Design with Shared Memory Using Catapult HLS C++

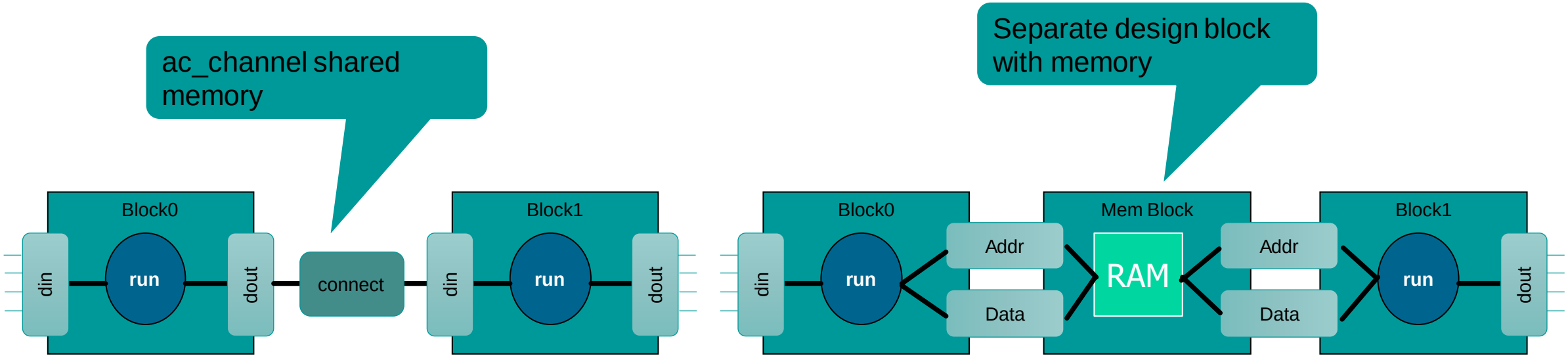
Sr. AE, Yuan-Teng Chang
yuan-teng.chang@siemens.com

SIEMENS

Shared Memories Between Design Blocks

There are two ways to share a memory between classes/member functions mapped to design blocks

- Using an ac_channel, ac_shared and required coding style
- Explicitly coding a separate design block that contains a memory



Create shared memories with ac_channel

03_Shared_Mem/ac_channel

SIEMENS

ac_channel Shared Memory Interfaces Between Blocks

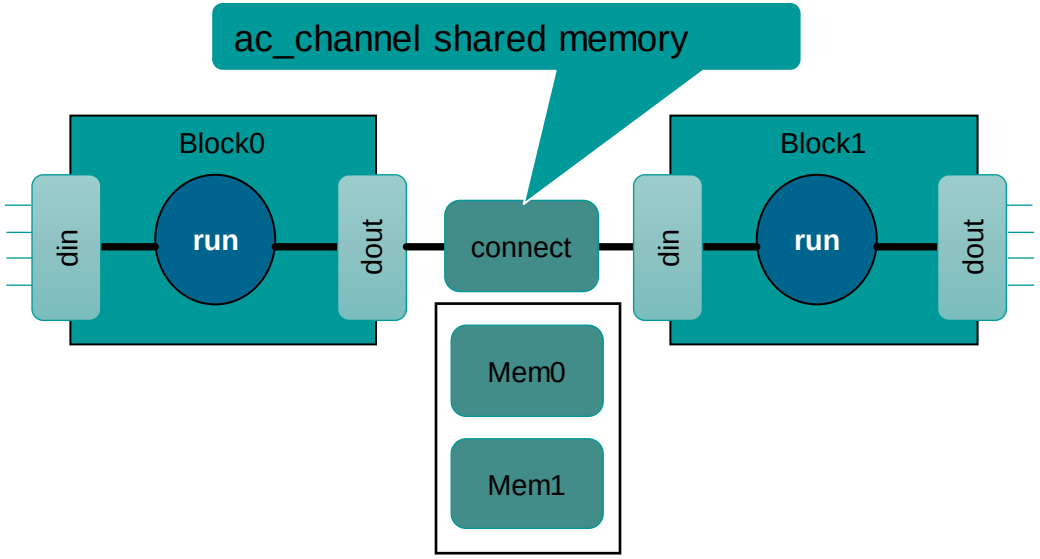
Need a way to create block-level memory interfaces

