FFT
설계 & 검증 Project

대한상공회의소 서울기술교육센터
AI 시스템 반도체 설계(2기)

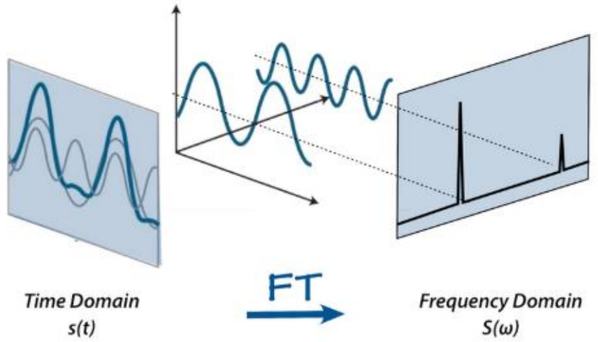
1조 고종완 권희식 장찬원 서윤철

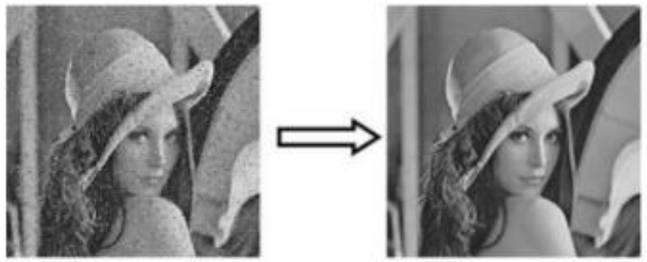
대한상공회의소
서울기술교육센터

CONTENT

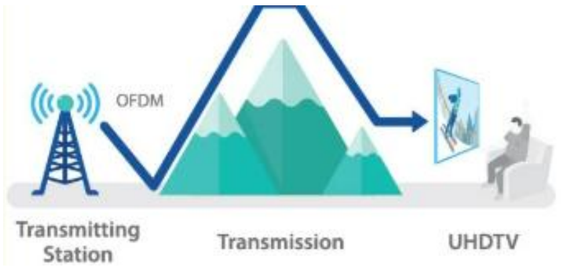
FFT (Fast Fourier Transform)

- 고속 푸리에 변환(FFT, Fast Fourier Transfrom)은 이산 푸리에 변환(DFT, Discrete Fourier Transform)을 효율적으로 계산하기 위한 알고리즘





영상 처리



OFDM

PART 1. Floating Point Model & Fixed Point Model

FFT (Fast Fourier Transform)

Floating-point FFT

•목적: 알고리즘 검증, 기능 구현

•특징:
- 정밀도(Precision)가 매우 높음.
- 오버플로(Overflow), 언더플로(Underflow) 문제 적음.
- 소프트웨어 기반 시뮬레이션(MATLAB, Python)에서 정확한 기준값 제공.

•용도:
- 알고리즘의 동작을 확인.
- Fixed-Point 설계 전에 레퍼런스 모델로 사용.

Fixed-point FFT

•목적: 하드웨어 구현

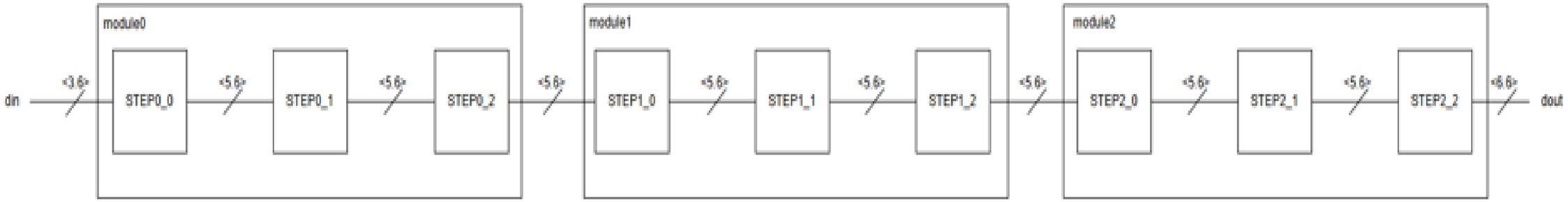
•특징:
- •정수 + 소수 부분을 비트 폭 제한으로 표현.
- •오버플로, 스케일링(Shift) 관리 필요.
- •FPGA, ASIC 설계에서 자원(LUT, DSP) 최적화 가능.

