718             z1 = *(magptr + 1);
719             z2 = *(magptr + ncols + 1);
720
721             mag1 = (z1 - m00)*xperp + (z1 - z2)*yperp;
722
723             /* Right point */
724             z1 = *(magptr - 1);

```

```

725         z2 = *(magptr - ncols - 1);
726
727         mag2 = (z1 - m00)*xperp + (z1 - z2)*yperp;
728     }
729     else
730     {
731         /* 000 */
732         /* Left point */
733         z1 = *(magptr + ncols);
734         z2 = *(magptr + ncols + 1);
735
736         mag1 = (z2 - z1)*xperp + (m00 - z1)*yperp;
737
738         /* Right point */
739         z1 = *(magptr - ncols);
740         z2 = *(magptr - ncols - 1);
741
742         mag2 = (z2 - z1)*xperp + (m00 - z1)*yperp;
743     }
744 }
745 }
746
747 /* Now determine if the current point is a maximum point */
748
749 if ((mag1 > 0) || (mag2 > 0))
750 {
751     *resultptr = (unsigned char) NOEDGE;
752 }
753 else
754 {
755     if (mag2 == 0)
756         *resultptr = (unsigned char) NOEDGE;
757     else
758         *resultptr = (unsigned char) POSSIBLE_EDGE;
759 }
760 }
761 }
762 return 0;
763 }
764
765 void main()
766 {
767
768     img result;
769     int i;
770     //initialise nms to all zero by jiangwan
771     for( i=0; i<SIZE; i++ )
772     {
773         result[i] = 0;
774     }
775
776     non_max_supp(magnitude, delta_x, delta_y, ROWS, COLS, result);

```

```

777         magnitude_p1 = magnitude;
778         nms = result;
779     }
780 }
781 };
782
783
784 behavior Apply_Hysteresis(in imgs magnitude, in img nms, i_img_sender ImgOut)
785 {
786     unsigned char edge[SIZE];
787
788     /*****
789     * PROCEDURE: follow_edges
790     * PURPOSE: This procedure edges is a recursive routine that traces eds along
791     * all paths whose magnitude values remain above some specifyable lower
792     * threshold.
793     * NAME: Mike Heath
794     * DATE: 2/15/96
795     *****/
796     void follow_edges(unsigned char *edgemapptr, short *edgemagptr, short lowval,
797                     int cols)
798     {
799         note _SCC_ANALYSIS_IgnoreParThAnalyzeRecursiveFunction=true;
800         short *tempmagptr;
801         unsigned char *tempmapptr;
802         int i;
803
804         int x[8] = {1,1,0,-1,-1,-1,0,1},
805                 y[8] = {0,1,1,1,0,-1,-1,-1};
806
807         for(i=0;i<8;i++){
808             tempmapptr = edgemapptr - y[i]*cols + x[i];
809             tempmagptr = edgemagptr - y[i]*cols + x[i];
810
811             if((*tempmapptr == POSSIBLE_EDGE) && (*tempmagptr > lowval)){
812                 *tempmapptr = (unsigned char) EDGE;
813                 follow_edges(tempmapptr,tempmagptr, lowval, cols);
814             }
815         }
816     }
817 }
818
819 /*****
820 * PROCEDURE: apply_hysteresis
821 * PURPOSE: This routine finds edges that are above some high threshold or
822 * are connected to a high pixel by a path of pixels greater than a low
823 * threshold.
824 * NAME: Mike Heath
825 * DATE: 2/15/96
826 *****/
827 void apply_hysteresis(short int *mag, unsigned char *nmsimg, int rows, int cols,
828                     float tlow, float thigh, unsigned char *edge0)

```