- Interface Synthesis allows arrays on the function interface to be mapped to a memory interface

Required coding style is to pack arrays inside a struct and read/write through an ac_channel

- Required style for interconnecting design blocks using shared memories
- Array operations still performed locally
- Struct/array mapped to memory using Interface Synthesis

```
typedef ac_int<8,false> dType;

struct chanStruct
{
  ac_int<8,false> data[NUM_WORDS];
};
```



Ping-pong memory

Memory Write Interfaces

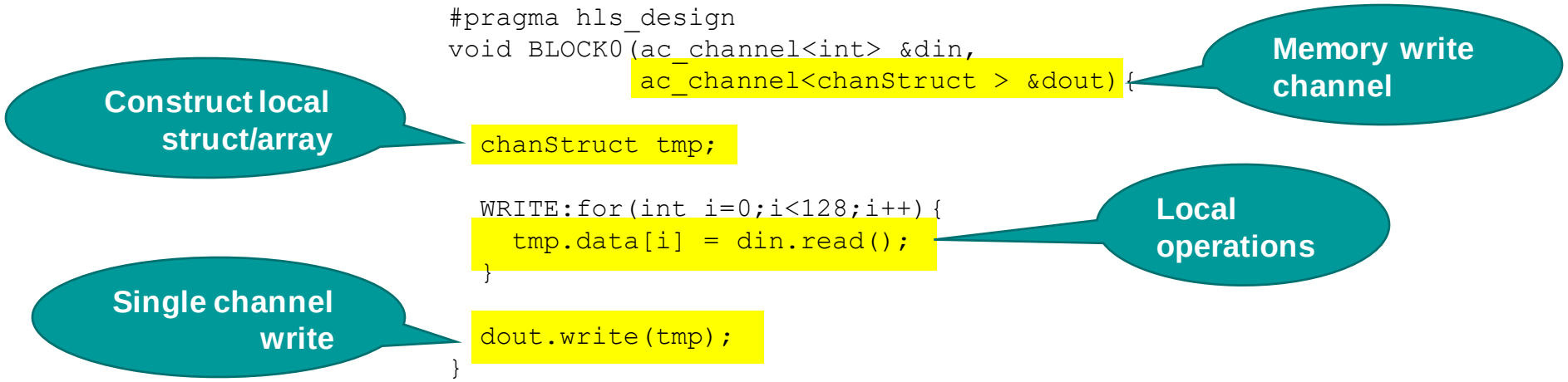
Define a “local” variable using struct type that packs the array

Operate locally on the array inside the struct

Write the struct containing the array into the channel

- Only write the channel once!

Have the HLS tool map the interface channel to a memory interface



Memory Write Interface Pitfalls

Temporary struct with array will be optimized away if coded using recommended style

- Be careful when performing memory writes
- Keep local memory operations between temporary struct declaration and memory channel write

