Advanced SoC Group 2 & 3

Final Project Presentation

Team Member:
陳冠晰、劉祐瑋、陳昇達、王彥智、陳柏翰

Project Overview

- Why Falcon?
- System View
- Hardware Acceleration
- HW/SW Codesign
- From FPGA to Caravel-SoC
- Catapult HLS
- Synopsys Flow
- Future Work

Why Falcon ?

- Novelty and Contributions:
  - In response to the threat posed by quantum computers...
  - Implement PQC with **hardware accelerator using HLS**
  - Hardware / Software Co-design
  - Integrate to **SoC RTL source code** with **FSIC**
  - **Future tapeout...**
- Reference:
  - Falcon algorithm and software code

Has broad market applications in the future...

FSIC: a set of IPs designed for system validation)

FALCON

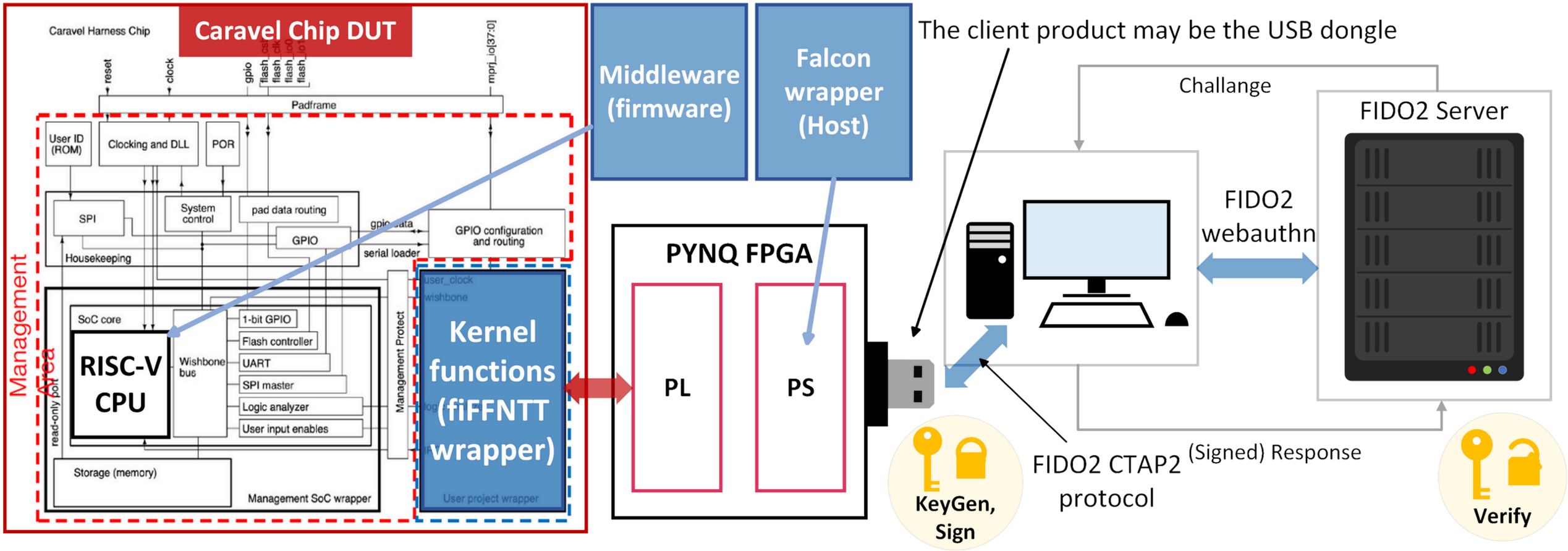
Fast-Fourier Lattice-based Compact Signatures over NTRU

Speed: FFSampling

Scalability: 256、512、1024-bit

https://falcon-sign.info

System View


Caravel Chip DUT
Caravel Harness Chip
reset
clock
gpio
mprj_io[37:0]
Padframe
User ID (ROM)
Clocking and DLL
POR
SPI
System control
pad data routing
GPIO
GPIO configuration and routing
Housekeeping
serial loader
Management Area
SoC core
RISC-V CPU
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
Management Protect
Storage (memory)
Management SoC wrapper
user_clock
wishbone
Kernel functions (fiFFNTT wrapper)
User project wrapper
Middleware (firmware)
Falcon wrapper (Host)
PYNQ FPGA
PL
PS
The client product may be the USB dongle
Challange
FIDO2 Server
FIDO2 webauthn
KeyGen, Sign
FIDO2 CTAP2 protocol
(Signed) Response
Verify

Top Down

- Falcon API -> Middleware -> Hardware Kernels

Falcon Application Code

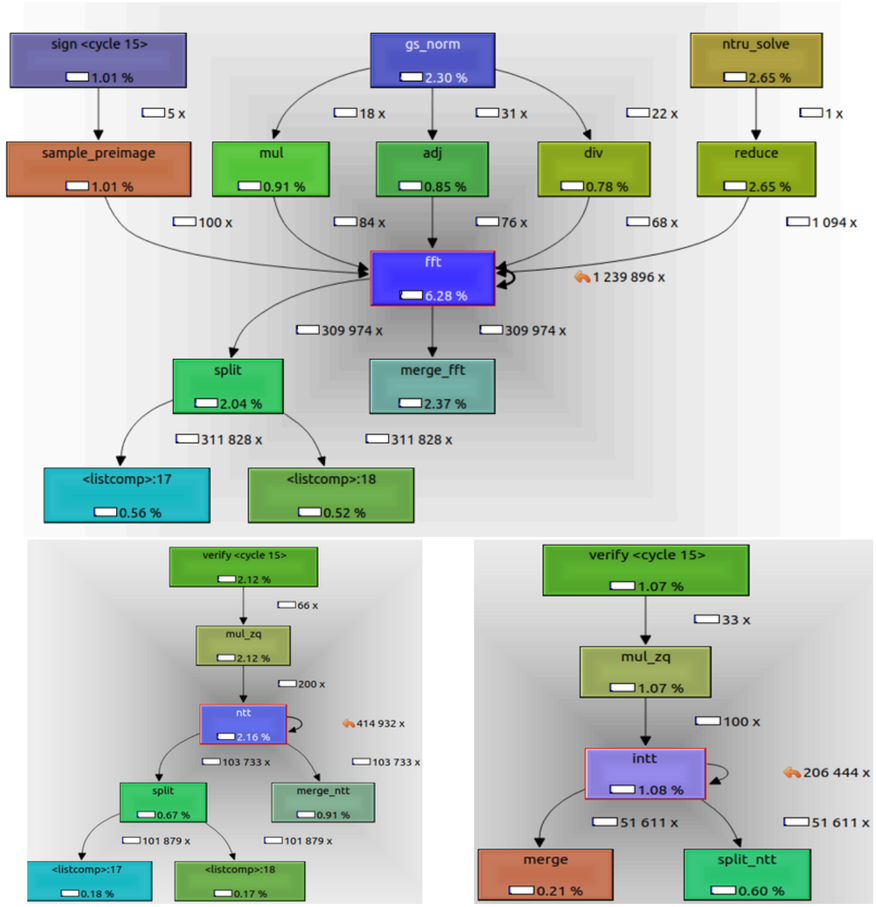
Scheduling kernel tasks

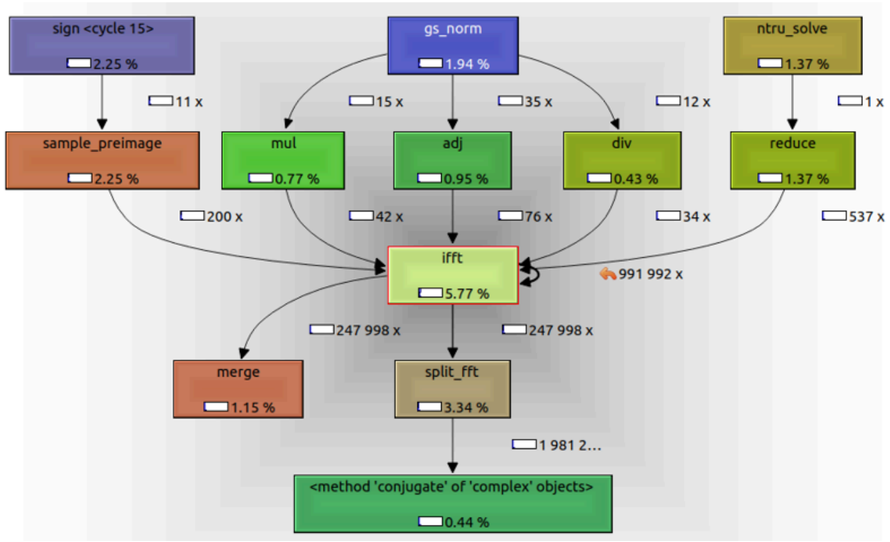
Hardware accelerators

Hardware Acceleration

Hardware Acceleration

- Identify the critical functions (kernel functions) using call graph





Forward/Inverse FFT (Fast-Fourier Transform)

Forward/Inverse NTT (Number Theoretic Transform)

Hardware Acceleration

- Initial implementation (software to HLS code)

Algorithm FFT for Falcon PQC

Inputs: Polynomial coefficients array *f*, Base-2 logarithm of the polynomial degree *logn*

Output: Transformed coefficients in array *f*

```
1: procedure FFT(f, logn)
2:   n ← 2^lon
3:   stride ← n /2
4:   for each stage u from 1 to logn-1
5:     section_size ← 2^u
6:     for each subsection i from 0 to section_size/2-1
7:     j1 ← i * stride
8:     j2 ← j1 + stride/2
9:     Retrieve twiddle factor s for the subsection from GM table
10:    for each j from j1 to j2-1
11:      Combine elements at j with twiddle factor s
12:      Store results in f
13:    end for
14:  end for
15:  stride /= 2
16: end procedure
```

Algorithm Inverse FFT for Falcon PQC

Inputs: Polynomial coefficients array *f*, Base-2 logarithm of the polynomial degree *logn*

Output: Inverse FFT-transformed coefficients in array *f*

```
1: procedure iFFT(f, logn)
2:   n ← 2^lon
3:   t ← 1
4:   m ← n
5:   for each stage from logn down to 1
6:     half_m ← m * 2
7:     double_t ← 2 * t
8:     for i1 from 0 to half_m-1
9:     j1 ← i1 * double_t
10:    Retrieve inverse twiddle factor s from iGM table for half_m+i1
11:    for j from j1 to j1+t-1
12:      Combine elements at j and j+t with s
13:      Store results in f
14:    end for
15:  end for
16:  t ← double_t
17:  m ← half_m
18:  end for
19:  if logn > 0
20:    Scale each element in f by the inverse scaling factor
21:  end if
22: end procedure
```

Algorithm NTT for Falcon PQC

Inputs: Array *a* of integers (polynomial coefficients), Base-2 logarithm of the number of coefficients *logn*

Output: Transformed coefficients in array *a*

```
1: procedure NTT(a, logn)
2:   n ← 2^lon
3:   t ← n
4:   for each stage from 0 to logn-1
5:     ht ← t / 2
6:     for i from 0 to 2^stage-1
7:       j1 ← i * t
8:       Retrieve root of unity s from the GMb table at index 2^ stage+i
9:       j2 ← j1 + ht
10:      for j from j1 to j2-1
11:        u ← a[j]
12:        v ← (a[j+ht]*s) mod Q
13:        a[j] ← (u+v) mod Q
14:        a[j+ht] ← (u-v) mod Q
15:      end for
16:    end for
17:    t ←ht
18:  end for
19: end procedure
```

Algorithm Inverse NTT for Falcon PQC

Inputs: Array *a* of integers (polynomial coefficients), Base-2 logarithm of the number of coefficients *logn*