```

829     {
830         int r, c, pos, numedges, highcount, lowthreshold, highthreshold,
831             hist[32768];
832         short int maximum_mag;
833
834         /******
835          * Initialize the edge map to possible edges everywhere the non-maximal
836          * suppression suggested there could be an edge except for the border. At
837          * the border we say there can not be an edge because it makes the
838          * follow_edges algorithm more efficient to not worry about tracking an
839          * edge off the side of the image.
840          *****/
841         for (r=0, pos=0; r<rows; r++){
842             for (c=0; c<cols; c++, pos++){
843                 if (nmsimg[pos] == POSSIBLE_EDGE) edge0[pos] = POSSIBLE_EDGE;
844                 else edge0[pos] = NOEDGE;
845             }
846         }
847
848         for (r=0, pos=0; r<rows; r++, pos+=cols){
849             edge0[pos] = NOEDGE;
850             edge0[pos+cols-1] = NOEDGE;
851         }
852         pos = (rows-1) * cols;
853         for (c=0; c<cols; c++, pos++){
854             edge0[c] = NOEDGE;
855             edge0[pos] = NOEDGE;
856         }
857
858         /******
859          * Compute the histogram of the magnitude image. Then use the histogram to
860          * compute hysteresis thresholds.
861          *****/
862         for (r=0; r<32768; r++) hist[r] = 0;
863         for (r=0, pos=0; r<rows; r++){
864             for (c=0; c<cols; c++, pos++){
865                 if (edge0[pos] == POSSIBLE_EDGE) hist[mag[pos]]++;
866             }
867         }
868
869         /******
870          * Compute the number of pixels that passed the nonmaximal suppression.
871          *****/
872         for (r=1, numedges=0; r<32768; r++){
873             if (hist[r] != 0) maximum_mag = r;
874             numedges += hist[r];
875         }
876
877         highcount = (int)(numedges * thigh + 0.5);
878
879         /******
880          * Compute the high threshold value as the (100 * thigh) percentage point

```

```

881     * in the magnitude of the gradient histogram of all the pixels that passes
882     * non-maximal suppression. Then calculate the low threshold as a fraction
883     * of the computed high threshold value. John Canny said in his paper
884     * "A Computational Approach to Edge Detection" that "The ratio of the
885     * high to low threshold in the implementation is in the range two or three
886     * to one." That means that in terms of this implementation, we should
887     * choose tlow  $\approx$  0.5 or 0.33333.
888     *****/
889     r = 1;
890     numedges = hist[1];
891     while((r<(maximum_mag-1)) && (numedges < highcount)){
892         r++;
893         numedges += hist[r];
894     }
895     highthreshold = r;
896     lowthreshold = (int)(highthreshold * tlow + 0.5);
897
898     if(VERBOSE){
899         printf("The input low and high fractions of %f and %f computed to\n",
900             tlow, thigh);
901         printf("magnitude of the gradient threshold values of: %d %d\n",
902             lowthreshold, highthreshold);
903     }
904
905     /******
906     * This loop looks for pixels above the highthreshold to locate edges and
907     * then calls follow_edges to continue the edge.
908     *****/
909     for(r=0,pos=0;r<rows;r++){
910         for(c=0;c<cols;c++,pos++){
911             if((edge0[pos] == POSSIBLE_EDGE) && (mag[pos] >= highthreshold)){
912                 edge0[pos] = EDGE;
913                 follow_edges((edge0+pos), (mag+pos), lowthreshold, cols);
914             }
915         }
916     }
917
918     /******
919     * Set all the remaining possible edges to non-edges.
920     *****/
921     for(r=0,pos=0;r<rows;r++){
922         for(c=0;c<cols;c++,pos++) if(edge0[pos] != EDGE) edge0[pos] = NOEDGE;
923     }
924 }
925
926 void main()
927 {
928     apply_hysteresis(magnitude, nms, ROWS, COLS, TLOW, THIGH, edge);
929     ImgOut.send(edge);
930 }
931 };
932

```



```

933
934 /*****
935 * DUT: To perform canny edge detection.
936 *****/
937
938 behavior DUT(i_img_receiver ImgIn, i_img_sender ImgOut)
939 {
940 //changed by Jiang Wan for pipeline
941     piped imgs smoothedim;
942     piped imgs delta_x;
943     piped imgs delta_y;
944     piped imgs magnitude;
945     piped imgs magnitude_p1;
946     piped img nms;
947     int i;
948
949     Gaussian_Smooth      gaussian_smooth(ImgIn, smoothedim);
950     Mag_Delta            mag_delta(smoothedim, delta_x, delta_y, magnitude);
951     Non_Max_Supp        non_max_supp(delta_x, delta_y, magnitude, nms, magnitude_p1);
952     Apply_Hysteresis    apply_hysteresis(magnitude_p1, nms, ImgOut);
953
954     void main()
955     {
956         pipe(i=0; i<IMGNUM; i++)
957         {
958             gaussian_smooth.main();
959             mag_delta.main();
960             non_max_supp.main();
961             apply_hysteresis.main();
962         }
963     }
964 };
965
966
967 behavior DataIn(i_img_receiver ImgIn, i_img_sender ImgOut)
968 {
969     unsigned char image[SIZE];
970
971     void main()
972     {
973         while(1)
974         {
975             ImgIn.receive(&image);
976             ImgOut.send(image);
977         }
978     }
979 };
980
981
982 behavior DataOut(i_img_receiver ImgIn, i_img_sender ImgOut)
983 {
984     unsigned char image[SIZE];

```