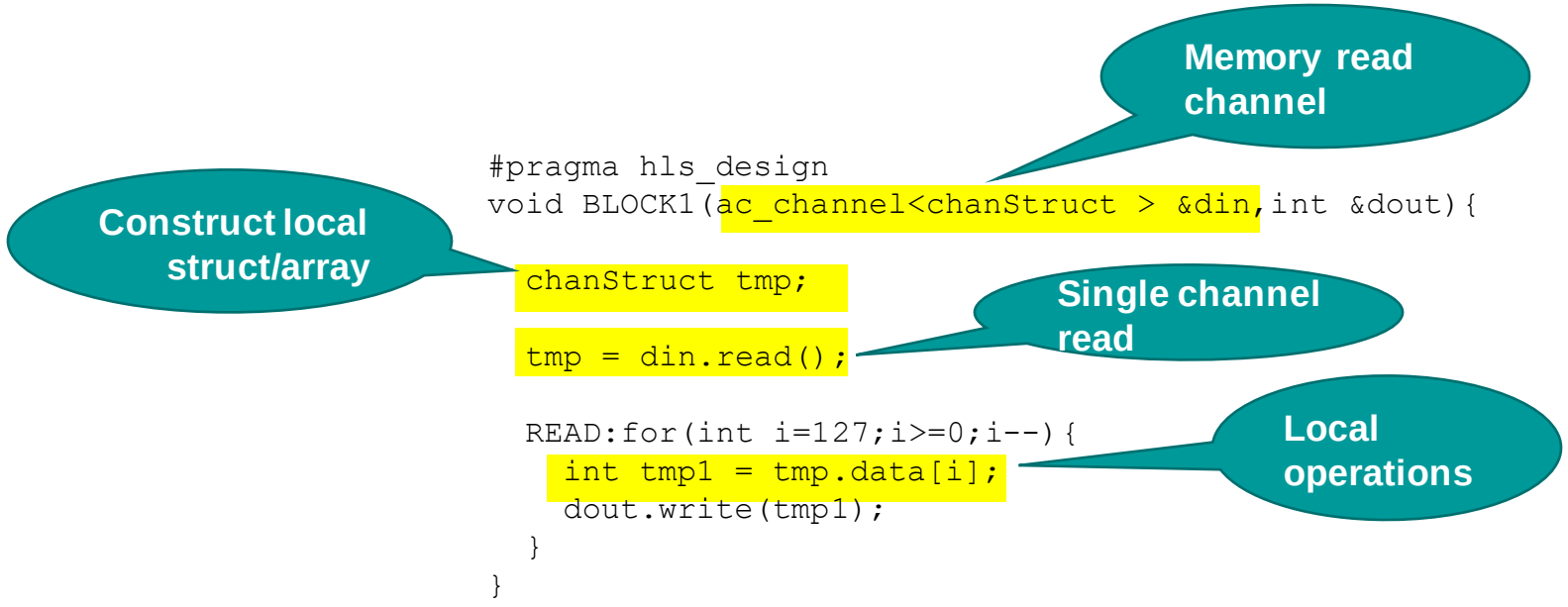
```
#pragma hls_design
void BLOCK0(ac_channel<int> &din,
            ac_channel<chanStruct > &dout){

  chanStruct tmp;

  WRITE:for(int i=0;i<128;i++){
    tmp.data[i] = din.read();
  }

  dout.write(tmp);

  …..

}
```

Memory Read Interfaces

Define a “local” variable using struct type that packs the array

Restrict operations on the local struct to be between the local struct declaration and the memory channel read

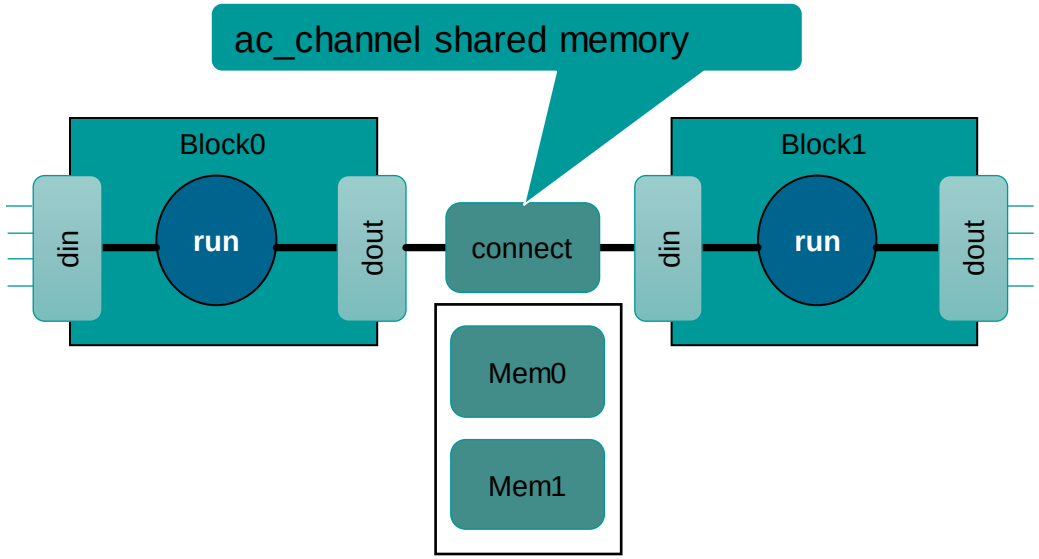
- Doing anything else will cause a local memory to be generated



