

Frontier



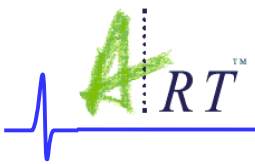
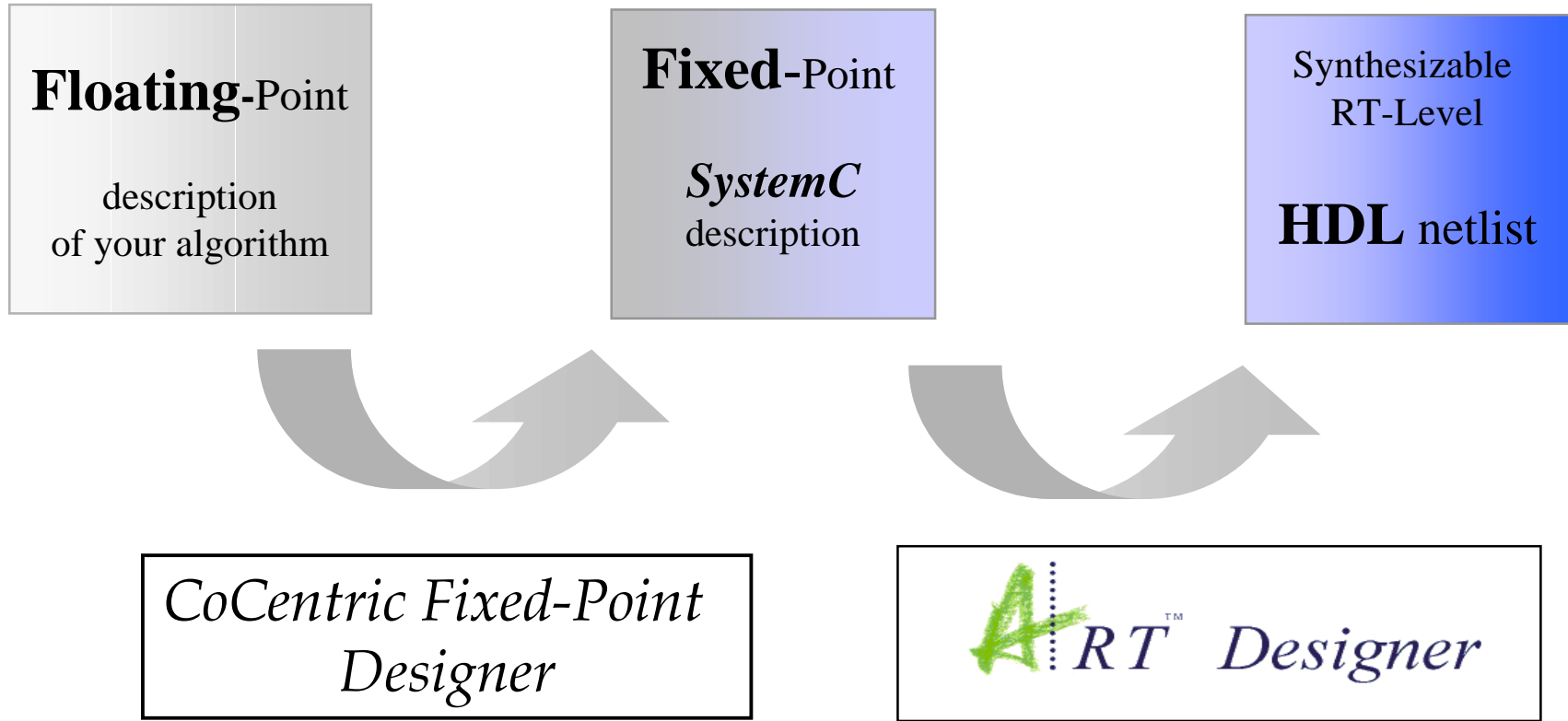
<http://www.frontierd.com>