```

985
986     void main()
987     {
988         while(1)
989         {
990             ImgIn.receive(&image);
991             ImgOut.send(image);
992         }
993     }
994 };
995
996
997 behavior Platform(i_img_receiver ImgIn, i_img_sender ImgOut)
998 {
999     c_img_queue    q1(2 ul),
1000                 q2(2 ul);
1001     DataIn        din(ImgIn, q1);
1002     DUT           canny(q1, q2);
1003     DataOut       dout(q2, ImgOut);
1004
1005     void main()
1006     {
1007         par{
1008             din.main();
1009             canny.main();
1010             dout.main();
1011         }
1012     }
1013 };
1014
1015
1016 behavior Main()
1017 {
1018     sim_time      t [IMGNUM];
1019     c_img_queue   q1(2 ul),
1020                 q2(2 ul);
1021
1022     Stimulus      stimulus(q1, t);
1023     Platform      platform(q1, q2);
1024     Monitor       monitor(q2, t);
1025
1026     int main()
1027     {
1028         par{
1029             stimulus.main();
1030             platform.main();
1031             monitor.main();
1032         }
1033     return 0; // never reached
1034     }
1035 };

```

## A.2 Makefile for TLM generation in SCE

SCE tool has a graphical user interface for design space exploration. However, in order to ease the regeneration of the TLM model, a Makefile has been created. The Makefile contains the commands to SCE tool for different design settings and refinements.

```
1 #clock period for ARM CPU (in nanoseconds)
2 CPUCLKP      = 10000
3
4 #bus
5 BUSCLKP      = 20000
6
7 SPECC        = /opt/sce
8 SCE_PATH     = $(SPECC)
9 SCE_DB_PATH  = $(SCE_PATH)/share/sce/db
10 PROC_DB_PATH = $(SCE_DB_PATH)/processors
11 COMM_DB_PATH = $(SCE_DB_PATH)/communication
12 BUS_DB_PATH  = $(SCE_DB_PATH)/busses
13
14 SYSC_PATH    = /opt/pkg/systemc-2.1.v1
15
16 SCENV        =SPECC=$(SPECC) SCERC_PATH=$PWD/.sce \
17              PATH=$(SPECC)/bin:${PATH} \
18              LD_LIBRARY_PATH=$(SPECC)/lib:${LD_LIBRARY_PATH:+;}${LD_LIBRARY_PATH}
19
20
21 PROC_CACHE   = .sce/processors
22 COMM_CACHE   = .sce/communication
23 BUS_CACHE    = .sce/busses
24 RTL_CACHE    = .sce/rtl
25 SCE_LOCK     = .sce/.sceerc.lock
26
27 FINALPKG     = final.tar.gz
28
29 MAINSIRFILES = cannySpec.sir cannyArch.sir cannySched.sir \
30               cannyNet.sir cannyTlm.sir cannyComm.sir \
31               cannyRTLC.sir cannyISS.sir
32
33 SYSC          = g++ -m32
34 SYSCOPT       = -Di386
35 SYSCINC       = -I$(SYSC_PATH)/include
36 SYSCLIB       = -L$(SYSC_PATH)/lib-linux -lsystemc
37
38 TIME          = /usr/bin/time
39
40 SCC           = scc
41 SCCOPT        = -ww -vv -d -x1x
42 SCCINC        = -Isrc
43 SCCIMP        = -Psrc
44 SIR_RENAME    = sir_rename
45 SIR_RENAMEOPT = -v
46
47
48 SIR_STATS     = sir_stats
49 SIR_STATSOPT  =
50
51 SIR_GEN       = sir_gen
52 SIR_GENOPT    = -vv
53
54 SIR_IMPORT    = sir_import
55 SIR_IMPORTOPT = -v
56
57 SCSH         = scsh
58 SCSHOPT      =
59
60 SCE_RETYPE    = sce_retype
61 SCE_RETYPEOPT = -v
62
63 SCE_IMPORT    = sce_import
64 SCE_IMPORTOPT = -v
```