•용도:
- •RTL 설계, FPGA/ASIC 구현.
- •연산 복잡도를 줄여 고속 연산 가능.

FFT (Fast Fourier Transform)

MATLAB

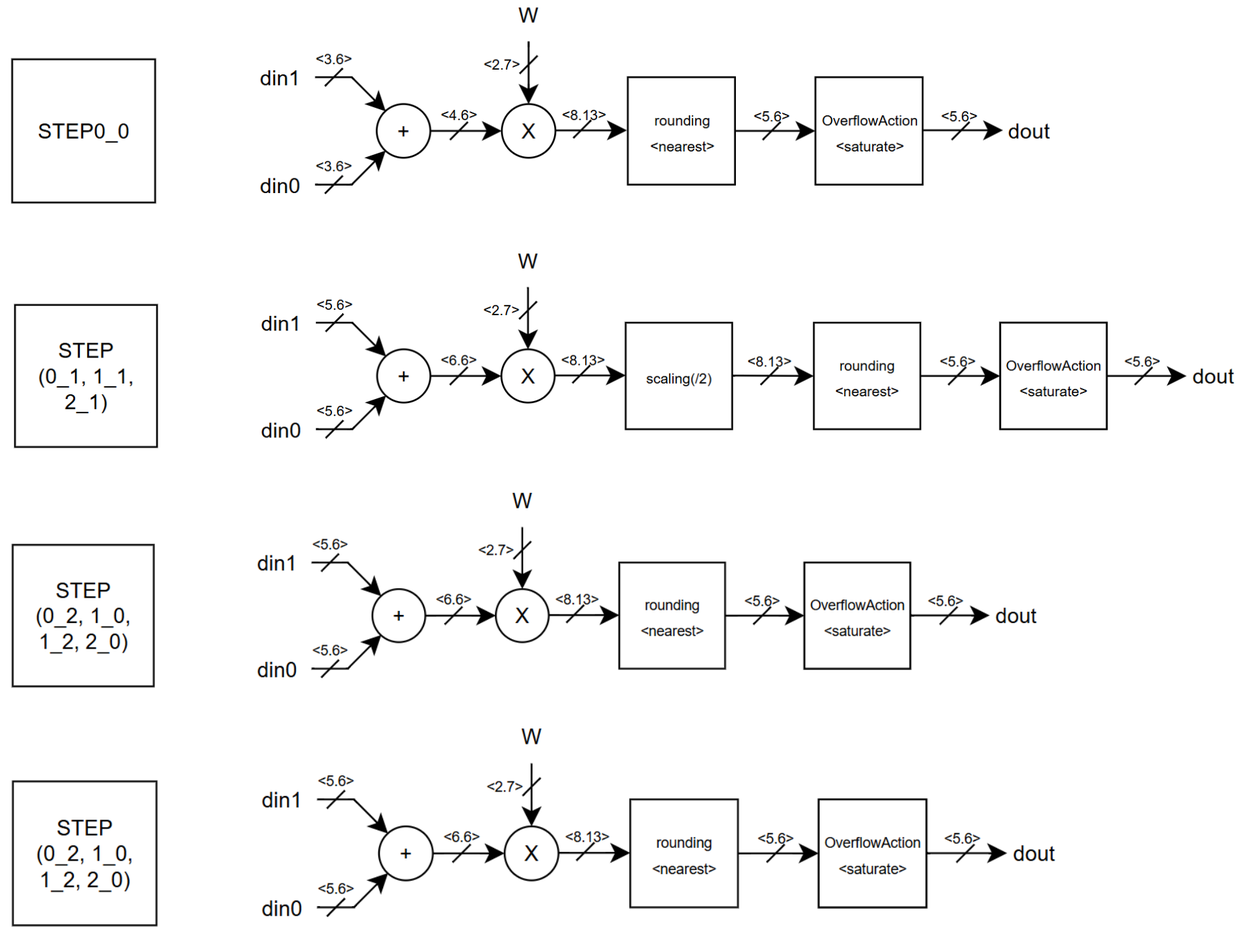
FFT Block Diagram



FFT (Fast Fourier Transform)