 *ARTTM Designer*

With SystemC

Case-Study: DCT

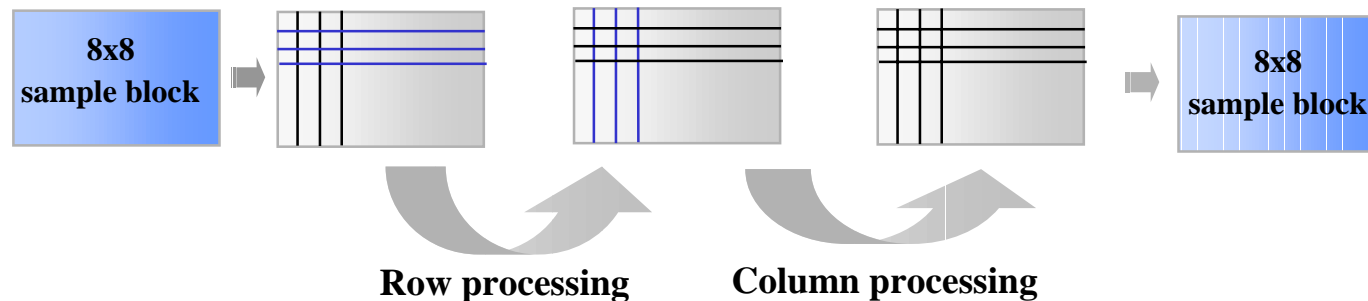
Design flow



Case Study

Discrete Cosine Transformation

What: The DCT transforms without loss, the source image samples to a domain where they can be more efficiently encoded



Constraints: 2 cycles are available for every data element to be processed, meaning in this case 128 cycles

Floating point Algorithm

```
jfdctflt_ccfxd.input_v3.cxx - WordPad
File Edit View Insert Format Help
[Icons]
sc_fixed<11,15,SC_RND,SC_SAT> data_output[64];

void jpeg_fdct_fixed ()
{
#pragma FP_ANALYZE architecture Parrot_types
//#pragma FP_ANALYZE architecture Parrot_architect
float tmp0, tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7;
float tmp10, tmp11, tmp12, tmp13;
float z1, z2, z3, z4, z5, z11, z13;
/*****/
const float O_seven = 0.707106781;
const float O_five = 0.541196100;
const float One_three = 1.306562965;
const float O_three = 0.382683433;
/*****/
int ctr;
int i = 0;

/*added for freeze...*/
/*copy the data*/

float dataptr[64];
for (ctr=0;ctr<64;ctr++)
{
    dataptr[ctr]=sc_fix(data_output[ctr],8,8,SC_RND,SC_SAT);
}

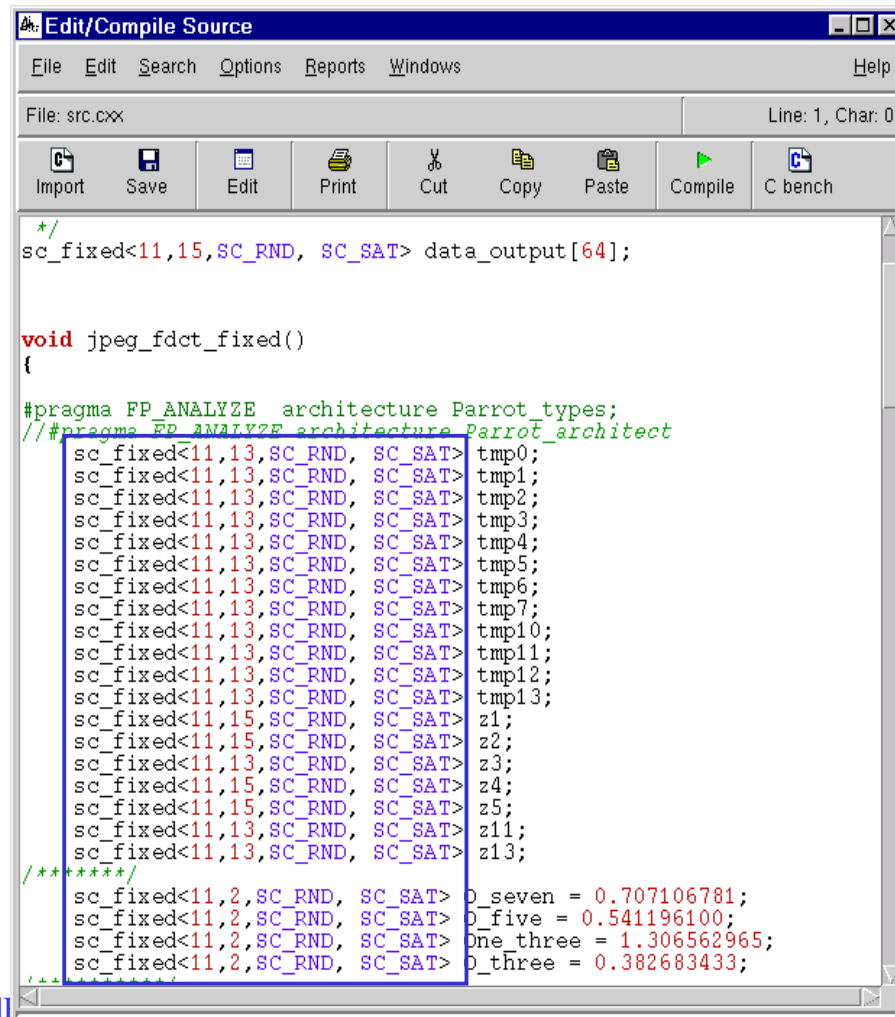
/* Pass 1: process rows. */

for (ctr = 8-1; ctr >= 0; ctr--) {
    tmp0 = dataptr[0+i] + dataptr[7+i];
    tmp7 = dataptr[0+i] - dataptr[7+i];
    tmp1 = dataptr[1+i] + dataptr[6+i];
    tmp6 = dataptr[1+i] - dataptr[6+i];
    tmp2 = dataptr[2+i] + dataptr[5+i];
    tmp5 = dataptr[2+i] - dataptr[5+i];
    tmp3 = dataptr[3+i] + dataptr[4+i];
    tmp4 = dataptr[3+i] - dataptr[4+i];
    tmp10 = dataptr[4+i] + dataptr[3+i];
    tmp11 = dataptr[4+i] - dataptr[3+i];
    tmp12 = dataptr[5+i] + dataptr[2+i];
    tmp13 = dataptr[5+i] - dataptr[2+i];
    tmp9 = dataptr[6+i] + dataptr[1+i];
    tmp8 = dataptr[6+i] - dataptr[1+i];
}
}
```