```

65
66 SCE_ALLOCATE = sce_allocate
67 SCE_ALLOCATEOPT = -v
68
69 SCE_TOP = sce_top
70 SCE_TOPOPT = -v
71
72 SCE_MAP = sce_map
73 SCE_MAPOPT = -v
74
75 SCE_SCHEDULE = sce_schedule
76 SCE_SCHEDULEOPT = -v
77
78 SCE_CONNECT = sce_connect
79 SCE_CONNECTOPT = -v
80
81 SCE_PROJECT = sce_project
82 SCE_PROJECTOPT = -v
83
84 SC_PROF = $(TIME) scprof
85 SC_PROFOPT = -v
86
87 SCAR = $(TIME) scar
88 SCAROPT =
89
90 SCOS = $(TIME) scos
91 SCOSOPT = -v
92
93 SCNR = $(TIME) scnr
94 SCNROPT = -v -O -falign -fmerge
95
96 SCCR = $(TIME) sccr
97 SCCROPT = -v -O
98
99 ECHO = echo
100 MAKE = make
101 # --- TARGET RULES ---
102
103
104 all:
105     $(SCENV) $(MAKE) cannyTlm.SIM_OK
106
107 test:
108     $(SCENV) $(MAKE) cannySpec.SIM_OK
109
110 final: $(FINALPKG)
111
112 clean:
113     $(RM) cannySpec cannyArch cannySched cannyNet \
114         cannyTlm cannyComm cannyRTLc cannyISS
115     $(RM) $(FINALPKG)
116     $(RM) *.ins *.sysc
117     $(RM) *.si *.sir *.cc *.h *.o *.cpp *.hpp
118     $(RM) *.dpr *.prf
119     $(RM) *.SIM_OK
120     $(RM) core
121     $(RM) image_s_0.60_1.0.30_h_0.80_*
122     $(RM) -r $(PROC_CACHE)
123     $(RM) -r $(COMM_CACHE)
124     $(RM) -r $(BUS_CACHE)
125     $(RM) -r $(RTL_CACHE)
126     $(RM) $(SCE_LOCK)
127
128 stats: $(MANSIRFILES)
129     $(SIR_STATS) $(SIR_STATSOPT) $(MANSIRFILES)
130
131 loc: $(MANSIRFILES)
132     $(SIR_STATS) $(SIR_STATSOPT) -BCI $(MANSIRFILES)
133
134
135 # --- GENERAL RULES --- # o $@ $*

```