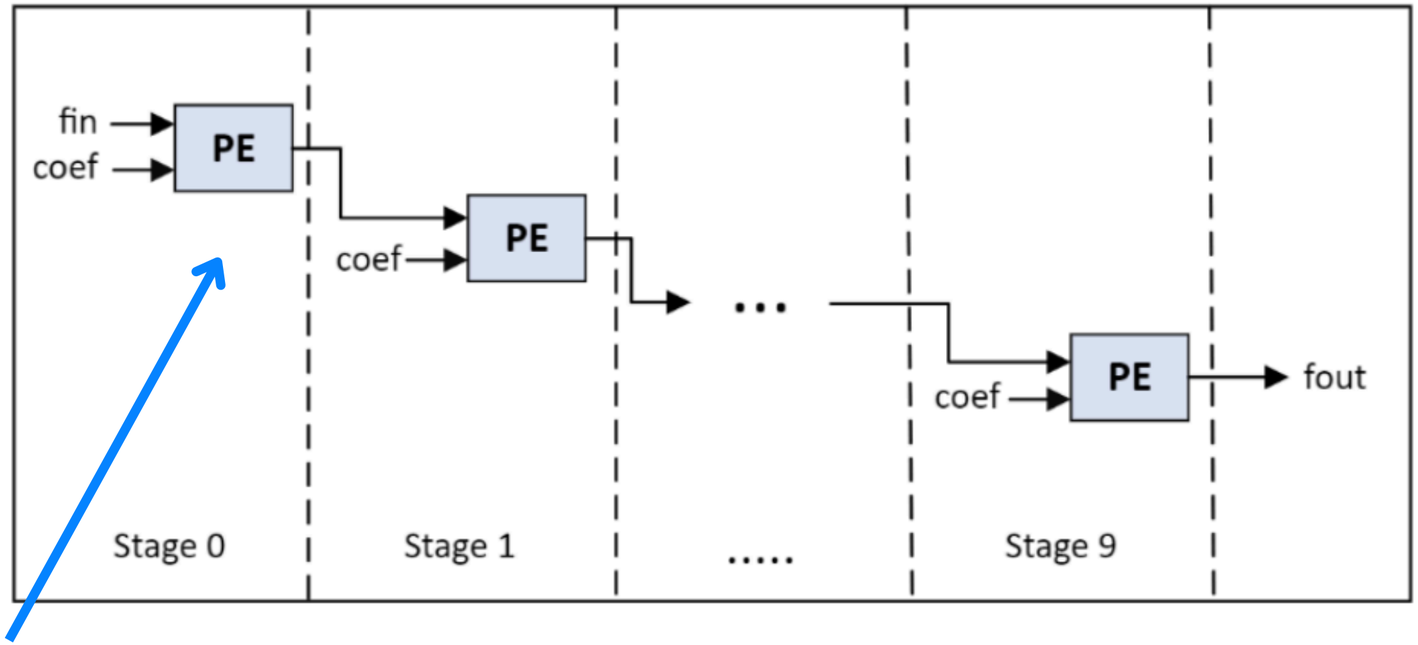
Output: Inverse transformed coefficients in array *a*

```
1: procedure iNTT(a, logn)
2:   n ← 2^lon
3:   t ← 1
4:   m ← n
5:   while m > 1
6:     hm ← m / 2
7:     dt ← t * 2
8:     for i from 0 to hm-1
9:       j1 ← i * dt
10:      Retrieve inverse root of unity s from the GMb table at index hm+i
11:      j2 ← j1 + t
12:      for j from j1 to j2-1
13:        u ← a[j]
14:        v ← a[j+ht]
15:        a[j] ← (u+v) mod Q
16:        a[j+ht] ← ((u-v)*s) mod Q
17:      end for
18:    end for
19:    t ←dt
20:    m ← hm
21:  end while
22:  Normalize each element in a using a precomputed factor ni
23: end procedure
```

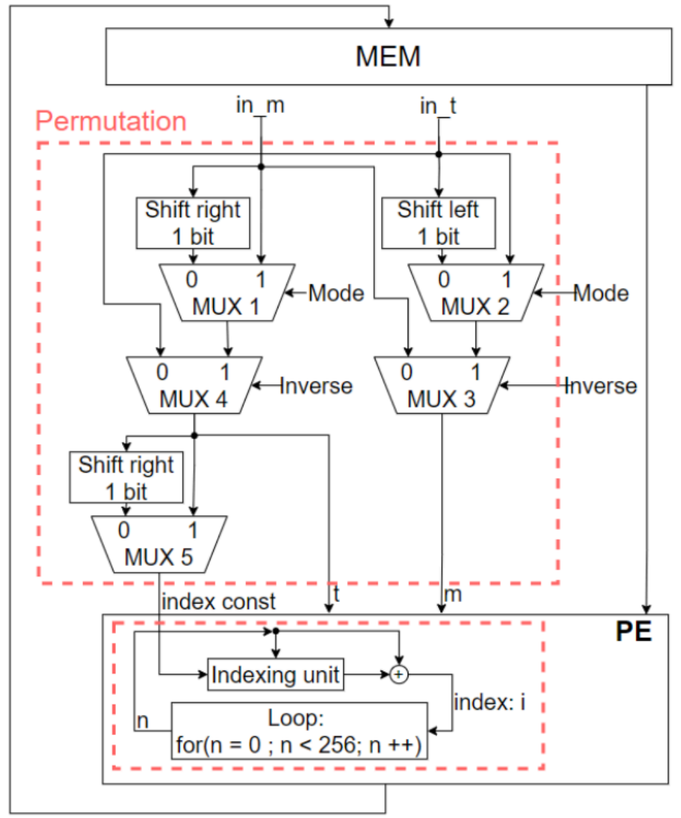
Hardware Acceleration

Decision: Each kernel has 1 PE
Integrate multi-kernels

- Custom permutation algorithm / Datapath Restructure
  -> Synthesizable (multi(logN)-stage iteration structure, bounded loop)



Reuse the same PE (Processing Element)



Latency: ~120k cycles -> ~10k cycles

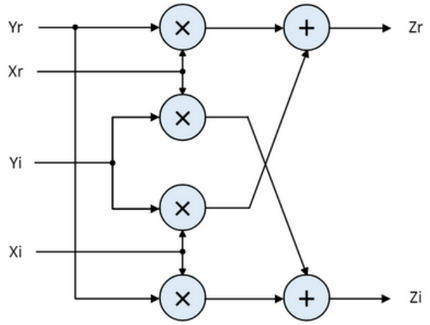
Hardware Acceleration

- Optimization forward/inverse FFT&NTT
  - **Combine 4 algorithms into a single hardware (fiFFNTT)**

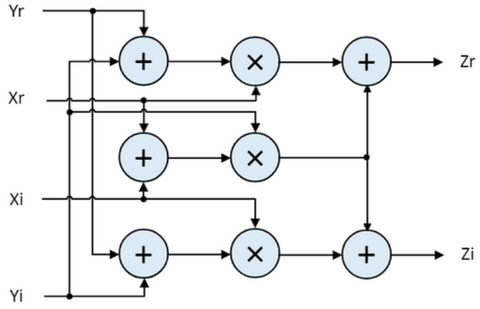
Resource	BRAM	DSP	FF	LUT
FFT	40	61	8414	11456
IFFT	80	62	13140	13027
NTT	5	30	4016	7278
INTT	5	31	4787	7594
fiFFNTT	22	56	14423	12483

Support 4 algorithms with only a little resource overhead

  - Complex multiplication - reduce the usage of multiplier



$$Z_r = X_r \cdot Y_r - X_i \cdot Y_i$$
$$Z_i = X_r \cdot Y_i + X_i \cdot Y_r$$

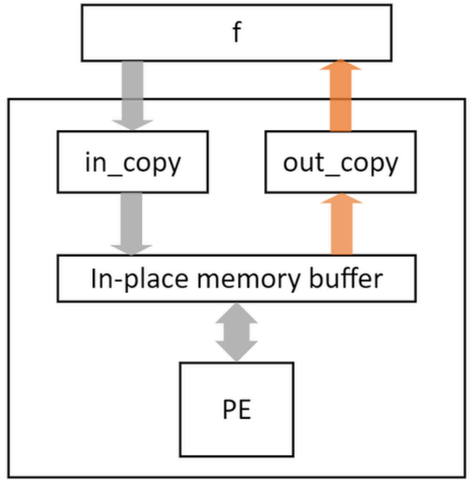


$$Z_r = X_r \cdot (Y_r - Y_i) + Y_i \cdot (X_r - X_i)$$
$$Z_i = X_i \cdot (Y_r + Y_i) + Y_i \cdot (X_r - X_i)$$

4 multiplier => 3 multiplier

Hardware Acceleration

- Optimization
  - Sharing Buffers



  - Share Memories

63	.....	48	47	.....	32	31	.....	16	15	.....	0
64-bits double											
16-bits unsigned integer				16-bits unsigned integer			16-bits unsigned integer			16-bits unsigned integer	

Hardware Acceleration Flow

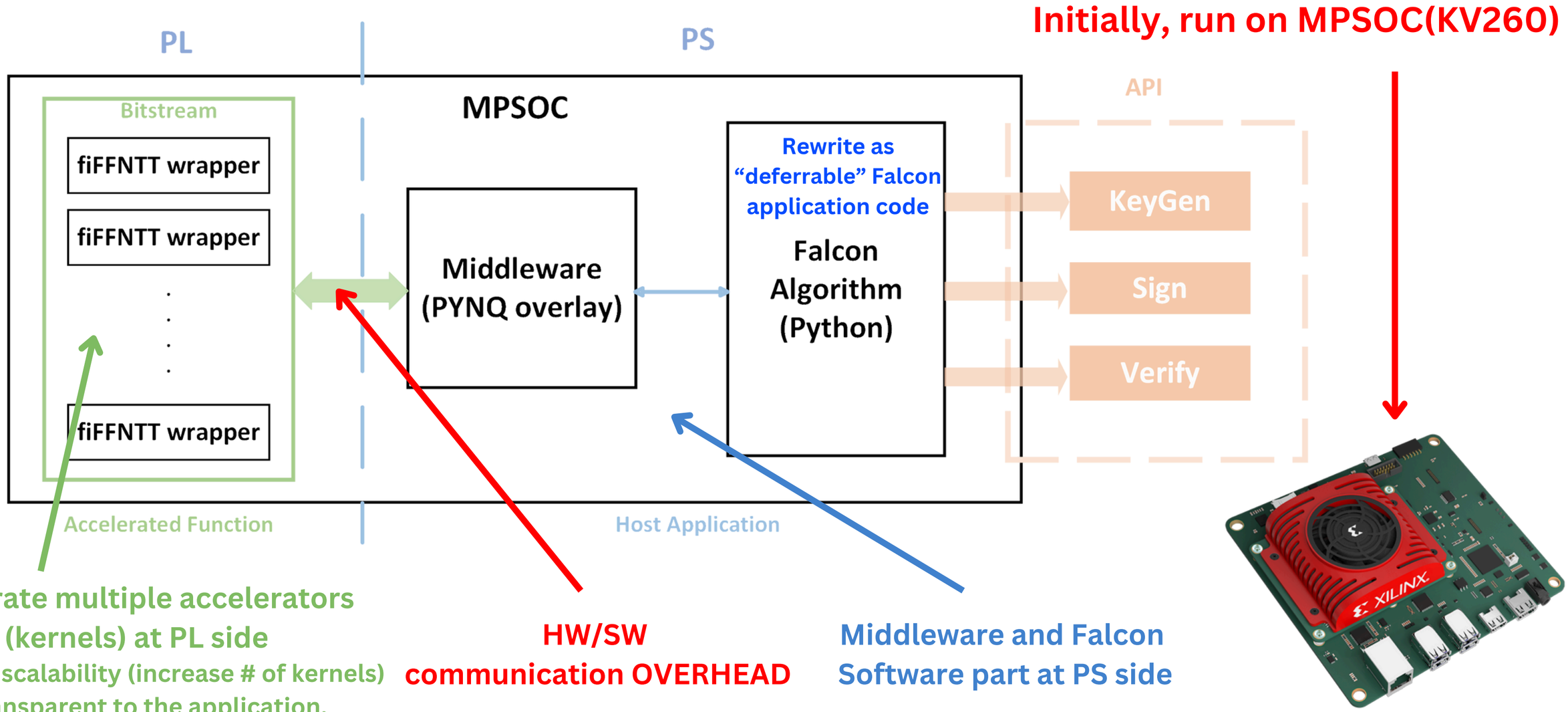
- Skeleton Restructure: PE
- Ver0 : Integration four algorithms in one hardware, Replace the multiplication of iFFT to ‘double shifter’,
- Ver1 : Using only one buffer for in/out (inplace buffer)
- Ver2 : sharing the multiplier and the adder of FFT/iFFT/NTT/iNTT