Case Study

Fixed-point SystemC Algorithm

Converted with CoCentric Fixed-Point Designer (Synopsys)



```
File: src.cxx Line: 1, Char: 0
Import Save Edit Print Cut Copy Paste Compile C bench

*/
sc_fixed<11,15,SC_RND, SC_SAT> data_output[64];

void jpeg_fdct_fixed()
{
#pragma FP_ANALYZE architecture Parrot_types;
//#pragma FP_ANALYZE architecture Parrot_architect
sc_fixed<11,13,SC_RND, SC_SAT> tmp0;
sc_fixed<11,13,SC_RND, SC_SAT> tmp1;
sc_fixed<11,13,SC_RND, SC_SAT> tmp2;
sc_fixed<11,13,SC_RND, SC_SAT> tmp3;
sc_fixed<11,13,SC_RND, SC_SAT> tmp4;
sc_fixed<11,13,SC_RND, SC_SAT> tmp5;
sc_fixed<11,13,SC_RND, SC_SAT> tmp6;
sc_fixed<11,13,SC_RND, SC_SAT> tmp7;
sc_fixed<11,13,SC_RND, SC_SAT> tmp10;
sc_fixed<11,13,SC_RND, SC_SAT> tmp11;
sc_fixed<11,13,SC_RND, SC_SAT> tmp12;
sc_fixed<11,13,SC_RND, SC_SAT> tmp13;
sc_fixed<11,15,SC_RND, SC_SAT> z1;
sc_fixed<11,15,SC_RND, SC_SAT> z2;
sc_fixed<11,13,SC_RND, SC_SAT> z3;
sc_fixed<11,15,SC_RND, SC_SAT> z4;
sc_fixed<11,15,SC_RND, SC_SAT> z5;
sc_fixed<11,13,SC_RND, SC_SAT> z11;
sc_fixed<11,13,SC_RND, SC_SAT> z13;
/*****/
sc_fixed<11,2,SC_RND, SC_SAT> 0_seven = 0.707106781;
sc_fixed<11,2,SC_RND, SC_SAT> 0_five = 0.541196100;
sc_fixed<11,2,SC_RND, SC_SAT> 0ne_three = 1.306562965;
sc_fixed<11,2,SC_RND, SC_SAT> 0_three = 0.382683433;
/*****/
}
```