```

136 .SUFFIXES:
137 .SUFFIXES:      .ins .sir .sir .ins .ana .sir
138 .SUFFIXES:      .pgm .SIM_OK
139
140 .sir.ins.sir:
141     @$(ECHO)      "***"
142     @$(ECHO)      "***_Instrumenting_$_*_for_profiling_..."
143     @$(ECHO)      "***"
144     $(SCPROF) -v -m -i $< -o $@ $*
145
146 .sir:
147     @$(ECHO)      "***"
148     @$(ECHO)      "***_Compiling_$_*_for_execution_..."
149     @$(ECHO)      "***"
150     $(SCC) $* -sir2out $(SCCOPT) $(SCCINC) $(SCCIMP)
151
152 .ins.sir.ins:
153     @$(ECHO)      "***"
154     @$(ECHO)      "***_Compiling_$_*_for_execution_with_profiling_..."
155     @$(ECHO)      "***"
156     $(SCC) $* -sir2out $(SCCOPT) $(SCCINC)$(SCCIMP) -i $< -o $@ -lscprof
157
158 .pgm.SIM_OK:
159     @$(ECHO)      "***"
160     @$(ECHO)      "***_Simulating_$_*"
161     @$(ECHO)      "***"
162     $(TIME) ./$_.ins | tee log_execution_.$*
163     @$(ECHO)      "***"
164     @$(ECHO)      "***_Simulation_successful!"
165     @$(ECHO)      "***"
166
167 .sir.ana.sir:
168     @$(ECHO)      "***"
169     @$(ECHO)      "***_Back-annotating_profiling_data_and_running_estimation_..."
170     @$(ECHO)      "***"
171     $(SCPROF) -E $(SCPROFOPT) -i $< -o $@ $*
172
173 # — SPECIFIC RULES —————
174
175 # — compile the sources
176 cannySpecpre.sir: canny.sc
177     @$(ECHO)      "***"
178     @$(ECHO)      "***_Compiling_the_sources_..."
179     @$(ECHO)      "***"
180     $(SCC) canny -c2sir $(SCCOPT) $(SCCINC) $(SCCIMP)\
181         -i canny.sc -o cannySpecpre.sir
182
183 cannySpec.sir: cannySpecpre.sir
184     @$(ECHO)      "***"
185     @$(ECHO)      "***_Setting_top_level_of_design_under_test_..."
186     @$(ECHO)      "***"
187     $(SCE_TOP) $(SCE_TOPOPT) -s Platform \
188         -i cannySpecpre.sir -o cannySpec.sir cannySpec
189
190 cannySpec.ins.sir: cannySpec.sir
191 cannySpec:      cannySpec.sir
192 cannySpec.ins:  cannySpec.ins.sir
193 cannySpec.pgm:  cannySpec.ins
194 cannySpec.SIM_OK:      cannySpec.pgm
195 cannySpec.ana.sir: cannySpec.sir cannySpec.SIM_OK
196
197 # — back-annotate raw profiling data
198 cannySpec.prof.sir: cannySpec.sir cannySpec.SIM_OK
199     @$(ECHO)      "***"
200     @$(ECHO)      "***_Back-annotating_raw_profiling_data_..."
201     @$(ECHO)      "***"
202     $(SCPROF) -p $(SCPROFOPT) -i cannySpec.sir -o cannySpec.prof.sir cannySpec
203
204 cannySpec.3.sir: cannySpec.prof.sir
205     @$(ECHO)      "***"
206     @$(ECHO)      "***_Allocating_the_component_..."

```

```

207     @$(ECHO) "***"
208     $(SCE_ALLOCATE) $(SCE_ALLOCATEOPT) \
209         -g PORTA_HCLK_PERIOD=$(BUSCLKP) -g CLOCK_PERIOD=$(CPUCLKP) -p CPU=ARM,7TDMI \
210         -g CLOCK_PERIOD= -p IO_OUT=HW_Virtual \
211         -g CLOCK_PERIOD= -p IO_IN=HW_Virtual \
212         -g CLOCK_PERIOD= -p Hardware_BlurX=HW_Standard \
213         -g CLOCK_PERIOD= -p Hardware_BlurY=HW_Standard \
214         -i cannySpec.prof.sir -o cannySpec.3.sir cannySpec
215
216 cannySpec.4.sir:    cannySpec.3.sir
217     @$(ECHO) "***"
218     @$(ECHO) "***_Map_the_components..."
219     @$(ECHO) "***"
220     $(SCE_MAP) $(SCE_MAPOPT) -p Platform.canny=CPU \
221         -p Platform.din=IO_IN \
222         -p Platform.dout=IO_OUT \
223         -p Platform.canny.gaussian_smooth.blurY_par=Hardware_BlurY \
224         -p Platform.canny.gaussian_smooth.blurX_par=Hardware_BlurX \
225         -i cannySpec.3.sir -o cannySpec.4.sir cannySpec
226
227 # --- analyze the profile given the behavior mapping
228 cannySpec.ana1.sir:    cannySpec.4.sir
229     @$(ECHO) "***"
230     @$(ECHO) "***_Analyzing_the_behavior_mapping..."
231     @$(ECHO) "***"
232     $(SCPROF) -a $(SCPROFOPT) -i cannySpec.4.sir \
233         -o cannySpec.ana1.sir canny
234
235 # --- estimate the profile given the behavior mapping
236 cannySpec.ana2.sir:    cannySpec.4.sir
237     @$(ECHO) "***"
238     @$(ECHO) "***_Estimating_the_behavior_mapping..."
239     @$(ECHO) "***"
240     $(SCPROF) -e $(SCPROFOPT) -i cannySpec.4.sir \
241         -o cannySpec.ana2.sir canny
242
243 cannySpec.arch.in.sir:    cannySpec.ana2.sir cannySpec.ana1.sir
244     @$(ECHO) "***"
245     @$(ECHO) "***_Importing_the_PEs_into_the_design..."
246     @$(ECHO) "***"
247     $(SCE_IMPORT) $(SCE_IMPORTOPT) -a \
248         -i cannySpec.ana2.sir -o cannySpec.arch.in.sir \
249         cannySpec
250
251 cannyArch.sir:    cannySpec.arch.in.sir
252     @$(ECHO) "***"
253     @$(ECHO) "***_Performing_architecture_refinement..."
254     @$(ECHO) "***"
255     $(SCAR) cannySpec -b -m -c -w $(SCAROPT) \
256         -i cannySpec.arch.in.sir -o cannySpec.arch.sir
257     $(SIR_RENAME) $(SIR_RENAMEOPT) -i cannySpec.arch.sir \
258         cannySpec cannyArch
259     $(SIR_STATS) $(SIR_STATSOPT) $@
260
261 cannyArch.ins.sir:    cannyArch.sir
262 cannyArch.ins:    cannyArch.ins.sir
263 cannyArch.pgm:    cannyArch.ins
264 cannyArch.SIM_OK:    cannyArch.pgm
265 cannyArch.ana.sir:    cannyArch.sir cannyArch.SIM_OK
266
267 cannyArch.2.sir:    cannyArch.ana.sir
268     @$(ECHO) "***"
269     @$(ECHO) "***_Deciding_scheduling..."
270     @$(ECHO) "***"
271     $(SCE_SCHEDULE) $(SCE_SCHEDULEOPT) -r -t Platform \
272         -k CPU=ARM,7TDMI,OSNONE,$(CPUCLKP)_$(BUSCLKP)_0 \
273         -i cannyArch.ana.sir -o cannyArch.2.sir cannyArch
274
275 # --- import components for scheduling refinement
276 cannyArch.sched.in.sir:    cannyArch.2.sir
277     @$(ECHO) "***"

```