Execution time performance

Function	FFT	iFFT	NTT	iNTT
Python	28.3617	29.9833	32.8956	34.3557
Initial HLS	1.7429	2.2315	3.3797	4.3449
Skeleton Restructure	0.1034	0.2125	0.1255	0.1771
fiFFNTT Ver0	0.2223	0.2285	0.2633	0.2618
fiFFNTT Ver1	0.1419	0.1380	0.1807	0.1753
fiFFNTT Ver2	0.1372	0.1631	0.1951	0.1773

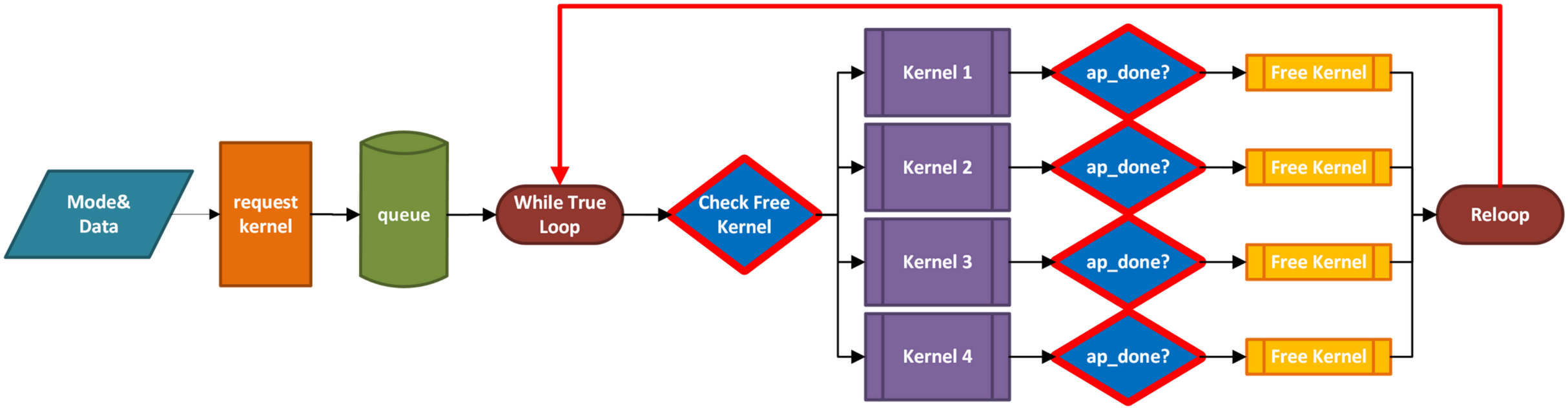
HW/SW Co-design

Hardware / Software Co-design (1/2)



Hardware / Software Co-design (2/2)

- Hardware/Software communication overhead comes from **middleware**:



PS side must continuously monitor the status of the kernel on the PL side.

=> move middleware to PL side.

Performance

Performance

- Hardware kernels compare to original software

Execution time (Unit: ms / execution)

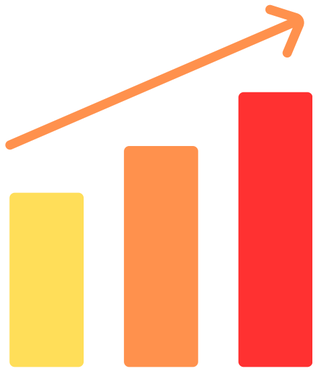
Function	FFT	iFFT	NTT	INTT
Python	28.3617	29.9833	32.8956	34.3557
Initial HLS	1.7429	2.2315	3.3797	4.3449
fiFFNTT Ver2	0.1372	0.1631	0.1951	0.1773
Speed up rate	207	184	170	194

Speed Up x200

- HW/SW co-design compare to original software

Final speed (Unit: ms)

Flow	KeyGen	Sign	Verify
Original software	100.4	2053.3	139.9
HW/SW with middleware	18.9	747.5	60.7



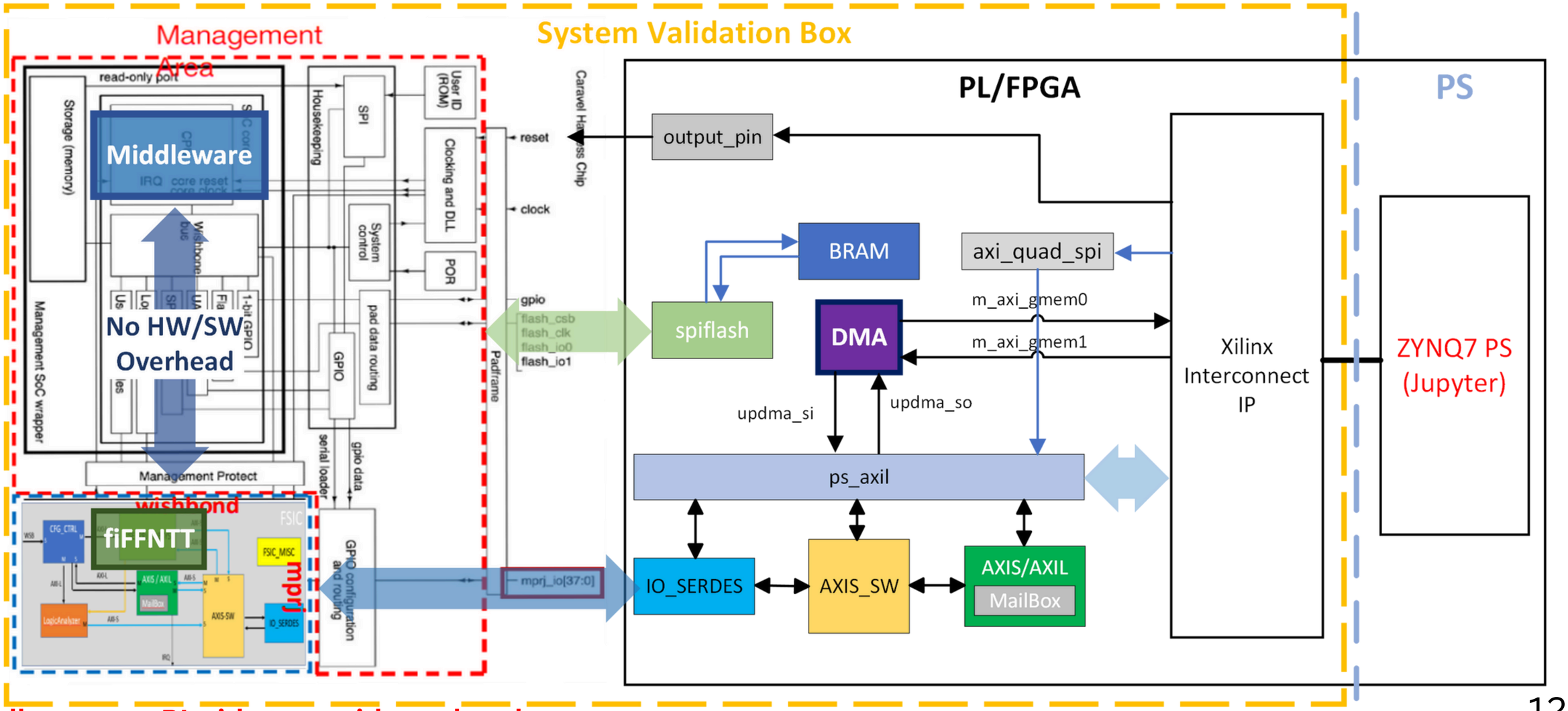
x2~5

From FPGA to Caravel-SoC

Purpose

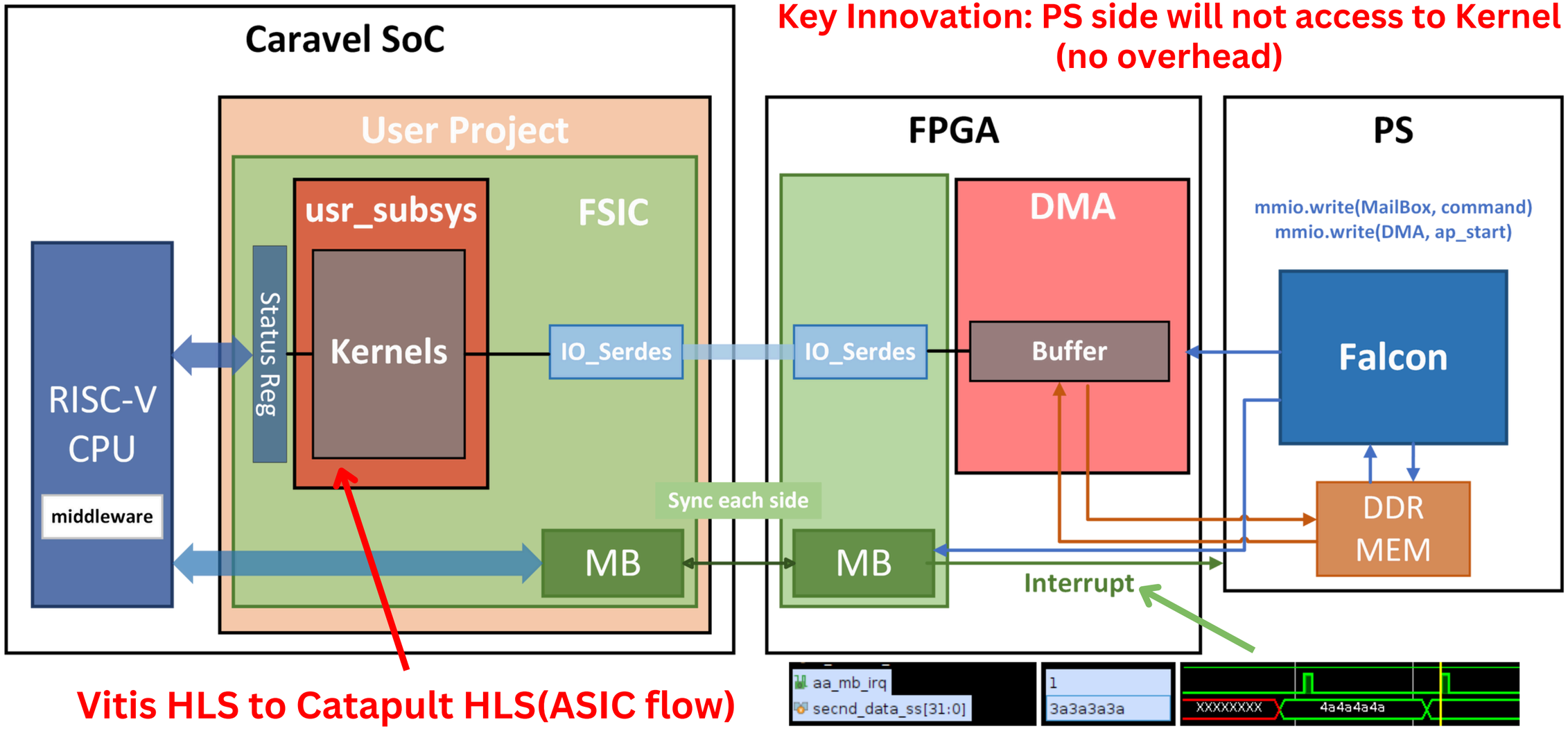
We then placed the kernels into the user project on the Caravel SoC side and write the middleware as firmware to be executed by the RISC-V CPU on the SoC side. By this implementation, we aims to solve the **PL-PS Communication overhead**.