Case Study

Initial Architecture

```
File Edit Search Options Windows Help
File: P:\marketing\ART\v2.1\demos\fm_demo\projects\design2\architecture.pra Line: 1, Char: 0
Import Save Edit Print Cut Copy Paste Run Lib

// Datapath
//-----
instantiate("artd_library","inport","inport_1");
instantiate("artd_library","outport","outport_1");

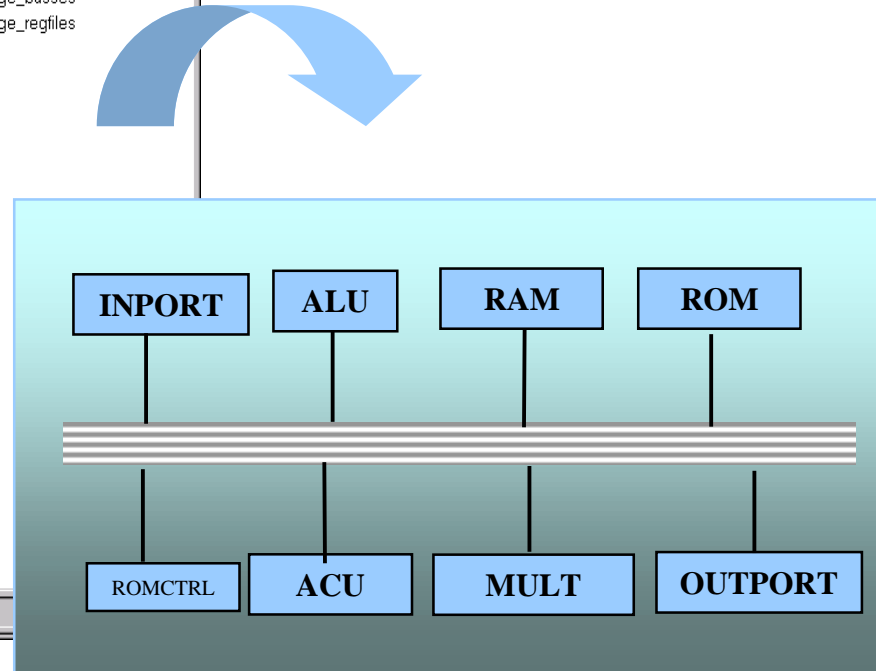
instantiate("artd_library","alu","alu_1");

instantiate("artd_library","romctrl","romctrl_1");
instantiate("artd_library","acu","acu_1");
instantiate("artd_library","rom","rom_1");
instantiate("artd_library","ram","ram_1");

instantiate("artd_library","multp","multp_1"); // pipeline

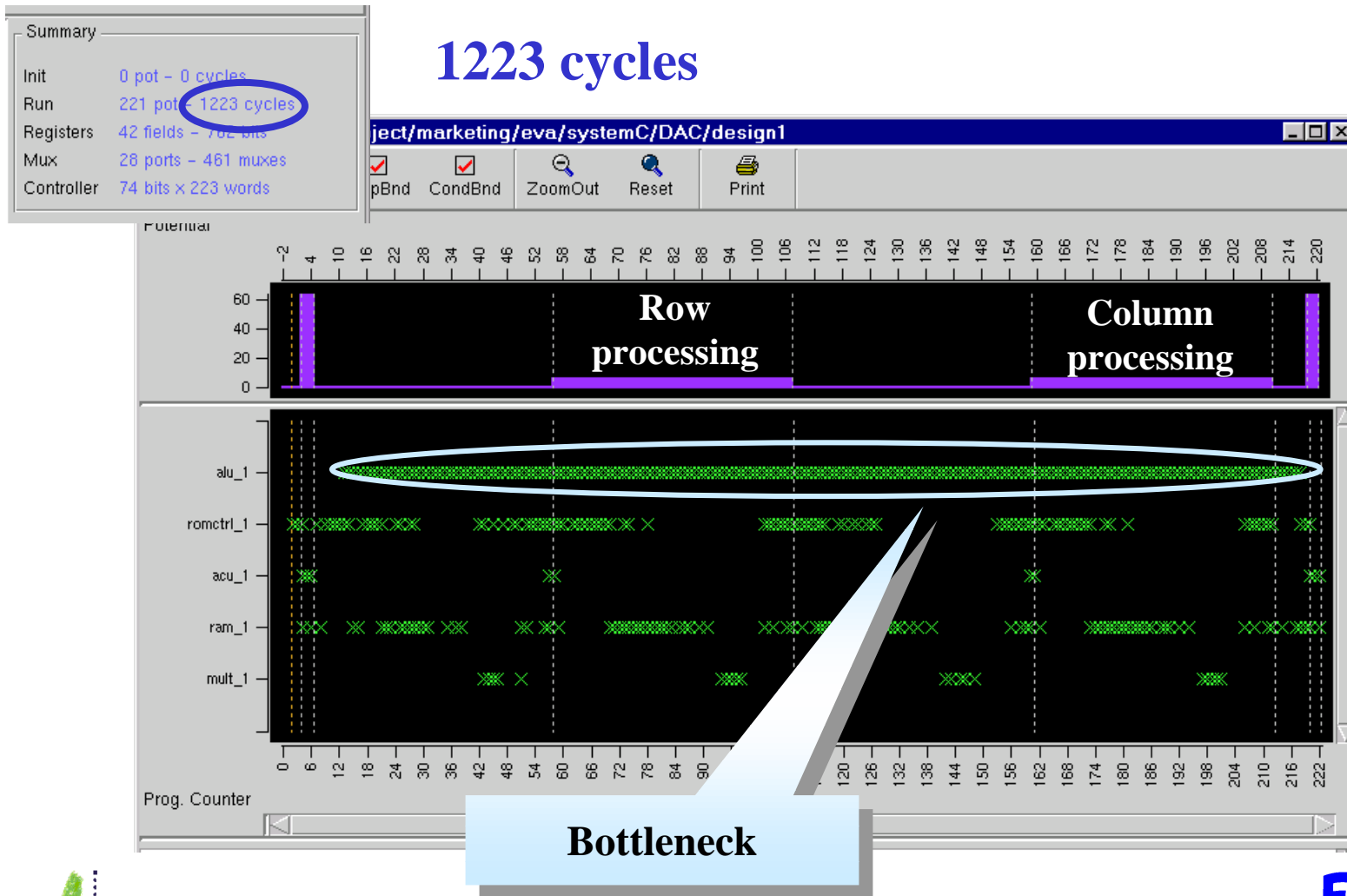
// Multibranch Controller (ctrldelay = 3, jumpdelay = 2)
//-----
instantiate("artd_library","mbc_23","ctrl");

instantiate("libraryName","resourceName","instanceName");
```