ac_channel_shared_mem.h

```
20 class top
21 {
22  private:
23   ac_channel<chanStruct> shr_mem; // static memory channel
24
25   #pragma hls_design
26   void BLOCK0(ac_channel<dType> &din,
27               ac_channel<chanStruct> &dout)
28   {
29     chanStruct tmp; // temporary ac_array
30     WRITE:for (int i=0; i<NUM_WORDS; i++) { // local operations
31       tmp.data[i] = din.read();
32     }
33     dout.write(tmp); // single memory channel write
34     //Do not access the local array after this point
35   }
36
37   #pragma hls_design
38   void BLOCK1(ac_channel<chanStruct> &din,
39               ac_channel<dType> &dout)
40   {
41
42     chanStruct tmp; //temporary ac_array
43     tmp = din.read();                // single memory channel read
44     READ:for (int i=NUM_WORDS-1; i>= 0; i--) {
45       dout.write(tmp.data[i]);
46     }
47   }
```

```
49  public:
50    top(){}
51    #pragma hls_design interface
52    void CCS_BLOCK(run)(ac_channel<dType> &din,
53                        ac_channel<dType> &dout)
54    {
55     if (din.available(NUM_WORDS))
56     {
57      BLOCK0(din,shr_mem);
58      BLOCK1(shr_mem,dout);
59     }
60    }
61
62 };
```



Ping-pong memory

Testbench (tb.h)

```
4  CCS_MAIN(int argv, char **argc)
5  {
6    top dut;
7    ac_channel<dType> din_chan;
8    ac_channel<dType> dout_chan;
9
10   for (int j=0; j<2; j++)
11   {
12     fprintf(stdout, "iter = %2d start...\n", j);
13     for (int i=0; i<NUM_WORDS; i++)
14     {
15       dType dat = i+j*NUM_WORDS;
16       din_chan.write(dat);
17       fprintf(stdout, "din @%3d = %4d\n", i, (int) dat);
18     }
19
20     dut.run(din_chan, dout_chan);
21
22     int cnt = 0;
23     while(dout_chan.available(1))
24     {
25       dType dat = dout_chan.read();
26       fprintf(stdout, "dout @%3d = %4d\n", cnt++, (int) dat);
27     }
28     fprintf(stdout, "iter = %2d finish!!!\n", j);
29   }
30 }
```