MATLAB

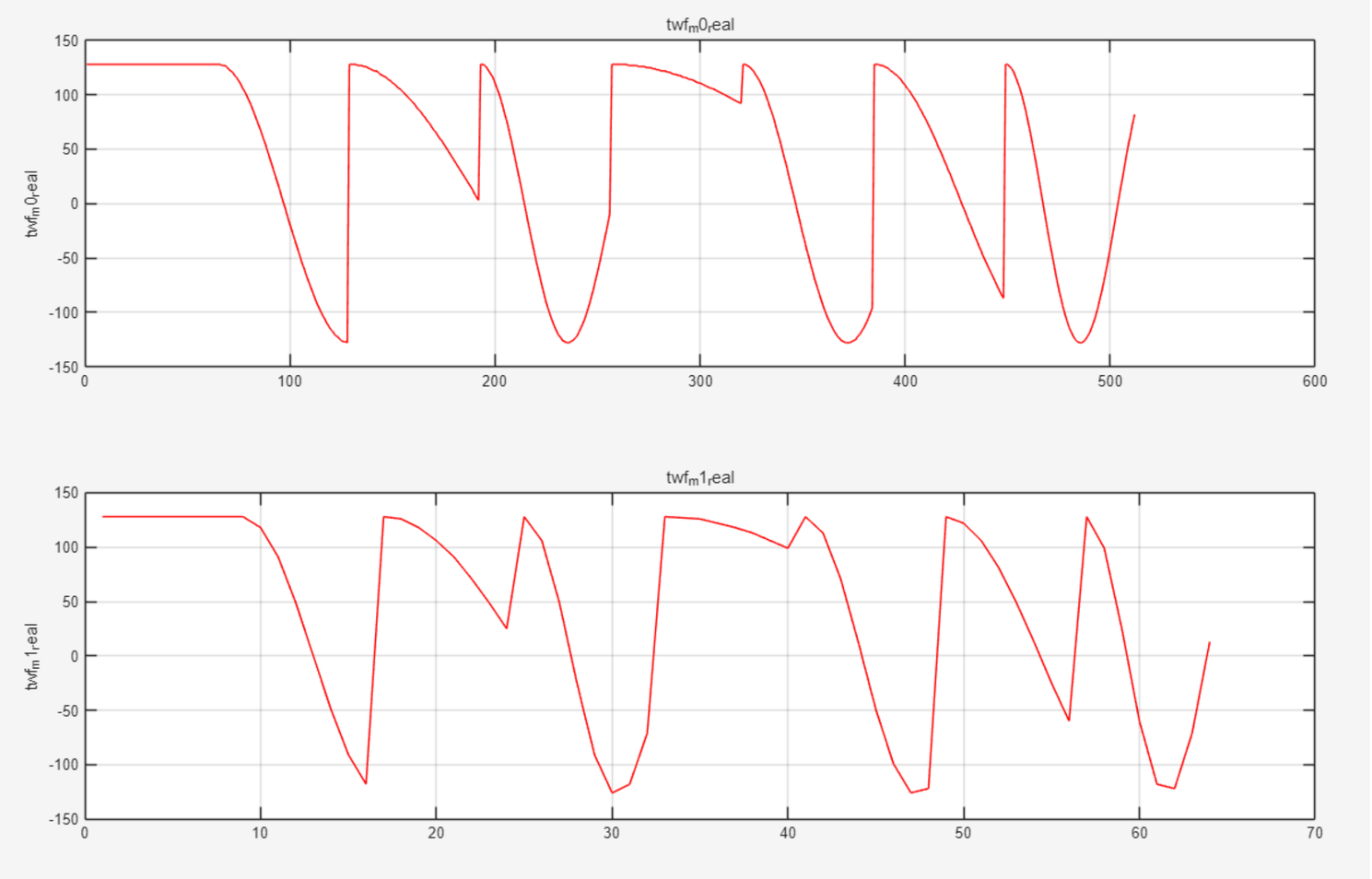
FFT BLOCK DIAGRAM



FFT (Fast Fourier Transform)