Current Block Diagram



middleware on PL side to avoid overhead

Brief Block Diagram

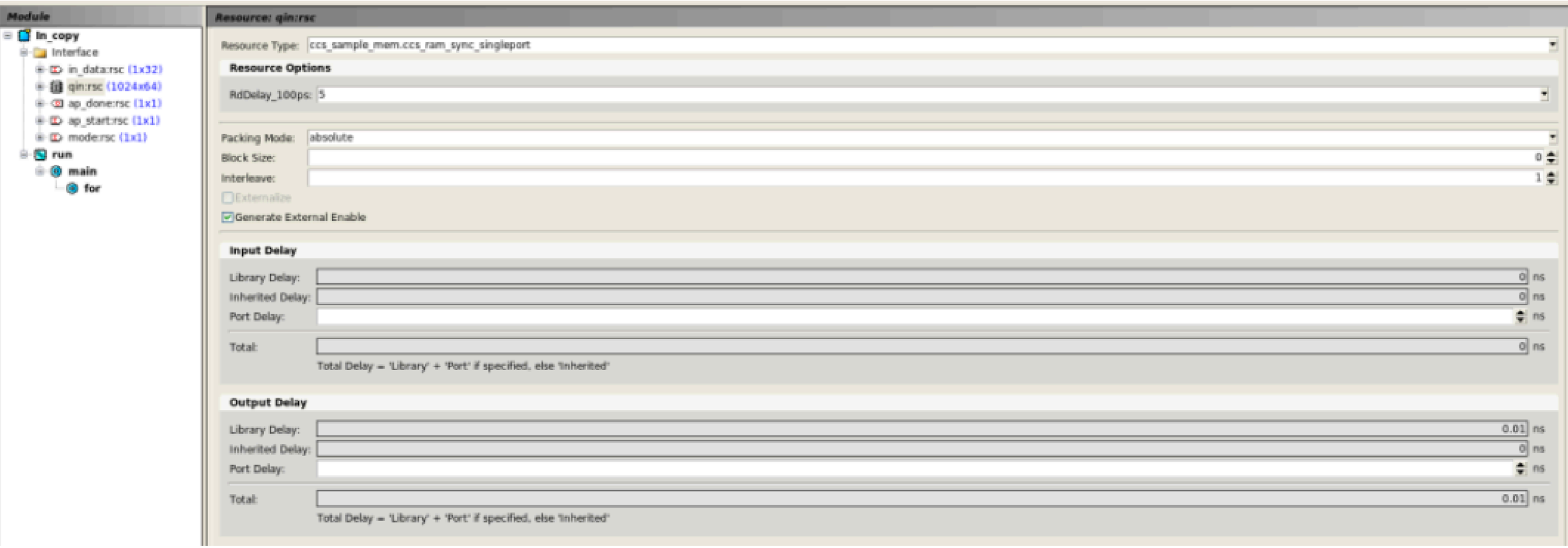


Catapult HLS-fiFFNTT

In_copy

```
class In_copy{
public:
        In_copy(){}
        #pragma hls_design interface
    void CCS_BLOCK(run) (ac_channel<ac_int<32, false>> &in_data, ac_int<64, false>qin[1024],
       ac_channel<bool> &ap_done,ac_channel<bool> &ap_start,bool &mode) {
        bool start;
        start=ap_start.read();
        ac_int<64, false>tmp1;
        for (int x=0;x<1024;x++){
                if(mode){
                ac_int<32, false> tmp;
                tmp =in_data.read();
                tmp1.set_slc(0,tmp);
                qin[x]=tmp1;}
                else {
        ac_int<32, false> tmp;
                tmp =in_data.read();
                tmp1.set_slc(0,tmp);
                tmp =in_data.read();
                tmp1.set_slc(32,tmp);
                qin[x]=tmp1;
                }
        }
        ap_done.write(1);
        }
};
```

Setting Externel RAM



Synthesis result: ln_copy

Component Name	Area Score	Area(Combinational)	AreaSeq	Delay Post Alloc	Post Assign
TOTAL AREA (After Assignment):	692.345	257.000	426.000		

Operator useage for add

Operator Bitwidth Summary

Operation	Size (bits)	Count
add		
-	11	1

fiFFNTT

```
9       void CCS_BLOCK(run) (ac_channel<bool> &ap_start,ac_channel<bool> &ap_done,u16 &mode1,Stream1 in[1024],
10  Stream1 out[1024],ac_channel<Stream1> &out1) {
11          bool start;
12          start=ap_start.read();
17                  if(mode1==0){
18                          mode=1;
19                          inverse=0;
20                          t_in=512;
21                          m_in=2;
22                  }
23                  else if (mode1==1){
24                          mode=1;
25                          inverse=1;
26                          t_in=512;
27                          m_in=2;
28                  }
29                  else if (mode1==2){
30                          mode=0;
31                          inverse=0;
32                          t_in=1024;
33                          m_in=1;
34                  }
35                  else{
36                          mode=0;
37                          inverse=1;
38                          t_in=1024;
39                          m_in=1;
40                  }
```

fiFFNTT

```
41          for(int i=0;;i++){
42              if(i%2){
43                  PE(t_in,m_in,mode,inverse,out,in);
44              }
45              else{
46                  PE(t_in,m_in,mode,inverse,in,out);
47              }
48             t_in=t_in>>1;
49             m_in=m_in<<1;
50             if(t_in==1){
51                 break;
52              }
53          }
```

fiFFNTT

```
57          for(int x=0;x<1024;x++){
58                      if(mode1==0){ // fft
59                                  out1.write(out[x]);
60                              }
61                              else if (mode1==1) { // ifft
62                      out_data.f=out[x].f* 0.00195312500;
63                                  out1.write(out_data);
64                              }
65                              else if(mode1==2){ // ntt
66                      out1.write(in[x]);
67                              }
68                              else{ // intt
69                                      monty_mul(&tmp3,in[x].u,64);
70                                      in[x].u=(u16)tmp3;
71                                      out1.write(in[x]);
72                              }
73
74                      }
75            ap done.write(1);
```

fiFFNTT

```
  //typedef   ac_fixed<54,2,true> fpr;
  typedef   ac_ieee_float<binary64> fpr;
  typedef   ac_int<16, false> u16;
  typedef   ac_int<32, false> u32;
struct Stream1{
    fpr f;
    u16 u;
};
```

Synthesis result: fiFFNTT

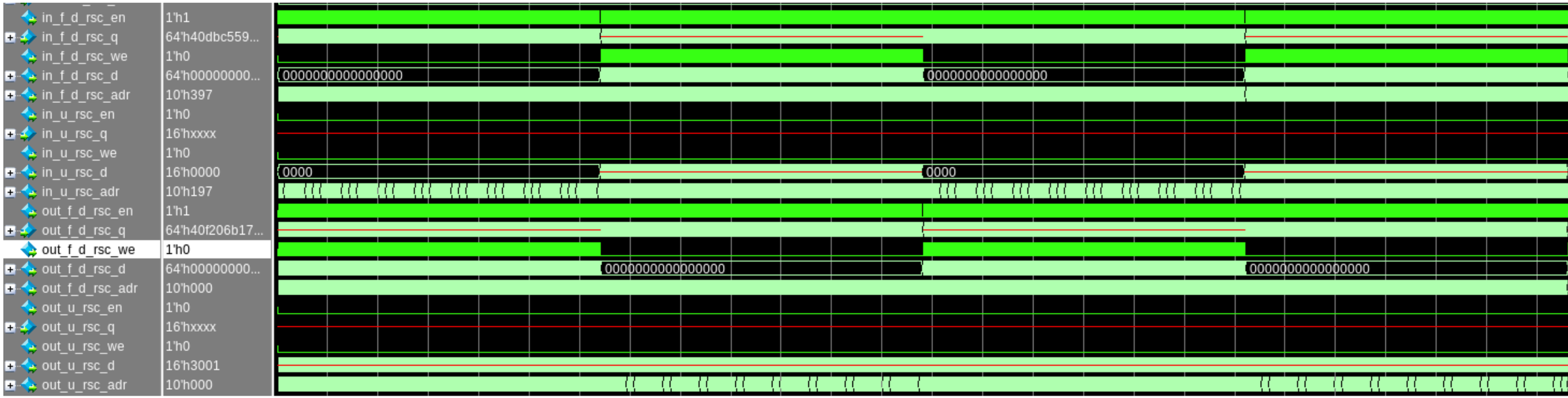
Component Name	Area Score	Area(Combinational)	AreaSeq	Delay	Post Alloc	Post Assign
TOTAL AREA (After Assignment):	71106.291	65451.000	6187.000			
mgc_nand(1,2,4)	0.698	0.000	0.000	0.044	0	54

Operator useage for add and div

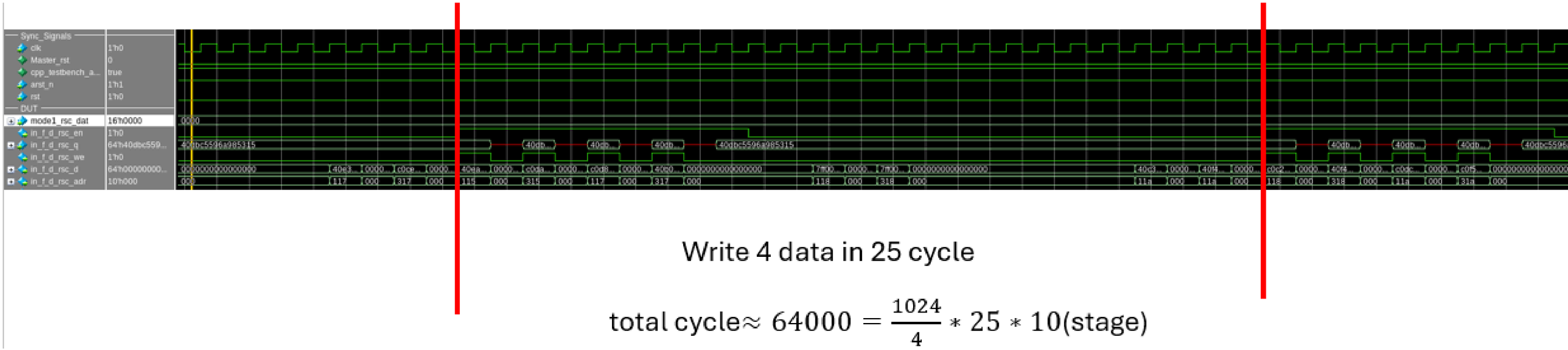
Operator Bitwidth Summary

Operation	Size (bits)	Count
add		
-	6	11
-	57	1
-	54	1
-	53	8
-	32	1
-	18	4
-	17	2
-	16	2
-	15	2
-	13	9
-	12	26
-	11	8
-	10	4
div		
-	16	1

Cosim Result(1/2)



Cosim Result(2/2)



Write 4 data in 25 cycle

$$\text{total cycle} \approx 64000 = \frac{1024}{4} * 25 * 10(\text{stage})$$

Integrate AP in FSIC

FSM

```
always@(*)begin
  case(state)
    Command:
        if(ss_tvalid && ss_tdata[3:2]==2'b01) next_state = IN_COPY;
                else next_state = Command;
    IN_COPY:
        if(In_copy_done) next_state = OUT_COPY;
                else next_state = IN_COPY;// should change
    OUT_COPY:
        if(Out_copy_done) next_state=RESET;
                else next_state=OUT_COPY;
    RESET:
        if(awvalid_in && wvalid_in &&(awaddr[11:0] == 12'h000)&&(wdata[0]==1)) next_state=Command;
        else next_state = RESET;
    default:next_state = Command;
  endcase
end
```

In_copy

```
In_copy In_copy (
  .clk(axi_clk),
  .rst(reg_rst_incpopy),
  .arst_n(axi_reset_n),
  .in_data_rsc_dat(ss_tdata),
  .in_data_rsc_vld(In_vld), //I
  .in_data_rsc_rdy(In_rdy),
  .qin_rsc_adr(Inram_adr),
  .qin_rsc_d(Inram_d),
  .qin_rsc_we(Inram_we),
  .qin_rsc_q(),
  .qin_rsc_en(Inram_en),
  .qin_triosy_lz(),
  .ap_done_rsc_dat(),
  .ap_done_rsc_vld(In_copy_done),
  .ap_done_rsc_rdy(1'b1),
  .ap_start_rsc_dat(1'b1),
  .ap_start_rsc_vld(state==IN_COPY),
  .ap_start_rsc_rdy(),
  .mode_rsc_dat(reg_mode1_in==2||reg_mode1_in==3)
);
```

```
assign ss_tready=(state==Command)?1'b1:(state==IN_COPY)?In_rdy:1'b0;
assign In_vld=(state==IN_COPY)?ss_tvalid:1'b0;
```