Case Study

Default run



Case Study

Architecture improvements : extra alu, acu's and ram

```
File Edit Search Options Windows
File: architecture.pra (modified)
Import Save Edit Print Cut Copy Paste Run Lib

// Datapath
//-----
instantiate("artd_library","alu","alu_1");
instantiate("artd_library","alu","alu_2");

instantiate("artd_library","romctrl","romctrl_1");
instantiate("artd_library","romctrl","romctrl_2");
instantiate("artd_library","romctrl","romctrl_3");

instantiate("artd_library","acu","acu_1");
instantiate("artd_library","acu","acu_2");
instantiate("artd_library","acu","acu_3");

instantiate("artd_library","ram","ram_1");
instantiate("artd_library","ram","ram_2");

instantiate("artd_library","mult","mult_1");

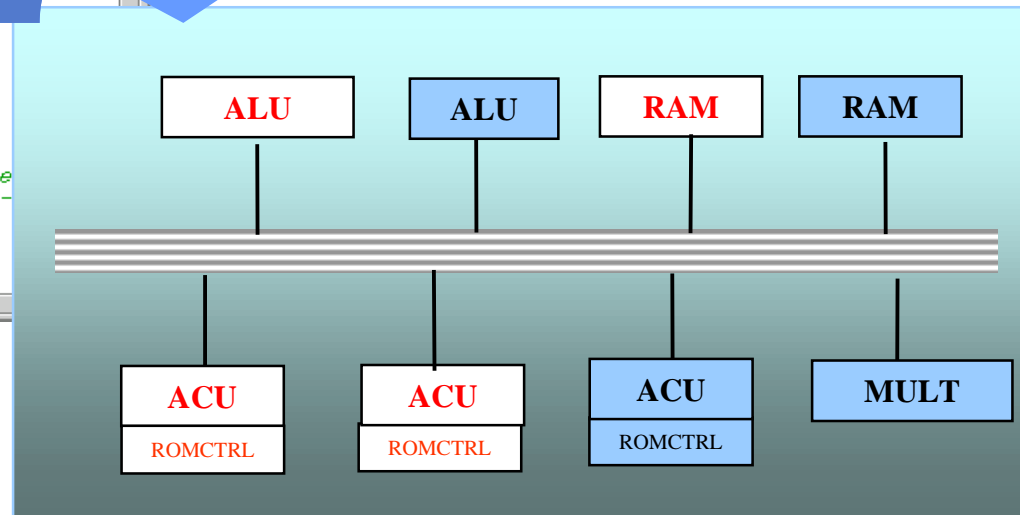
// Multibranch Controller (ctrlldelay = 3, jumpde
//-----
instantiate("artd_library","mbc_23","ctrl");
```