```

278     @$(ECHO) "****Importing components needed for scheduling..."
279     @$(ECHO) "****"
280     $(SCE_IMPORT) $(SCE_IMPORTOPT) -s \
281         -i cannyArch.2.sir -o cannyArch.sched.in.sir \
282         cannyArch
283
284 # --- perform static scheduling refinement
285 cannyArch.sched.tmp.sir:    cannyArch.sched.in.sir
286     @$(ECHO) "****"
287     @$(ECHO) "****Performing static scheduling refinement..."
288     @$(ECHO) "****"
289     $(SCAR) cannyArch $(SCAROPT) -s \
290         -i cannyArch.sched.in.sir -o cannyArch.sched.tmp.sir
291
292 # --- perform dynamic scheduling refinement
293 cannySched.sir: cannyArch.sched.tmp.sir
294     @$(ECHO) "****"
295     @$(ECHO) "****Performing dynamic scheduling refinement..."
296     @$(ECHO) "****"
297     $(SCOS) cannyArch $(SCOSOPT) \
298         -i cannyArch.sched.tmp.sir -o cannyArch.sched.sir
299     $(SIR_RENAME) $(SIR_RENAMEOPT) -i cannyArch.sched.sir \
300         cannyArch cannySched
301     $(SIR_STATS) $(SIR_STATSOPT) $@
302
303 cannySched.ins.sir: cannySched.sir
304 cannySched.ins: cannySched.ins.sir
305 cannySched.pgm: cannySched.ins
306 cannySched.SIM.LOK: cannySched.pgm
307 cannySched.ana.sir: cannySched.sir cannySched.SIM.LOK
308
309
310 cannySched.2.sir:    cannySched.ana.sir
311     @$(ECHO) "****"
312     @$(ECHO) "****Allocating the CPU bus and the Hardware Bus..."
313     @$(ECHO) "****"
314     $(SCE_ALLOCATE) $(SCE_ALLOCATEOPT) \
315         -b CPU_Bus=AMBA_AHB \
316         -b HW_Bus=HardwareBus \
317         -i cannySched.ana.sir -o cannySched.2.sir cannySched
318
319 # --- connect the PE and the busses
320 cannySched.3.sir:    cannySched.2.sir
321     @$(ECHO) "****"
322     @$(ECHO) "****Connecting PE and the busses..."
323     @$(ECHO) "****"
324     $(SCE_CONNECT) $(SCE_CONNECTOPT) -c CPU.PortA=CPU_Bus,master0 \
325         -c Hardware.BlurX,Port0=CPU_Bus,slave4 -c Hardware.BlurX,Port1=HW_Bus,Master \
326         -c Hardware.BlurY,Port0=CPU_Bus,slave5 -c Hardware.BlurY,Port1=HW_Bus,Slave \
327         -c IO.OUT,Port0=CPU_Bus,slave8 \
328         -c IO.IN,Port0=CPU_Bus,slave7 \
329         -i cannySched.2.sir -o cannySched.3.sir cannySched
330
331 # --- import components for network refinement
332 cannySched.net.in.sir: cannySched.3.sir
333     @$(ECHO) "****"
334     @$(ECHO) "****Importing components needed for network refinement..."
335     @$(ECHO) "****"
336     $(SCE_IMPORT) $(SCE_IMPORTOPT) -e \
337         -i cannySched.3.sir -o cannySched.net.in.sir \
338         cannySched
339
340 # --- perform network refinement
341 cannyNet.sir: cannySched.net.in.sir
342     @$(ECHO) "****"
343     @$(ECHO) "****Performing network refinement..."
344     @$(ECHO) "****"
345     $(SCNR) cannySched $(SCNROPT) \
346         -i cannySched.net.in.sir -o cannySched.net.sir
347     $(SIR_RENAME) $(SIR_RENAMEOPT) -i cannySched.net.sir \
348         cannySched cannyNet

```