MATLAB

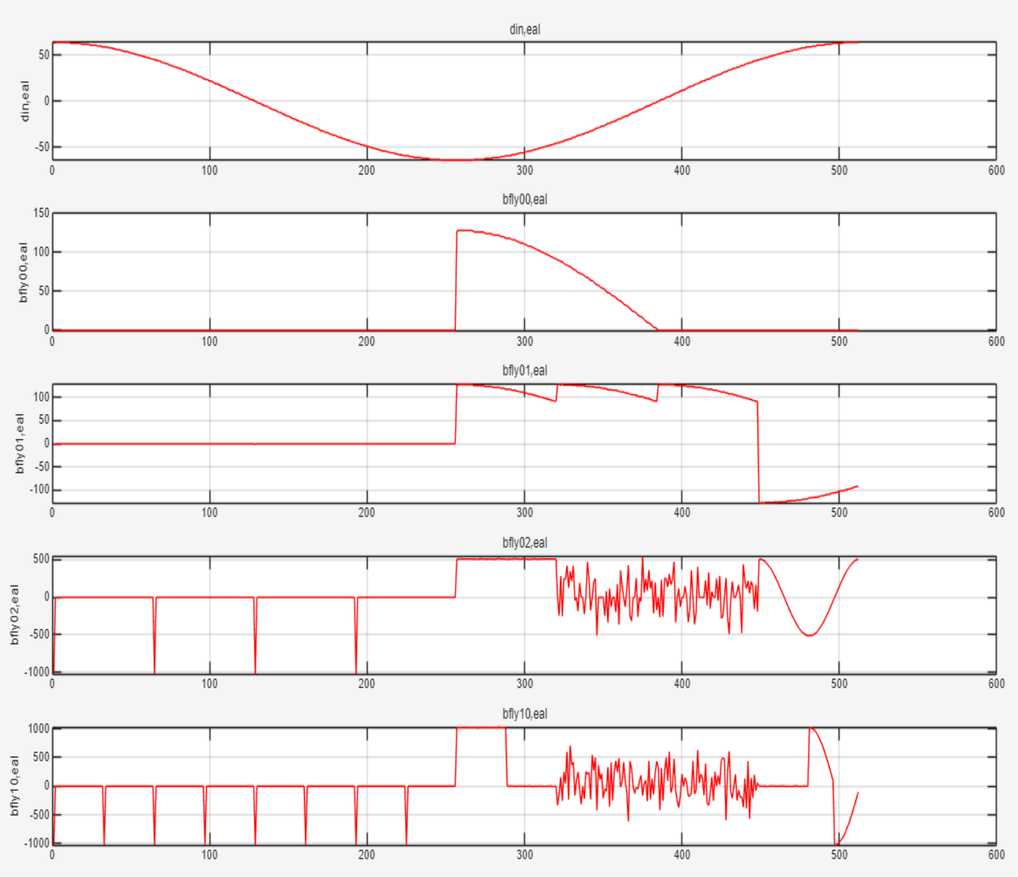
```
fac8_0 = [1, 1, 1, -j];

fac8_1 = [256, 256, 256, -j*256, 256, 181-j*181, 256, -181-j*181];
```