Stimuli

DUT

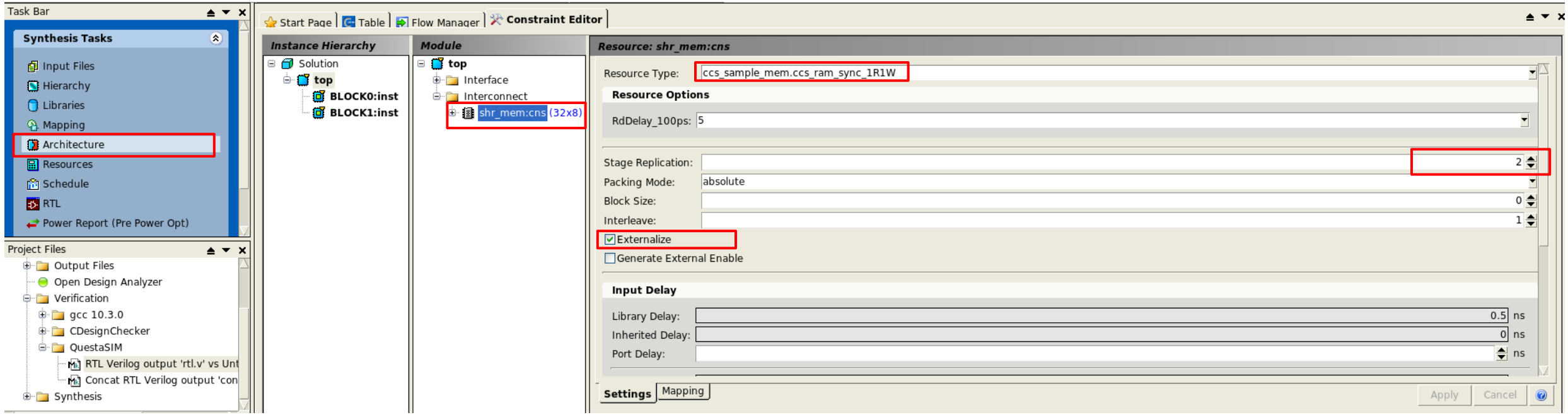
Response

Define Shared Memory Architecture

Stage Replication

- 1: a single shared memory architecture
- 2: a ping-pong architecture

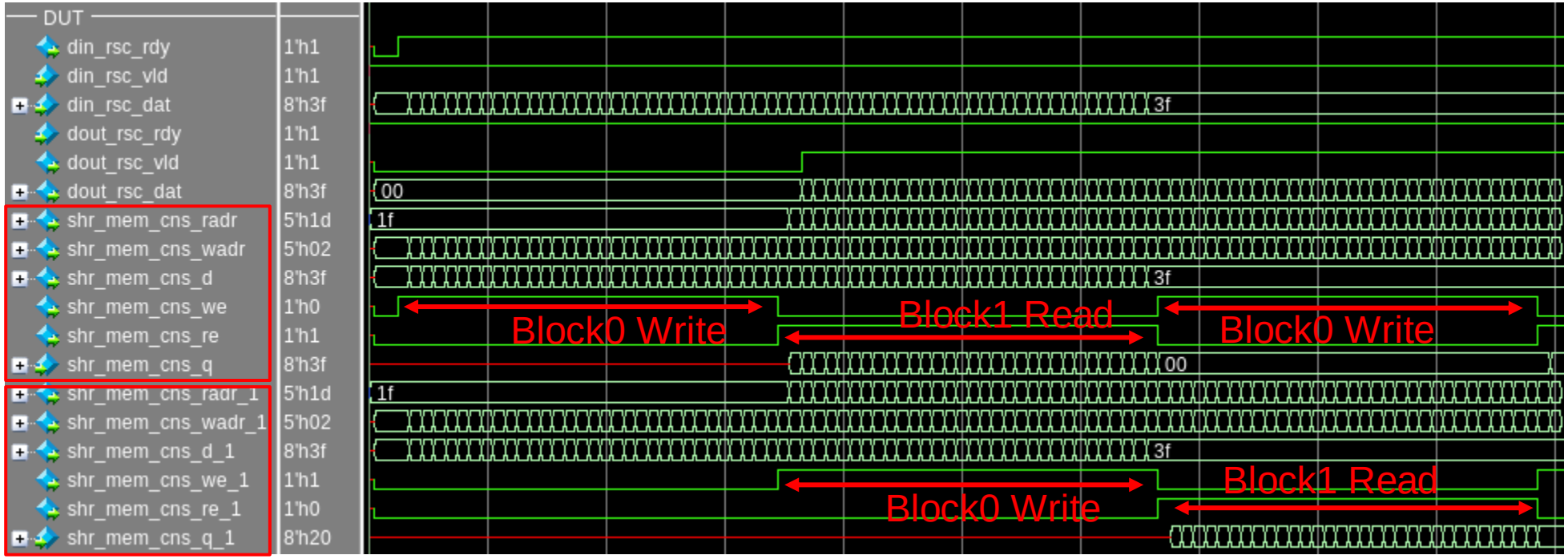
```
[mentor@RHEL74 ac_channel]$ catapult -p ultra -f directives.tcl &
```