Mapping: Assignment of variables /operations /expressions to the extra resources

```
assign_operation("/.../loop_row/alu2*","alu_2");
assign_operation("/.../loop_col/alu2*","alu_2");

assign_variable("data_io","ram_1");
assign_variable("data_tmp","ram_2");
dedicate("romctrl_2","acu_2");
dedicate("romctrl_3","acu_3");

assign_expression("/.../loop_row",_+i,"acu_2");
assign_expression("/.../loop_row",_+j,"acu_3");
assign_expression("/.../loop_col",_+i,"acu_2");
assign_expression("/.../loop_col",_+j,"acu_3");
```



Case Study

Algorithm improvements: better memory access

Only 1 fetch, by introducing temporary variables

```
/* Pass 1: process rows. */
for (ctr = (8 - 1); ctr >= 0; ctr-- )
{
  tmp0 = (dataptr[(0 + i)] + dataptr[(7 + i)]);
  tmp7 = (dataptr[(0 + i)] - dataptr[(7 + i)]);
  tmp1 = (dataptr[(1 + i)] + dataptr[(6 + i)]);
  tmp6 = (dataptr[(1 + i)] - dataptr[(6 + i)]);
  tmp2 = (dataptr[(2 + i)] + dataptr[(5 + i)]);
  tmp5 = (dataptr[(2 + i)] - dataptr[(5 + i)]);
  tmp3 = (dataptr[(3 + i)] + dataptr[(4 + i)]);
  tmp4 = (dataptr[(3 + i)] - dataptr[(4 + i)]);
  /* Even part */
}
```

```
/* Pass 1: process rows. */
loop_row: for (ctr = (8 - 1); ctr >= 0; ctr-- )
{
  tmp_in0 = data_io[(0 + i)];
  tmp_in1 = data_io[(1 + i)];
  tmp_in2 = data_io[(2 + i)];
  tmp_in3 = data_io[(3 + i)];
  tmp_in4 = data_io[(4 + i)];
  tmp_in5 = data_io[(5 + i)];
  tmp_in6 = data_io[(6 + i)];
  tmp_in7 = data_io[(7 + i)];
  alu2_1: tmp0 = (tmp_in0 + tmp_in7);
  alu2_2: tmp7 = (tmp_in0 - tmp_in7);
  tmp1 = (tmp_in1 + tmp_in6);
  tmp6 = (tmp_in1 - tmp_in6);
  tmp2 = (tmp_in2 + tmp_in5);
  tmp5 = (tmp_in2 - tmp_in5);
  alu2_3: tmp3 = (tmp_in3 + tmp_in4);
  alu2_4: tmp4 = (tmp_in3 - tmp_in4);
  /* Even part */
  tmp10 = (tmp0 + tmp3);
}
```