RAM

O

I

fiFFNTT

```
418     fiFFNTT fiFFNTT(
419     .clk(axi_clk),          // I
420     .rst(reg_rst),          // I
421     .arst_n(axi_reset_n),   // I
422     .ap_start_rsc_dat(1'b1),// I
423     .ap_start_rsc_vld(state==OUT_COPY),    // I
424     .ap_start_rsc_rdy(),    // O
425     .ap_done_rsc_dat(),     // O
426     .ap_done_rsc_vld(Out_copy_done), // O
427     .ap_done_rsc_rdy(1'b1),     // I
428     .mode1_rsc_dat(reg_mode1_in),  //I 16
429     .mode1_triosy_lz(),
```

```
430     .in_f_d_rsc_adr(in_ramf_adr),  // O 10
431     .in_f_d_rsc_d(in_ramf_d),    // O 64
432     .in_f_d_rsc_we(in_ramf_we),   // O 1
433     .in_f_d_rsc_q(in_ramf_q),    // I 64
434     .in_f_d_rsc_en(in_ramf_en),   // O 1
435     .in_f_d_triosy_lz(),
436     .in_u_rsc_adr(in_ramu_adr), // O 10
437     .in_u_rsc_d(in_ramu_d),   // O 16
438     .in_u_rsc_we(in_ramu_we),  // O
439     .in_u_rsc_q(in_ramu_q),   // I 16
440     .in_u_rsc_en(in_ramu_en),  // O
441     .in_u_triosy_lz(),
442     .out_f_d_rsc_adr(out_ramf_adr),
443     .out_f_d_rsc_d(out_ramf_d),
444     .out_f_d_rsc_we(out_ramf_we),
445     .out_f_d_rsc_q(out_ramf_q),
446     .out_f_d_rsc_en(out_ramf_en),
447     .out_f_d_triosy_lz(),
448     .out_u_rsc_adr(out_ramu_adr),
449     .out_u_rsc_d(out_ramu_d),
450     .out_u_rsc_we(out_ramu_we),
451     .out_u_rsc_q(out_ramu_q),
452     .out_u_rsc_en(out_ramu_en),
453     .out_u_triosy_lz(),
```

Next page explain

```
454     .out1_rsc_dat(Out_data),//O,80 bit{16'b,64'b},
455     .out1_rsc_vld(Out_vld),//O;
456     .out1_rsc_rdy(Out_rdy)
```

→ RAMs

sm channel (1/2)

```
always@(*)begin
  case(Out_state)
    Command:
        if(ss_tvalid && ss_tdata[3:2]==2'b01 && (ss_tdata[1:0]==2'd0 || ss_tdata[1:0]== 2'd1))  next_Out_state = F_WAIT1;
        else if(ss_tvalid && ss_tdata[3:2]==2'b01 && (ss_tdata[1:0]==2'd2 || ss_tdata[1:0]==2'd3))  next_Out_state = U_OUT;
        else next_Out_state=Command;

    F_WAIT1:
        if(Out_copy_done)next_Out_state = Command;
                        else if(Out_vld)next_Out_state = F_OUT1;
        else next_Out_state = F_WAIT1;

    F_OUT1:
        if(sm_tready) next_Out_state = F_OUT2;
        else next_Out_state = F_OUT1;

    F_OUT2:
        if(sm_tready) next_Out_state = F_WAIT1;
        else next_Out_state = F_OUT2;

    U_OUT:
        if(Out_copy_done)next_Out_state = Command;
        else next_Out_state = U_OUT;

    default:next_Out_state=Command;
  endcase
end
```

sm channel (2/2)

```
wire [79:0] Out_data;

always @(posedge axi_clk or negedge axi_reset_n)  begin
  if ( !axi_reset_n ) begin
                regx_data <= 32'b0;
                regy_data <= 32'b0;
  end
  else begin
        if(Out_state==F_WAIT1&&Out_vld)begin
                        regx_data <= Out_data[31:0];
                        regy_data <= Out_data[63:32];
        end
        else begin
        end
  end
end

/********** sm_tlast ***********/
assign sm_tlast  = (reg_mode1_in[1])  ? ((tlast_counter==11'd1023&&sm_tready&&Out_vld) ? 1 : 0) : ((tlast_counter==11'd2047&&sm_tready&&Out_state==F_OUT2) ? 1 : 0);
assign sm_tvalid = (Out_state==U_OUT) ? Out_vld : (Out_state==F_OUT1||Out_state==F_OUT2);
assign sm_tdata  = (Out_state==U_OUT) ? {16'b0,Out_data[79:64]} : ((Out_state==F_OUT1)?regx_data:regy_data);
assign Out_rdy   = (Out_state==U_OUT||Out_state==F_OUT2) ? sm_tready : 0;
```

SRAM Connection

```
wire mux_state;
assign mux_state=!(reg_mode1_in==2||reg_mode1_in==3);
assign ram0_en   = (state==IN_COPY)?Inram_en  : ((mux_state)? in_ramf_en:in_ramu_en);
assign ram0_we   = (state==IN_COPY)?Inram_we  : ((mux_state)? in_ramf_we:in_ramu_we);
assign ram0_adr  = (state==IN_COPY)?Inram_adr : ((mux_state)? in_ramf_adr:in_ramu_adr);
assign ram0_d    = (state==IN_COPY)?Inram_d   : ((mux_state)? in_ramf_d:{48'b0,in_ramu_d});
assign in_ramu_q = ram0_q[15:0];
assign in_ramf_q = ram0_q;
```

```
assign ram1_en          = (mux_state) ? out_ramf_en  : out_ramu_en;
assign ram1_we          = (mux_state) ? out_ramf_we  : out_ramu_we;
assign ram1_adr         = (mux_state) ? out_ramf_adr : out_ramu_adr;
assign ram1_d           = (mux_state) ? out_ramf_d   : {48'b0,out_ramu_d};
assign out_ramu_q = ram1_q[15:0];
assign out_ramf_q = ram1_q;
```

Firmware

```
while (1) {
    value = reg_fsic_up;
    if (((value) && 1) == 1) {
        reg_fsic_up=1;
        reg_fsic_aa_mb0 = i;
        i++;
        //break;
    }
}
```

```
always @(posedge axi_clk or negedge axi_reset_n)  begin
  if ( !axi_reset_n ) begin
        rd_addr <= 0;
        rvalid_out <= 0;
  end
  else begin
    if (rd_state == RD_IDLE )
          if(arvalid && arready) begin
                rd_addr <= araddr;
                rvalid_out <= 1;
          end
        if (rd_state == RD_ADDR_DONE )
          if(rready && rvalid_out)
                rvalid_out <= 0;
  end
end

////
always @* begin
  if      (rd_addr[11:0] == 12'h000) rdata_tmp = (state==RESET);
  else                               rdata_tmp = 0;
end

RESET:
    if(awvalid_in && wvalid_in &&(awaddr[11:0] == 12'h000)&&(wdata[0]==1)) next_state=Command;
    else next_state = RESET;
```

DMA

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

SPEC

- FFT/iFFT **m2s** (DMA stream out 2049 data)
  - First data: kernel_mode
  - 2048 data: upper 32-bit and lower 32-bit of 1024 “double” data
- NTT/iNTT **m2s** (DMA stream out 1025 data)
  - First data: kernel_mode
  - 1024 data: 16-bit of 1024 “uint_16’ data
- FFT/iFFT **s2m** (stream in 2048 data)
  - upper 32-bit and lower 32-bit of 1024 “double” data
- NTT/iNTT **s2m** (stream in 1024 data)
  - 1024 16-bit “uint_16” data

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

AXIM -> AXIS (System Memory to Device)

- Parallel to stream (burst reading from memory)

```
do {
    if(final_m2s_len > MAX_BURST_LENGTH){
        count = MAX_BURST_LENGTH;
    }else{
        count = final_m2s_len;
    }
    high = 0;
    int a = 0;

    for (int i = 0; i < count; ++i) {
    #pragma HLS PIPELINE
        ////////////////////////////////////////////// /////
        if (high)
            out_val.data_filed = in_memory[a-1].upper;
        else
            out_val.data_filed = in_memory[a].lower;
        high = (even)? (!high) : high;
        ////////////////////////////////////////////// /////
        a = (even)? (high)? a + 1 : a : a + 1;

        if((final_m2s_len <= MAX_BURST_LENGTH) && (i == (count - 1)))
            out_val.last = 1;
        else
            out_val.last = 0;

        out_stream.write(out_val);
        final_m2s_len--;
    }
    if (even)
        in_memory += count / 2;
    else
        in_memory += count;

} while(final_m2s_len != 0);
```

for-loop indicates the burst cycles

read upper/lower bit from memory

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

AXIM -> AXIS (System Memory to Device)

- Stream-out

```
do {
    #pragma HLS PIPELINE
    data in_data = in_stream.read();
    out_val.data = in_data.data_filed;
    // out_val.user = in_data.upsb;
    out_val.last = in_data.last;
    out_stream.write(out_val);

    *buf_sts = (out_val.last)? 1 : 0;

} while(!out_val.last);
```

directly stream-out

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

INTELLIGENT COMPUTING
BOL edu

AXIS -> AXIM (Device to System Memory)

- Get in_stream from SOC side

```
do {
#pragma HLS PIPELINE
    in_val = in_stream.read();
    data out_val = {in_val.data, in_val.last};
    out_stream.write(out_val);

    s2m_err = 0;

    if ((in_len < s2m_len - 1) && (in_val.last == 1)) // t_la
        s2m_err = 1;
    if ((in_len == s2m_len - 1) && (in_val.last != 1)) // rea
        s2m_err = 2;
    count += 1; // burst count

    in_len += 1;

    if ((count == MAX_BURST_LENGTH) || (in_val.last == 1)) {
        out_counts.write(count);
        count = 0;
    }

} while(in_len < s2m_len);
```

tlast asserted but DMA hasn't reach stream length

reach stream length but tlast not asserted

Count to burst length (512)
if received tlast, directly write into memory
no matter the count

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

INTELLIGENT COMPUTING
BOL edu

AXIS -> AXIM (Device to System Memory)

- Stream to parallel (write to memory with burst)

```
do {
    count = in_counts.read();
    high = 0;
    int a = 0;
    for (int i = 0; i < count; ++i) {
    #pragma HLS PIPELINE
        in_val = in_stream.read();
        ////////////////////////////////////////////
        if (high)
            out_memory[a-1].upper = in_val.data_filed;
        else
            out_memory[a].lower = in_val.data_filed;
        high = (even)? (!high) : high;
        ////////////////////////////////////////////
        a = (even)? (high)? a + 1 : a : a + 1;
    }

    if (even) {
        out_memory += count / 2;
        final_s2m_len += count / 2;
    } else {
        out_memory += count;
        final_s2m_len += count;
    }

    if (final_s2m_len == 1024)
        out_memory -= 1024;

} while(final_s2m_len < s2m_len);
```

for-loop indicates the burst cycles

write upper/lower bit to memory

https://github.com/vic9112/PQC_Falcon/tree/main/impl_ASIC/UserDMA_v2/hls_userdma

Continuously read/update DMA status

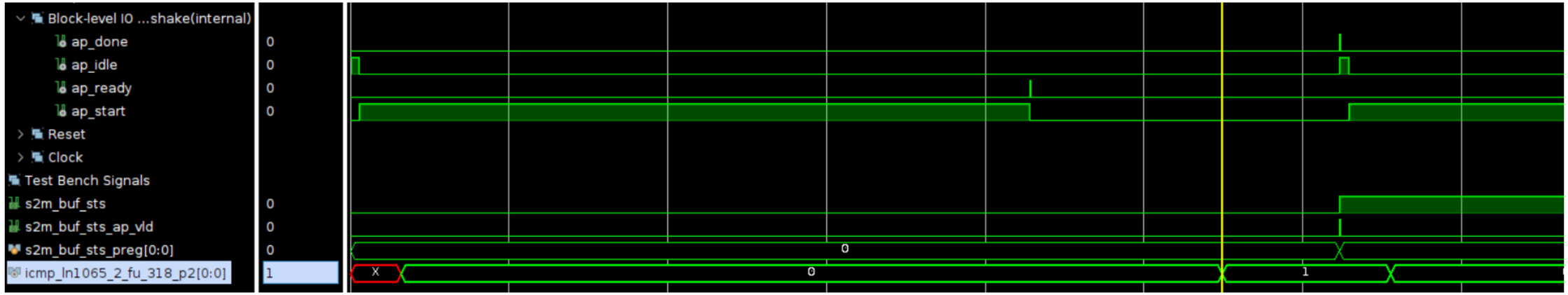
- Keyword: **volatile**

```
void userdma(
    hls::stream<trans_pkt> &inStreamTop,
    hls::stream<trans_pkt> &outStreamTop,
    ap_uint<2>             kernel_mode,  //
    bool volatile          *s2m_buf_sts,
    bool volatile          *m2s_buf_sts,
    memcell                s2mbuf[BUF_LEN],
    memcell                m2sbuf[BUF_LEN],
    ap_uint<2>             *s2m_err
    ) {
```