RTL Simulation

Mem0

Mem1



Create shared memories with ac_shared

03_Shared_Mem/ac_shared

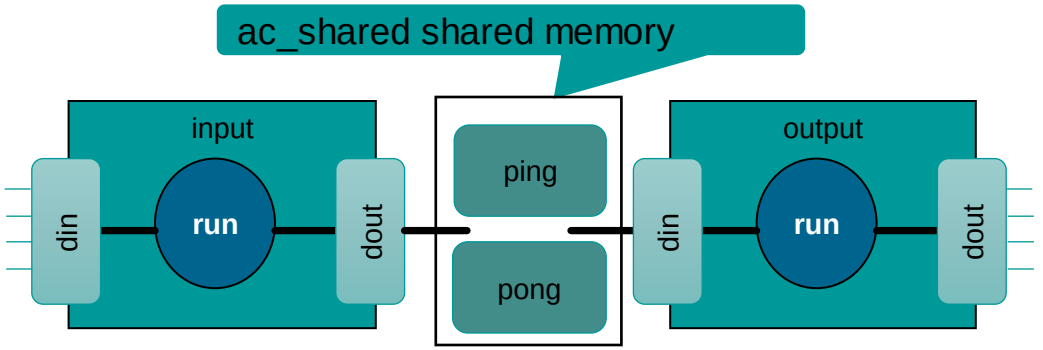
SIEMENS

ac_shared_memory.h

```
15 class input
16 {
17 public:
18   input():RAM_select(true) {}
19
20   #pragma hls_design interface
21   void run ( ac_shared<dType[NUM_WORDS]> &ping,
22              ac_shared<dType[NUM_WORDS]> &pong,
23              ac_sync &sync_RAMs,
24              ac_channel<dType> &in ) {
25     for ( int i = 0; i < NUM_WORDS; i++ ) {
26       dType data = in.read();
27       if ( RAM_select ) {
28         ping[i] = data;
29       } else {
30         pong[i] = data;
31       }
32     }
33     RAM_select = !RAM_select;
34     sync_RAMs.sync_out();
35   }
36
37 private:
38   bool RAM_select;
39 };
```

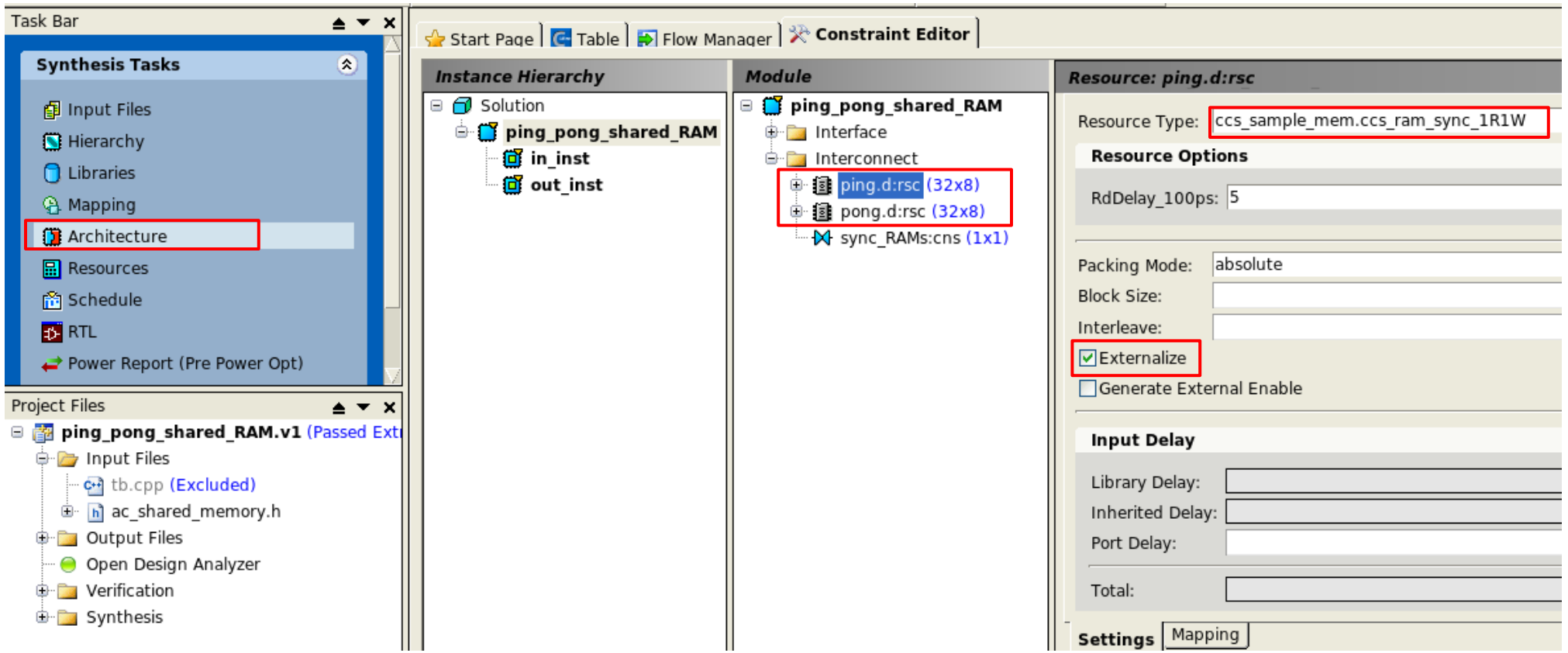
```
41 class output
42 {
43 public:
44   output():RAM_select(true) {}
45
46   #pragma hls_design interface
47   void run ( ac_shared<dType[NUM_WORDS]> &ping,
48              ac_shared<dType[NUM_WORDS]> &pong,
49              ac_sync &sync_RAMs,
50              ac_channel<dType> &out ) {
51     dType data;
52     sync_RAMs.sync_in();
53     for (int i=NUM_WORDS-1; i>= 0; i--) {
54       if ( RAM_select ) {
55         data = ping[i];
56       } else {
57         data = pong[i];
58       }
59       out.write(data);
60     }
61     RAM_select = !RAM_select;
62   }
63
64 private:
65   bool RAM_select;
66 };
```

```
68 #pragma hls_design top
69 class ping_pong_shared_RAM
70 {
71 public:
72   ping_pong_shared_RAM() {}
73
74   #pragma hls_design interface
75   void CCS_BLOCK(run) ( ac_channel<dType> &in,
76              ac_channel<dType> &out ) {
77     in_inst.run(ping,pong,sync_RAMs,in);
78     out_inst.run(ping,pong,sync_RAMs,out);
79   }
80
81 private:
82   input in_inst;
83   output out_inst;
84   ac_shared<dType[NUM_WORDS]> ping, pong;
85   ac_sync sync_RAMs;
86 };
```