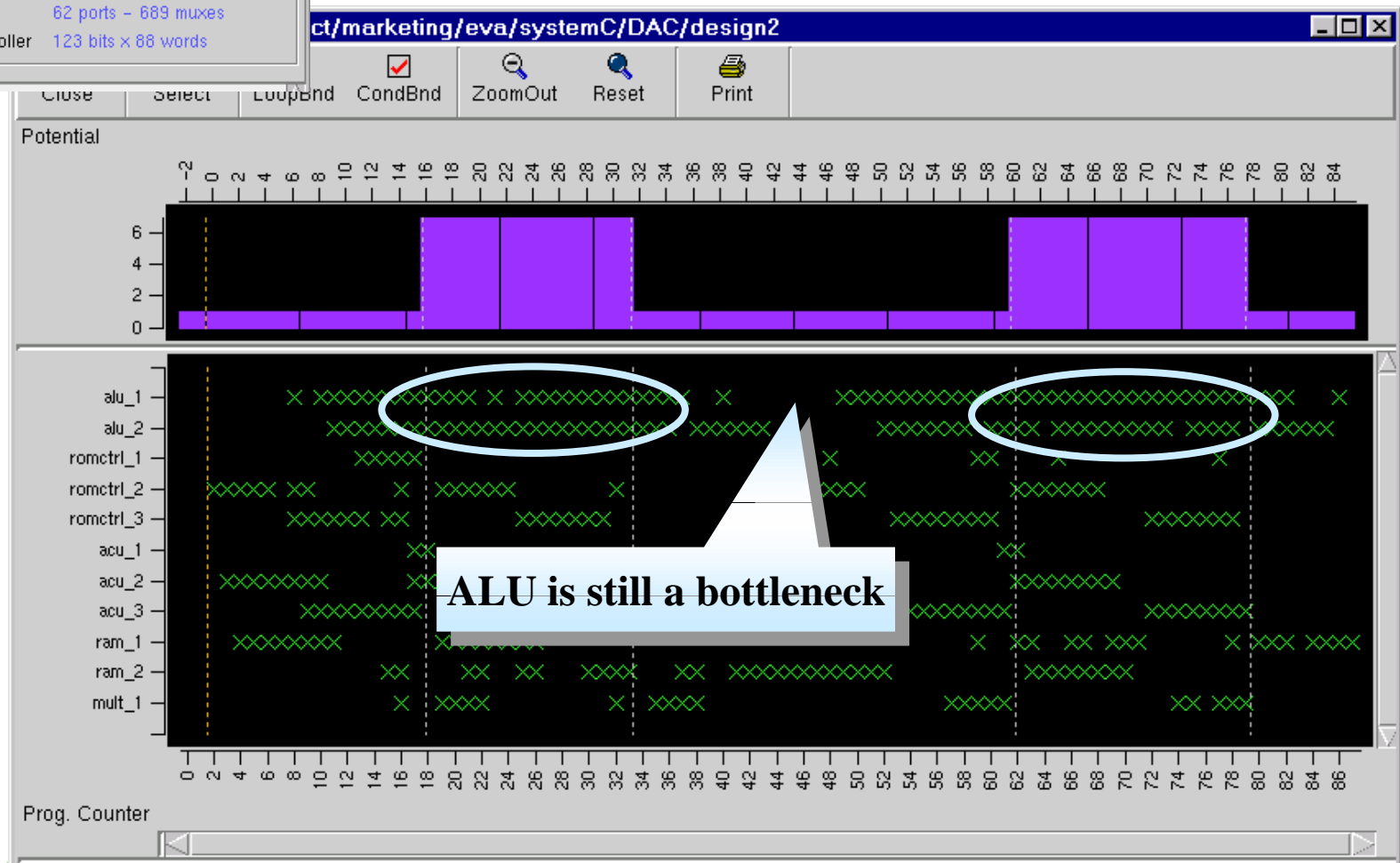
Case Study

Results

290 cycles ➡ 4 times better, but still too many

Summary

Init	0 pot - 0 cycles
Run	86 pot - 290 cycles
Registers	49 fields - 827 bits
Mux	62 ports - 689 muxes
Controller	123 bits x 88 words

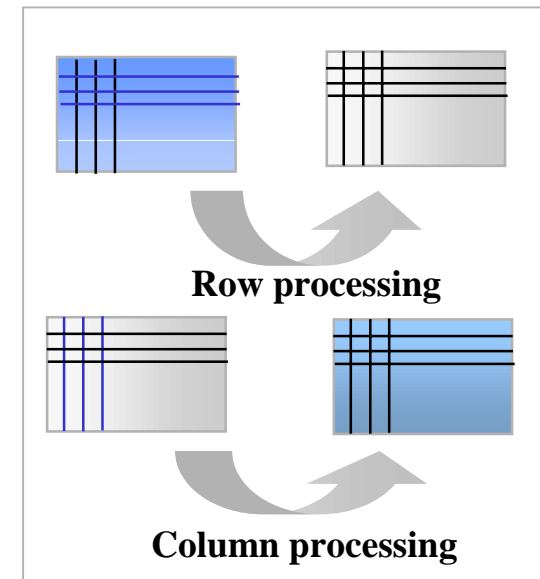


Solution:

1. User-defined core that can add and subtract in 1 cycle:

```
void add_sub1 ( DATA in1,  
               DATA in2,  
               DATA& out1,  
               DATA& out2)  
{  
  #pragma OUT out1 out2  
  out1= in1 + in2;  
  out2= in1-in2;  
}
```