```

349     $(SIR_STATS) $(SIR_STATSOPT) $@
350
351 cannyNet.ins.sir: cannyNet.sir
352 cannyNet.ins: cannyNet.ins.sir
353 cannyNet.pgm: cannyNet.ins
354 cannyNet.SIM_OK: cannyNet.pgm
355 cannyNet.ana.sir: cannyNet.SIM_OK cannyNet.sir
356
357 # --- define the link parameters
358 cannyNet.2.sir: cannyNet.ana.sir
359     @$(ECHO) "****"
360     @$(ECHO) "****_Defining_the_link_parameters_for_the_busses..."
361     @$(ECHO) "****"
362     $(SCE_MAP) $(SCE_MAPOPT) -l c_link_IO_IN_CPU=0x70000000,0 \
363         -l c_link_CPU_IO_OUT=0x80000000,0 \
364         -l c_link_CPU_Hardware_BlurX=0x40000000,0 \
365         -l c_link_CPU_Hardware_BlurY=0x50000000,0 \
366         -l c_link_Hardware_BlurX_Hardware_BlurY=0x0000,MasterSync0 \
367         -i cannyNet.ana.sir -o cannyNet.2.sir cannyNet
368
369 # --- import the busses into the design
370 cannyNet.comm.in.sir: cannyNet.2.sir
371     @$(ECHO) "****"
372     @$(ECHO) "****_Importing_the_busses_into_the_design..."
373     @$(ECHO) "****"
374     $(SCE_IMPORT) $(SCE_IMPORTOPT) -c \
375         -i cannyNet.2.sir -o cannyNet.comm.in.sir \
376         cannyNet
377
378 # --- perform communication refinement to TLM
379 cannyTlm.sir: cannyNet.comm.in.sir
380     @$(ECHO) "****"
381     @$(ECHO) "****_Performing_communication_refinement_to_TLM..."
382     @$(ECHO) "****"
383     $(SCCR) cannyNet $(SCCROPT) -t \
384         -i cannyNet.comm.in.sir -o cannyNet.comm.sir
385     $(SIR_RENAME) $(SIR_RENAMEOPT) -i cannyNet.comm.sir \
386         cannyNet cannyTlm
387     $(SIR_STATS) $(SIR_STATSOPT) $@
388
389 cannyTlm.ins.sir: cannyTlm.sir
390 cannyTlm.ins: cannyTlm.ins.sir
391 cannyTlm.pgm: cannyTlm.ins
392 cannyTlm.SIM_OK: cannyTlm.pgm
393 cannyTlm.ana.sir: cannyTlm.SIM_OK cannyTlm.sir

```

---