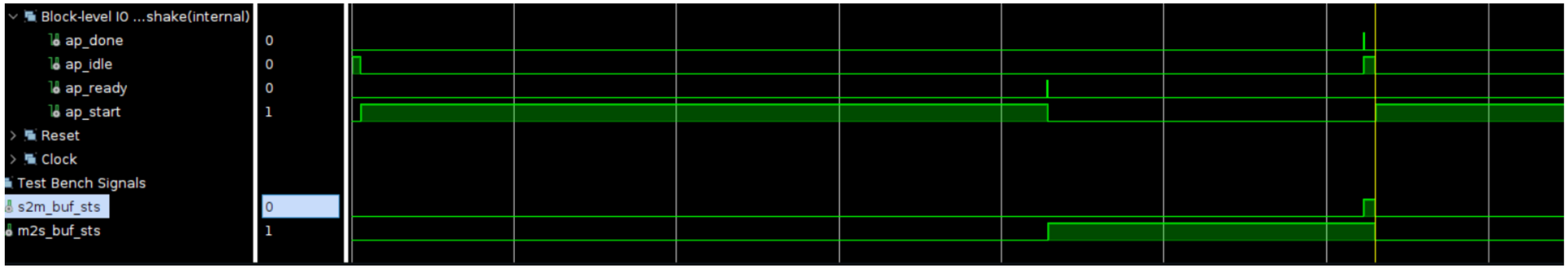
The volatile keyword informs the compiler that a variable's value may change at any time.

Continuously read/update DMA status

- Before setting “volatile”



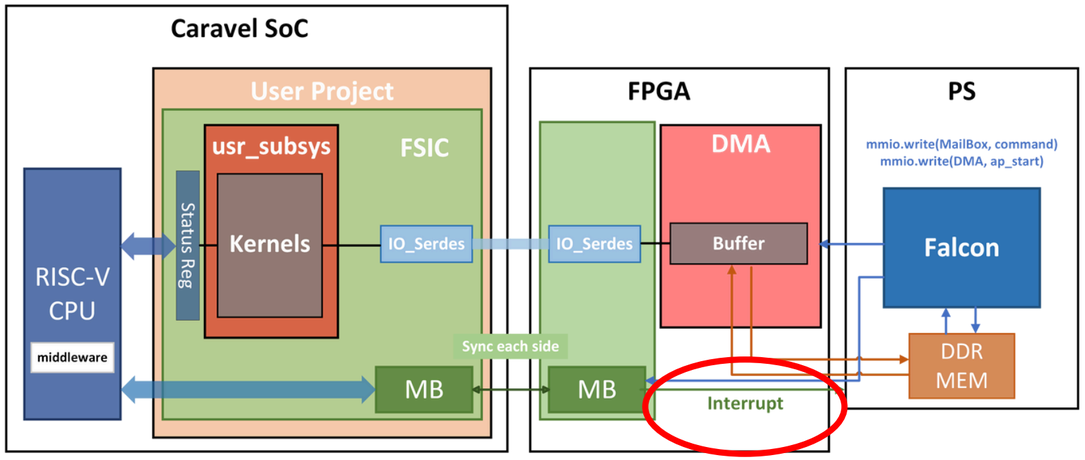
- After setting “volatile”



https://github.com/bol-edu/HLS-SOC-Discussions/discussions/224

Interrupt

Why using interrupt



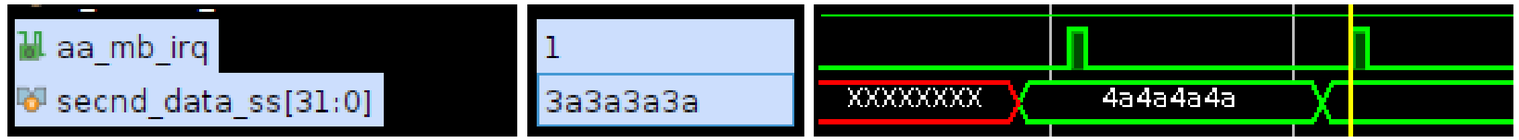
In our design, we put our kernel in user project Caravel SoC. When kernel finishing the task, we change the status register. The change in status register will trigger the middleware in RISC-V CPU to write the mailbox, which change the mailbox in FPGA correspondingly. After mailbox is written, it generates interrupt to the PS side.

To achieve this implementation, we add an interrupt controller in our original design. The interrupt is level trigger since we tried edge trigger but couldn’t get the interrupt. Also,We changed the design in our mailbox slightly. However, we can’t have the result we expect to get.

Improve Mailbox Design

- Initial: IRQ can’t reset/de-asserted
- Reset IRQ after reading mailbox:

```
//////////////////////////
if (rd_mb) begin
    aa_regs[1][0] <= 1'b0;
end
//////////////////////////
```



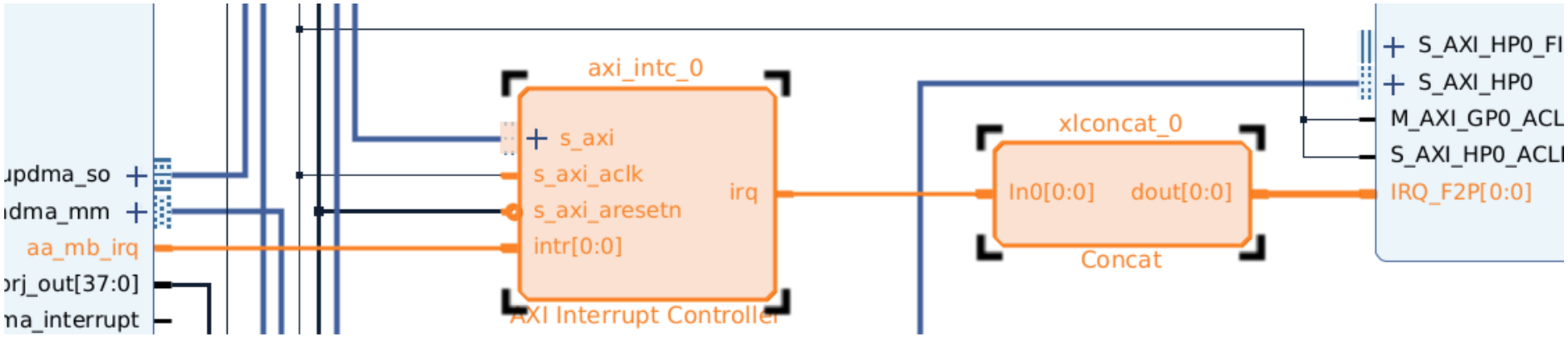

```
// combinational logic for interrupt control
always_comb begin
    if(aa_regs[0][0] && aa_regs[1][0]) begin
        axi_interrupt = 1'b1;
    end
    else begin
        axi_interrupt = 1'b0;
    end
end
```

aa_regs[0][0] stands for irq_enable
aa_regs[1][0] stands for irq trigger

https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/irq_test/vivado/vvd_srcs/caravel_soc/rtl/user/axilite_axis/rtl/axi_ctrl_logic.sv

Interrupt Controller

- Add AXI Interrupt Controller, connect pin aa_mb_irq on ps_axil to the input of AXI Interrupt Controller, then concatenate to pin IRQ_F2P on ZYNQ(PS)



Python

- Check interrupt pins and instance it

In [2]:
```
ol.interrupt_pins # Print out all interrupt pins
```

Out[2]:
```
{'axi_intc_0/intr': {'controller': 'axi_intc_0',
  'index': 0,
  'fullpath': 'axi_intc_0/intr'},
 'ps_axil_0/aa_mb_irq': {'controller': 'axi_intc_0',
  'index': 0,
  'fullpath': 'ps_axil_0/aa_mb_irq'}}
```

In [3]:
```
mbIRQ = ol.ps_axil_0.aa_mb_irq # Instance the IRQ from FPGA-side MailBox
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

INTELLIGENT COMPUTING
BOL edu

Python

- Asynchronous Interrupt Service Routine

```
# ================================================== #
# Create asynchronous task for Interrupt Service Routine
# ================================================== #
async def isr():
    ## Write aa_mb_irq_en ##
    #mmio.write(0x2100, 0x01)
    print("-> Waitting for MailBox interrupt")
    while(True):
        await mbIRQ.wait() # Wait for Interrupt
        ISR.set() # sets the interrupt
        print("**************************************")
        print("-> Interrupt asserted!!!")
        print("**************************************")
        print(f"Mailbox Pattern: {hex(mmio.read(0x2000))}")
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

FSIC Simulation

Simulation Flow

- Example : FFT

```
load_dma_FFT();
setting_dma_FFT();

$display($time, "=> Write PL_AA to enable IRQ");
offset = 0;
data = 32'h0000_0001;
axil_cycles_gen(WriteCyc, PL_AA, offset, data, 1);
axil_cycles_gen(ReadCyc, PL_AA, offset, data, 1);

start_algorithm();
start_DMA();

WaitIRQ(1'b1, 32'h3a3a3a3a);
CheckuserDMADone_FFT();
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

Simulation Tasks

- Load data to memory

```
reg [31:0] updma_FFT_x [0:2047];
reg [31:0] updma_FFT_x_data;
task load_dma_FFT;
    begin
        $display($time, "=> load dma FFT ...");
        $display($time, "=> ================================================================");

        $readmemh("../../../../../FFT_in.hex", updma_FFT_x);

        fd = $fopen ("../../../../../updma_FFT_input.log", "w");
        for (index = 0; index < 2048; index +=1) begin
            updma_FFT_x_data |= updma_FFT_x[index];
            slave_agent3.mem_model.backdoor_memory_write_4byte(addri+4*index,updma_FFT_x_data,4'b1111);
            updma_FFT_x_data = 0;
            $fdisplay(fd, "%08h", slave_agent3.mem_model.backdoor_memory_read_4byte(addri+4*index));
        end
        $fclose(fd);

    end
endtask
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

Simulation Tasks

- Configure kernel_mode...

```
task setting_dma_FFT;
    begin

        $display($time, "=> -----------------------------------------------------------------");
        $display($time, "=> FpgaLocal_Write: PL_UPDMA, s2m set buffer low...");
        offset = 32'h0000_0038;
        data = 32'h4508_0000;
        axil_cycles_gen(WriteCyc, PL_UPDMA, offset, data, 1);
        //#20us
        axil_cycles_gen(ReadCyc, PL_UPDMA, offset, data, 1);
        //#20us
        if(data == 32'h4508_0000) begin
            $display($time, "=> Fpga2Soc_Write PL_UPDMA offset %h = %h, PASS", offset, data);
        end else begin
            $display($time, "=> Fpga2Soc_Write PL_UPDMA offset %h = %h, FAIL", offset, data);
            ->> error_event;
        end
```