2. Pipelining row and column processing:

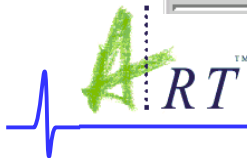
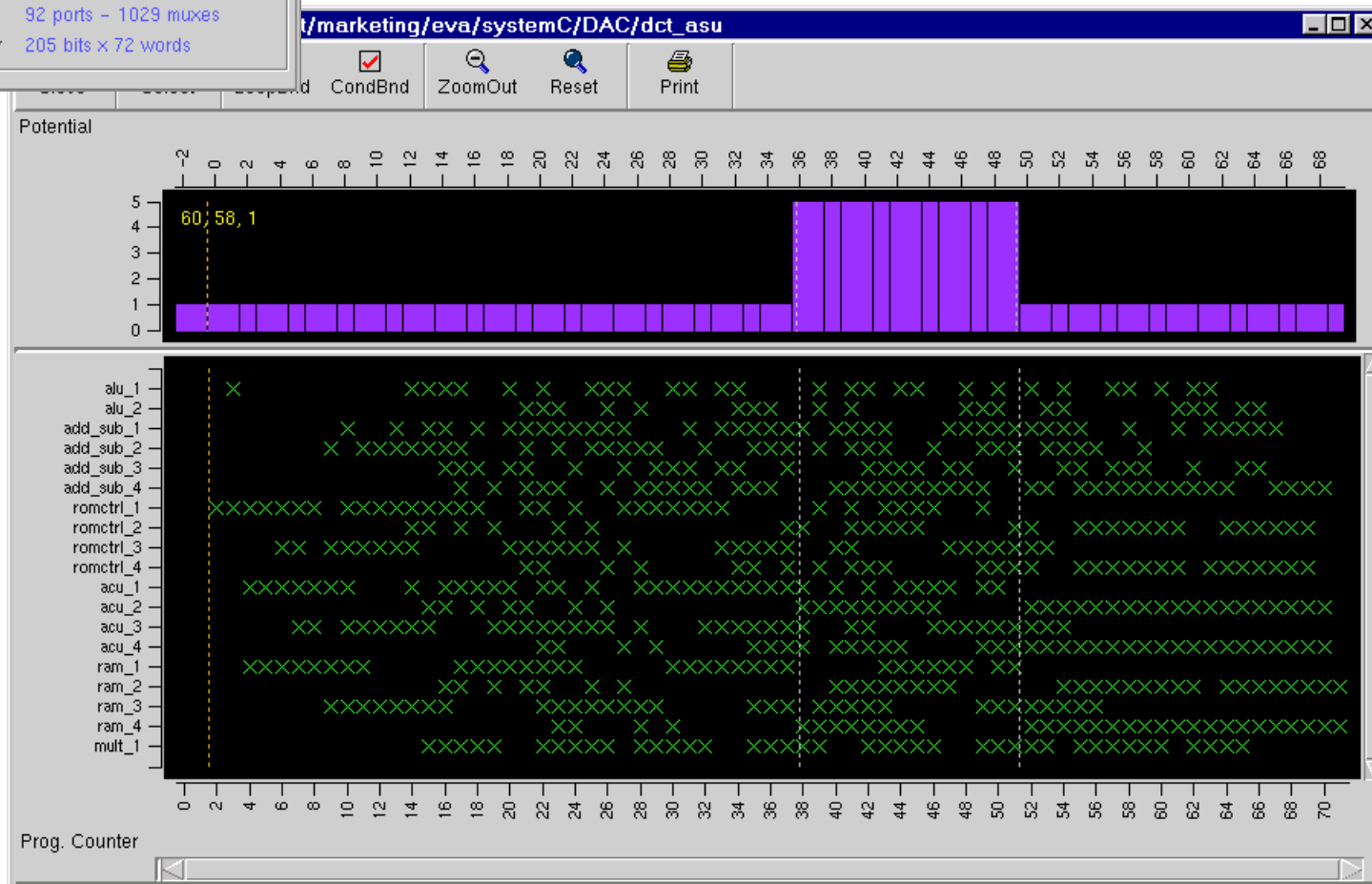


Case Study

Pipelined design :results

126 cycles \Rightarrow within the constraints

Summary	
Init	0 pot - 0 cycles
Run	70 pot - 126 cycles
Registers	80 fields - 1246 bits
Mux	92 ports - 1029 muxes
Controller	205 bits x 72 words



Summary