Define Shared Memory Architecture

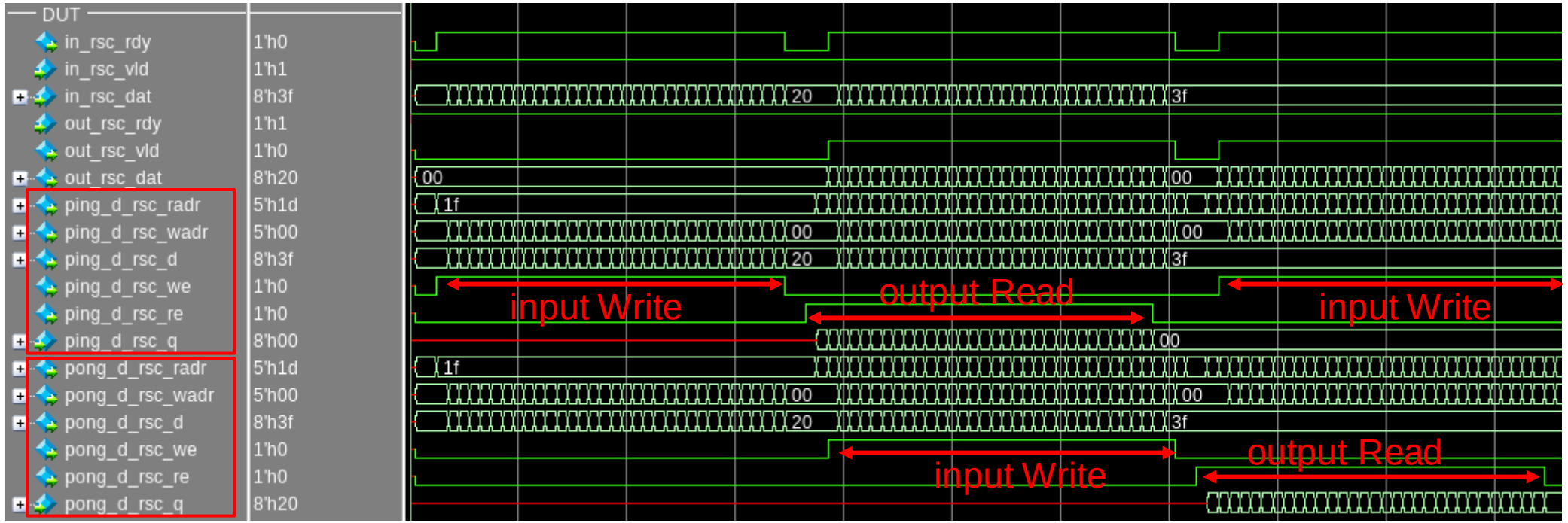
```
[mentor@RHEL74 ac_shared]$ catapult -f directives.tcl
```



RTL Simulation

Ping

Pong



Thank you

SIEMENS