0x38 -> 0x3c -> 0x44 -> 0x48 -> 0x10

Configuration Address	Description / Bit select
0x00	Control signals bit 0 - ap_start (Read/Write/COH) bit 1 - ap_done (Read/COR) bit 2 - ap_idle (Read) bit 3 - ap_ready (Read/COR)
0x10	Data signal of kernel_mode bit 1~0 - kernel_mode[1:0] (Read/Write) 0: FFT 1: iFFT 2: NTT 3: iNTT
0x18	Data signal of s2m_buf_sts bit 0 - s2m_buf_sts[0] (Read)
0x1c	Control signal of s2m_buf_sts bit 0 - s2m_buf_sts_ap_vld (Read/COR)
0x28	Data signal of m2s_buf_sts bit 0 - m2s_buf_sts[0] (Read)
0x2c	Control signal of m2s_buf_sts bit 0 - m2s_buf_sts_ap_vld (Read/COR)
0x38	Data signal of s2mbuf bit 31~0 - s2mbuf[31:0] (Read/Write)
0x3c	Data signal of s2mbuf bit 31~0 - s2mbuf[63:32] (Read/Write)
0x44	Data signal of m2sbuf bit 31~0 - m2sbuf[31:0] (Read/Write)
0x48	Data signal of m2sbuf bit 31~0 - m2sbuf[63:32] (Read/Write)
0x50	Data signal of s2m_err bit 1~0 - s2m_err[1:0] (Read)
0x54	Control signal of s2m_err bit 0 - s2m_err_ap_vld (Read/COR)

https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

INTELLIGENT COMPUTING
BOL edu

Simulation Tasks

- Enable IRQ

```
$display($time, "=> Write PL_AA to enable IRQ");
offset = 0;
data = 32'h0000_0001;
axil_cycles_gen(WriteCyc, PL_AA, offset, data, 1);
axil_cycles_gen(ReadCyc, PL_AA, offset, data, 1);
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

Simulation Tasks

- Start algorithm & DMA

```
task start_algorithm;
    begin
        // Select user project2
        $display($time, "=> Fpga2Soc_Write: SOC_CC");

        offset = 0;
        data = 32'h0000_0002;
        axil_cycles_gen(WriteCyc, SOC_CC, offset, data, 1);
        axil_cycles_gen(ReadCyc, SOC_CC, offset, data, 1);

        if(data == 32'h0000_0002) begin
            $display($time, "=> Fpga2Soc_Write SOC_CC offset %h = %h, PASS", offset, data);
        end else begin
            $display($time, "=> Fpga2Soc_Write SOC_CC offset %h = %h, FAIL", offset, data);
            ->> error_event;
        end

    end
endtask

task start_DMA;
    begin
   $display($time, "=> ================================================================================");
        $display($time, "=> Start DMA streaming x in, program ap_start to DMA");
        $display($time, "=> ================================================================================");
        offset = 32'h0000_0000;
        data = 32'h0000_0001;
        axil_cycles_gen(WriteCyc, PL_UPDMA, offset, data, 1);
    end
endtask
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

Simulation Tasks

- Interrupt

```
task WaitIRQ;
    input [0:0]  plaa_en;
    input [31:0] pattern;
    begin
        if (plaa_en == 1'b1) begin
            $display($time, "=> Write PL_AA to enable IRQ");
            offset = 0;
            data = 32'h0000_0001;
            axil_cycles_gen(WriteCyc, PL_AA, offset, data, 1);
            axil_cycles_gen(ReadCyc, PL_AA, offset, data, 1);
        end

        // Wait for IRQ
        keepChk = 1;
        while (keepChk) begin
        //$display($time, "=> Wating irq done...");
        #10
            if (DUT.design_1_i.ps_axil_0.aa_mb_irq == 1'b1) begin
                keepChk = 1'b0;
                $display($time, "=> ********************************************************************");
                $display($time, "=> iNTT Interrupt Asserted!!");
                $display($time, "=> ********************************************************************");
            end
        end
        // Read MailBox
        axil_cycles_gen(ReadCyc, PL_AA_MB, offset, data, 1);
        if(data == pattern) begin
            keepChk = 0;
            $display($time, "=> FpgaLocal_Read PL_AA_MB = %h, PASS", data);
        end

    end
endtask
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_sim_v3_irq/vivado/fsic_tb.sv

Simulation Tasks

- Check DMA transfer done

```
task CheckuserDMADone_FFT;
    begin
        $display($time, "=> Starting CheckuserDMADone()...");
        $display($time, "=> ================================================================================");
        $display($time, "=> FpgaLocal_Read: PL_UPDMA");

        keepChk = 1;
        offset = 32'h0000_0018;
        $display($time, "=> Waiting buffer transfer done...");
        while (keepChk) begin
            $display($time, "=> Waiting buffer transfer done...");
            #10us
            axil_cycles_gen(ReadCyc, PL_UPDMA, offset, data, 0);
            if(data == 32'h0000_0001) begin
                $display($time, "=> Buffer transfer done. offset %h = %h, PASS", offset, data);
                keepChk = 0;

                fd = $fopen ("../../../../../updma_output_FFT.log", "w");
                for (index = 0; index < 2048; index +=1) begin
                    reg signed [31:0]anser;
                    anser=slave_agent2.mem_model.backdoor_memory_read_4byte(addro+4*index);
                    $fdisplay(fd, "%08h",anser);
                end

                $fclose(fd);
           end
        end

        ->> userdma_done;
        $display($time, "=> End FFT CheckuserDMADone()...");
        $display($time, "=> ================================================================================");

    end
endtask
```

Verify the data

updma_output_FFT.log
~/PQC_Falcon/impl_ASIC/fiFFNTT_sim_v3_irq/vivado

```
1 7c194a63
2 40f206b1
3 1b348361
4 c0d09013
5 b8f8d817
6 40f71592
7 13e8d8b3
8 40f4097f
9 b2bef23a
10 c0f57c4a
11 428827e0
12 40c0365c
13 e50ead9e
14 c0e2b8a8
15 d08376de
16 c0e30888
17 c8aead2e
18 40f02f6d
19 e69b059e
20 40fa331f
21 321fe554
22 c0e8f328
23 d0abc354
24 40fa1fa3
25 e26da2e2
26 c0f8b388
27 890c2a75
28 410286a3
29 1faa0992
30 40eb56aa
31 c5c5e700
32 405e93a6
33 a3494960
34 40a67c8a
35 d4cbd06d
36 40f8cd84
37 10573a16
38 40f7e58d
```

FFT_out.hex
~/PQC_Falcon/impl_ASIC/fiFFNTT_sim_v3_irq/vivado

```
1 7c194a63
2 40f206b1
3 1b348361
4 c0d09013
5 b8f8d817
6 40f71592
7 13e8d8b3
8 40f4097f
9 b2bef23a
10 c0f57c4a
11 428827e0
12 40c0365c
13 e50ead9e
14 c0e2b8a8
15 d08376de
16 c0e30888
17 c8aead2e
18 40f02f6d
19 e69b059e
20 40fa331f
21 321fe554
22 c0e8f328
23 d0abc354
24 40fa1fa3
25 e26da2e2
26 c0f8b388
27 890c2a75
28 410286a3
29 1faa0992
30 40eb56aa
31 c5c5e700
32 405e93a6
33 a3494960
34 40a67c8a
35 d4cbd06d
36 40f8cd84
37 10573a16
38 40f7e58d
```

Simulation Result

- FFT

```
 8212458=> ================================================================
 8212458=> Start DMA streaming x in, program ap_start to DMA
 8212458=> ================================================================
 8214058=> AXI4LITE_WRITE_BURST 60009000, value: 0001, resp: 00
 8214058=> Write PL_AA to enable IRQ
 8216658=> AXI4LITE_WRITE_BURST 60002100, value: 0001, resp: 00
 8220258=> AXI4LITE_READ_BURST 60002100, value: 0001, resp: 00
23333868=> ****************************************************************
23333868=> FFT Interrupt Asserted!!
23333868=> ****************************************************************
23337458=> AXI4LITE_READ_BURST 60002000, value: 0001, resp: 00
23337458=> Starting CheckuserDMADone()...
23337458=> ================================================================
23337458=> FpgaLocal_Read: PL_UPDMA
24349458=> Buffer transfer done. offset 018 = 00000001, PASS
24349458=> End FFT CheckuserDMADone()...
24349458=> ================================================================
24849458=> End of the test...
```

Simulation Result

- iFFT

```
 8212458=> ================================================================
 8212458=> Start DMA streaming x in, program ap_start to DMA
 8212458=> ================================================================
 8214058=> AXI4LITE_WRITE_BURST 60009000, value: 0001, resp: 00
 8214058=> Write PL_AA to enable IRQ
 8216658=> AXI4LITE_WRITE_BURST 60002100, value: 0001, resp: 00
 8220258=> AXI4LITE_READ_BURST 60002100, value: 0001, resp: 00
23333868=> ****************************************************************
23333868=> iFFT Interrupt Asserted!!
23333868=> ****************************************************************
23337458=> AXI4LITE_READ_BURST 60002000, value: 0001, resp: 00
23337458=> Starting CheckuserDMADone()...
23337458=> ================================================================
23337458=> FpgaLocal_Read: PL_UPDMA
24349458=> Buffer transfer done. offset 018 = 00000001, PASS
24349458=> End iFFT CheckuserDMADone()...
24349458=> ================================================================
24849458=> End of the test...
```

Simulation Result

- NTT

```
 8212458=> ================================================================
 8212458=> Start DMA streaming x in, program ap_start to DMA
 8212458=> ================================================================
 8214058=> AXI4LITE_WRITE_BURST 60009000, value: 0001, resp: 00
 8214058=> Write PL_AA to enable IRQ
 8216658=> AXI4LITE_WRITE_BURST 60002100, value: 0001, resp: 00
 8220258=> AXI4LITE_READ_BURST 60002100, value: 0001, resp: 00
35681868=> ****************************************************************
35681868=> NTT Interrupt Asserted!!
35681868=> ****************************************************************
35685458=> AXI4LITE_READ_BURST 60002000, value: 0001, resp: 00
35685458=> Starting CheckuserDMADone()...
35685458=> ================================================================
35685458=> FpgaLocal_Read: PL_UPDMA
36213458=> Buffer transfer done. offset 018 = 00000001, PASS
36213458=> End NTT CheckuserDMADone()...
36213458=> ================================================================
36713458=> End of the test...
```

Simulation Result

- iNTT

```
 8212458=> ================================================================
 8212458=> Start DMA streaming x in, program ap_start to DMA
 8212458=> ================================================================
 8214058=> AXI4LITE_WRITE_BURST 60009000, value: 0001, resp: 00
 8214058=> Write PL_AA to enable IRQ
 8216658=> AXI4LITE_WRITE_BURST 60002100, value: 0001, resp: 00
 8220258=> AXI4LITE_READ_BURST 60002100, value: 0001, resp: 00
35681868=> ****************************************************************
35681868=> iNTT Interrupt Asserted!!
35681868=> ****************************************************************
35685458=> AXI4LITE_READ_BURST 60002000, value: 0001, resp: 00
35685458=> Starting CheckuserDMADone()...
35685458=> ================================================================
35685458=> FpgaLocal_Read: PL_UPDMA
36213458=> Buffer transfer done. offset 018 = 00000001, PASS
36213458=> End iNTT CheckuserDMADone()...
36213458=> ================================================================
36713458=> End of the test...
```

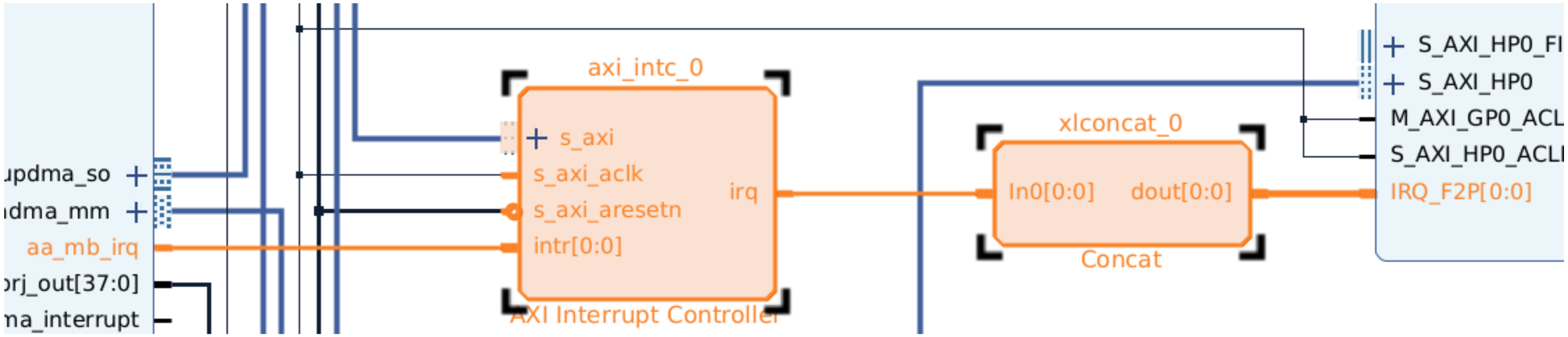
Simulation run time

Algorithm	Run time	Final time
FFT / iFFT	1512_1410	1512_5000
NTT / iNTT	2746_9410	2747_3000

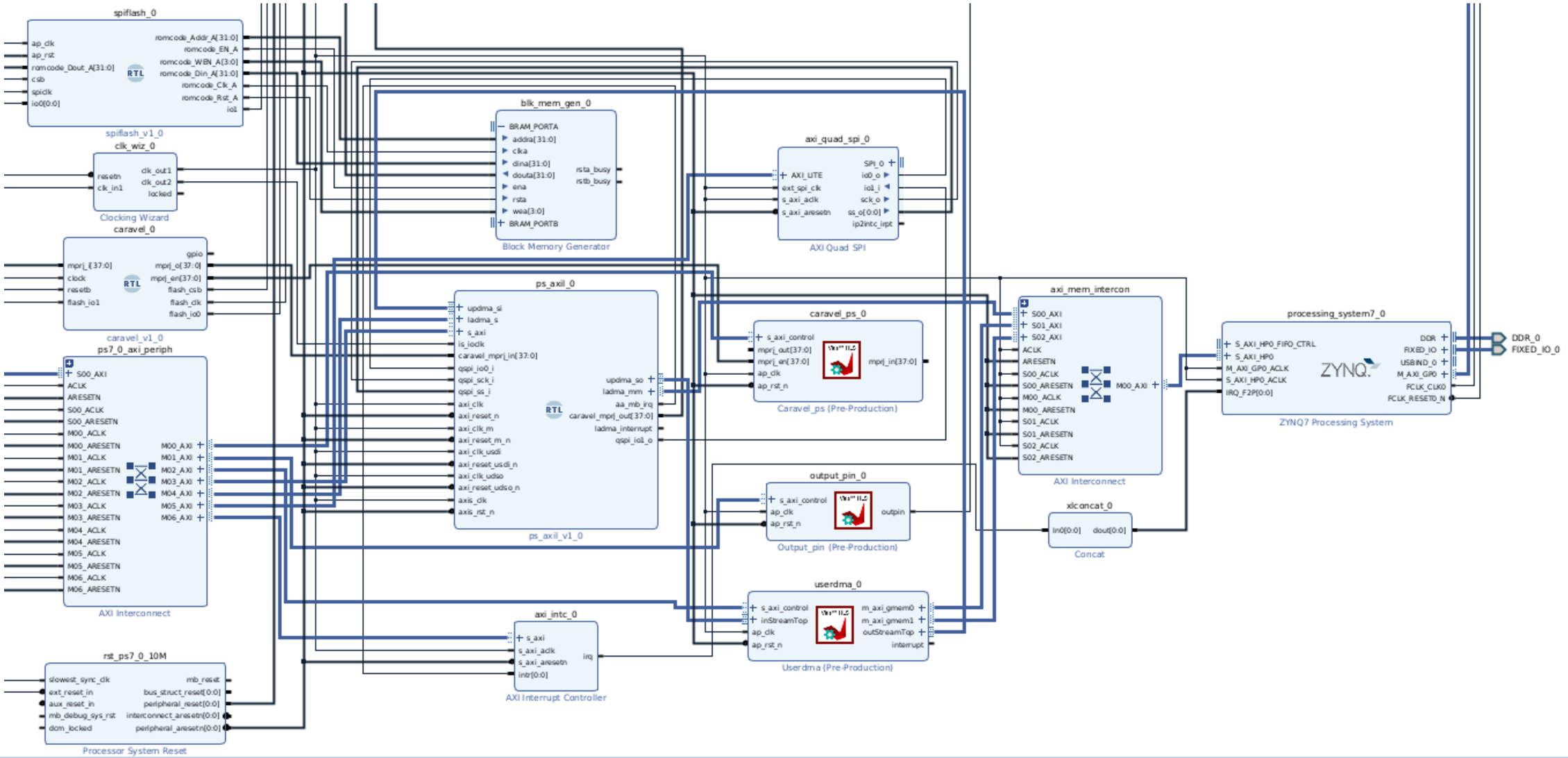
FSIC Validation

System Block Diagram on Vivado GUI

- Since we can't add a new ISR directly in Python code, we need an additional procedure after running run_vivado_fsic
  a. Add AXI Interrupt Controller in FPGA block design. Choose Interrupt type: level trigger and Interrupt Output Connection to Single.
  b. The mailbox(MB) must provide an interrupt pin(aa_mb_irq) and connect to the AXI Interrupt Controller.
  c. Connect the irq of Interrupt Controller to IRQ_F2P[0:0] in PS CPU.



System Block Diagram on Vivado GUI



Validation Tasks

```
n_s2mbuf = allocate(shape=(1024,), dtype=np.uint64)
n_m2sbuf = allocate(shape=(1024,), dtype=np.uint64)
f_s2mbuf = allocate(shape=(1024,), dtype=np.float64)
f_m2sbuf = allocate(shape=(1024,), dtype=np.float64)
```

- fiFFNTT

```
async def fiFFNTT(poly, mode):
    print("Kernel start")
    mmio.write(offset+0x10, mode) # mode

    # Check inverse
    if (mode == 1):
        poly = get_ifft1024(poly)
    elif (mode == 3):
        poly = get_intt(poly)

    if (mode in (0, 1)):
        mmio.write(offset + 0x38, f_s2mbuf.device_address)
        mmio.write(offset + 0x44, f_m2sbuf.device_address)
        for i in range(1024):
            f_m2sbuf[i] = poly[i]
    else:
        mmio.write(offset + 0x38, n_s2mbuf.device_address)
        mmio.write(offset + 0x44, n_m2sbuf.device_address)
        for i in range(1024):
            n_m2sbuf[i] = poly[i]

    # dma_start
    mmio.write(offset + 0x00, 0x00000001)
```

```
    timeKernelStart = time()

    await ISR.wait()
    ISR.clear()
    #while True:
    #    if mmio.read(offset + 0x18) == 0x01:
    #        break
    timeKernelEnd = time()

    if (mode == 0):
        poly_out = get_fft1024(f_s2mbuf)
    elif (mode == 1):
        poly_out = f_s2mbuf
    elif (mode == 2):
        poly_out = get_ntt(n_s2mbuf)
    elif (mode == 3):
        poly_out = n_s2mbuf

    # Execution time
    print("Kernel execution time: " + str(timeKernelEnd - timeKernelStart) + " s")

    return poly_out
```

https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

Validation Tasks

- Test fiFFNTT multiple times with ISR

```python
async def falcon():
    FFT_in = [0 for i in range(1024)]
    with open("FFT_in.txt", "r+") as file:
        for i in range(1024):
            line = file.readline()
            FFT_in[i] = float(line)
    NTT_in = [i for i in range(1024)]
    fft_out  = await fiFFNTT(FFT_in, 0)
    ifft_out = await fiFFNTT(fft_out, 1)
    ntt_out  = await fiFFNTT(NTT_in, 2)
    intt_out = await fiFFNTT(ntt_out, 3)
    print(f"FFT out: {fft_out}")
    print(f"iFFT out: {ifft_out}")
    print(f"NTT out: {ntt_out}")
    print(f"iNTT out: {intt_out}")
```

```python
async def async_main():
    # Create ISR
    task1 = asyncio.create_task(isr())
    task2 = asyncio.create_task(falcon())
    await asyncio.sleep(5)
    task1.cancel()
    try:
        await task1
    except asyncio.CancelledError:
        print("-> ISR is cancelled.")
```

https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

Validation Tasks

- Test fiFFNTT multiple times with ISR

```python
# ================================================== #
# Create asynchronous task for Interrupt Service Routine
# ================================================== #
async def isr():
    ## Write aa_mb_irq_en ##
    #mmio.write(0x2100, 0x01)
    print("-> Waitting for MailBox interrupt")
    while(True):
        await mbIRQ.wait() # Wait for Interrupt
        ISR.set() # sets the interrupt
        print("*****************************************")
        print("-> Interrupt asserted!!!")
        print("*****************************************")
        print(f"Mailbox Pattern: {hex(mmio.read(0x2000))}")
```

```python
async def async_main():
    # Create ISR
    task1 = asyncio.create_task(isr())
    task2 = asyncio.create_task(falcon())
    await asyncio.sleep(5)
    task1.cancel()
    try:
        await task1
    except asyncio.CancelledError:
        print("-> ISR is cancelled.")
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

Run time

```
-> Waitting for MailBox interrupt
Kernel start
*****************************************
-> Interrupt asserted!!!
*****************************************
Mailbox Pattern: 0x3a3a3a3a
Kernel execution time: 0.021342992782592773 s
Kernel start
*****************************************
-> Interrupt asserted!!!
*****************************************
Mailbox Pattern: 0x3a3a3a3a
Kernel execution time: 0.020910978317260742 s
Kernel start
*****************************************
-> Interrupt asserted!!!
*****************************************
Mailbox Pattern: 0x3a3a3a3a
Kernel execution time: 0.033033132553100586 s
Kernel start
*****************************************
-> Interrupt asserted!!!
*****************************************
Mailbox Pattern: 0x3a3a3a3a
Kernel execution time: 0.029328107833862305 s
```


https://github.com/vic9112/PQC_Falcon/blob/main/impl_ASIC/fiFFNTT_val_v3/fiFFNTT_test_irq.ipynb

Synopsys Flow

Lab Synthesis

- QoR report

Timing Path Group 'clk'	
Levels of Logic:	292.00
Critical Path Length:	5.91
Critical Path Slack:	-4.20
Critical Path Clk Period:	2.00
Total Negative Slack:	-2984.08
No. of Violating Paths:	878.00
Worst Hold Violation:	0.00
Total Hold Violation:	0.00
No. of Hold Violations:	0.00

Cell Count	
Hierarchical Cell Count:	94
Hierarchical Port Count:	8912
Leaf Cell Count:	50043
Buf/Inv Cell Count:	10825
Buf Cell Count:	2559
Inv Cell Count:	8267
CT Buf/Inv Cell Count:	0
Combinational Cell Count:	48903
Sequential Cell Count:	1140
Macro Count:	0

Area	
Combinational Area:	18233.214946
Noncombinational Area:	1423.730427
Buf/Inv Area:	2211.830387
Total Buffer Area:	719.15
Total Inverter Area:	1493.84
Macro/Black Box Area:	0.000000
Net Area:	38246.330675
Cell Area:	19656.945373
Design Area:	57903.276047

Future Work

Future Work

- Turn KeyGen, Sign into hardware kernel

- Tapeout (T18-113D, Nov)

梯次	開放申請	教育申請截止	前瞻申請截止	速審申請截止	學術自費申請截止	審查會	第一階段簽認截止	Chip Out	測試報告繳交期限
T18-113A	113.01.29	113.02.05	113.02.05	113.02.19	113.02.19	X	113.02.28	113.06.07	113.08.11
T18-113B	113.05.13	113.05.20	113.05.20	113.05.27	113.05.27	X	113.06.05	113.09.13	113.11.17
T18-113C	113.08.19	113.08.26	113.08.26	113.09.02	113.09.02	X	113.09.11	113.12.20	114.02.23
T18-113D	113.11.04	113.11.11	113.11.11	113.11.18	113.11.18	X	113.11.27	114.03.07	114.05.11
*T18-114A	114.02.03	114.02.10	114.02.10	114.02.17	114.02.17	X	114.02.26	114.06.06	114.08.10