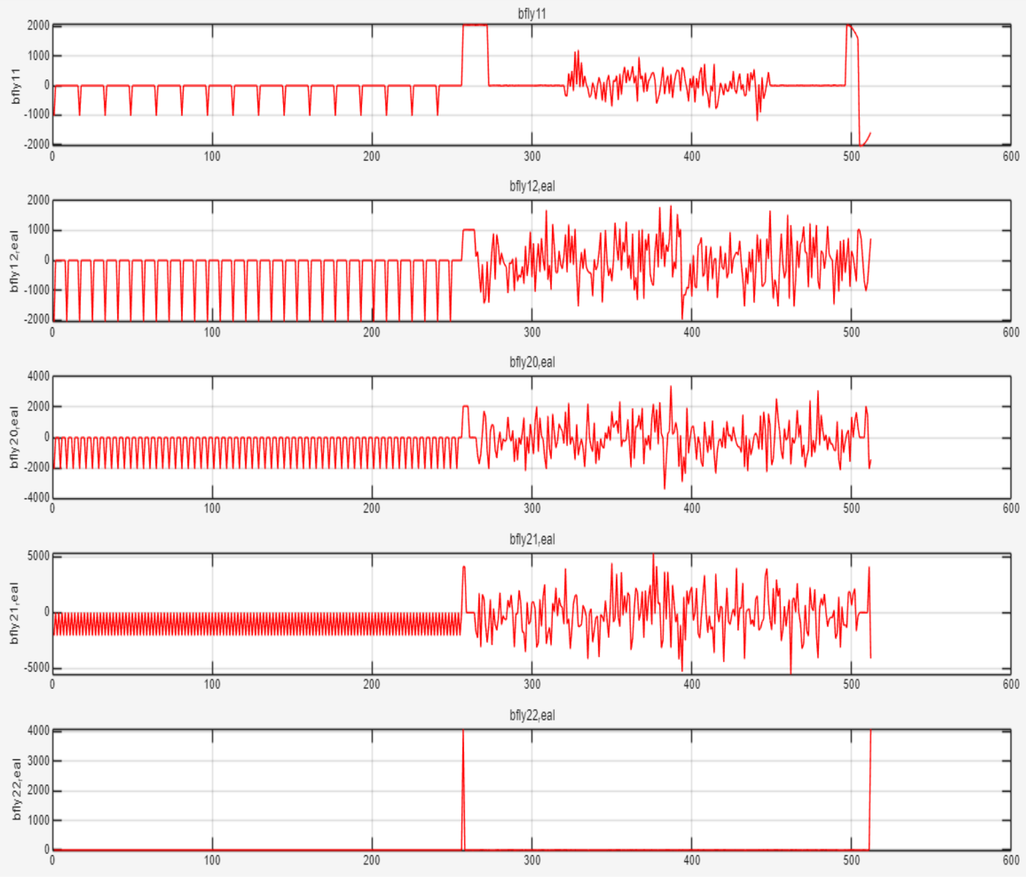


FFT (Fast Fourier Transform)

MATLAB

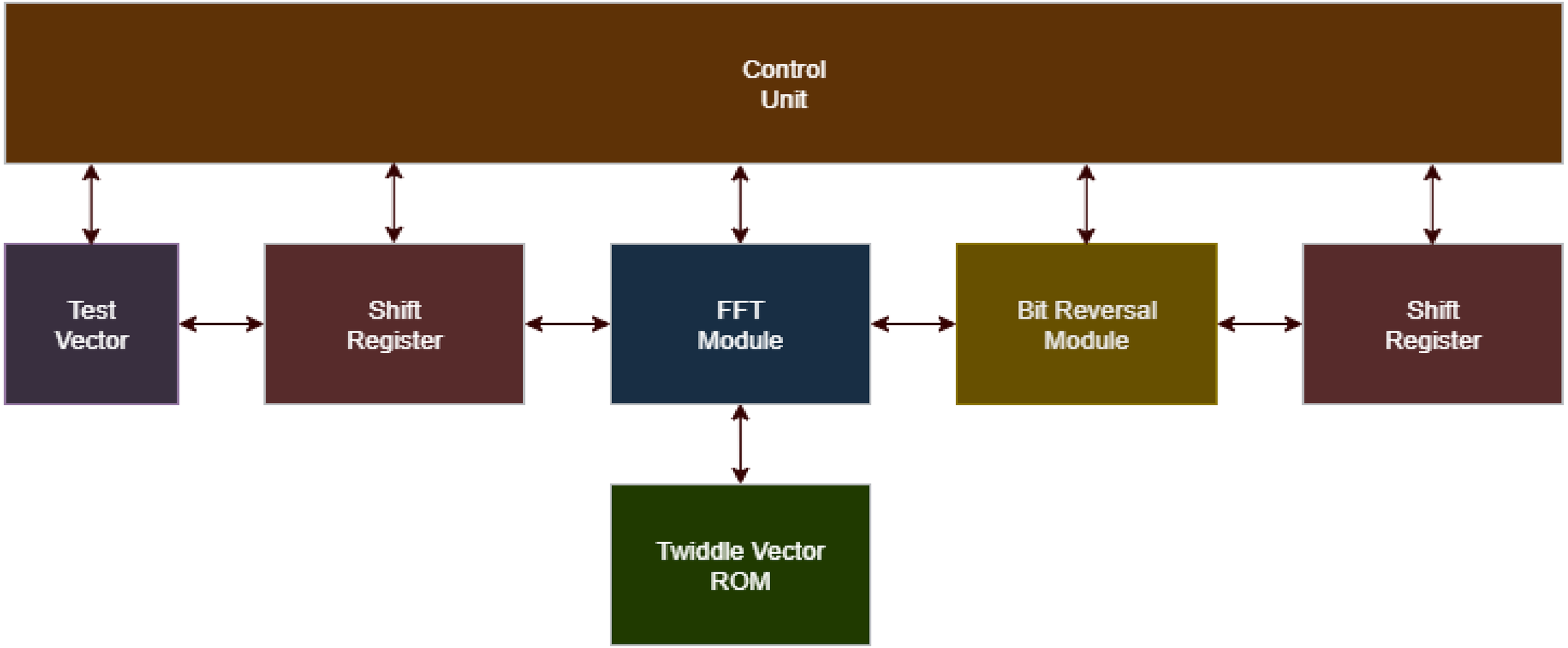
Fixed Point FFT





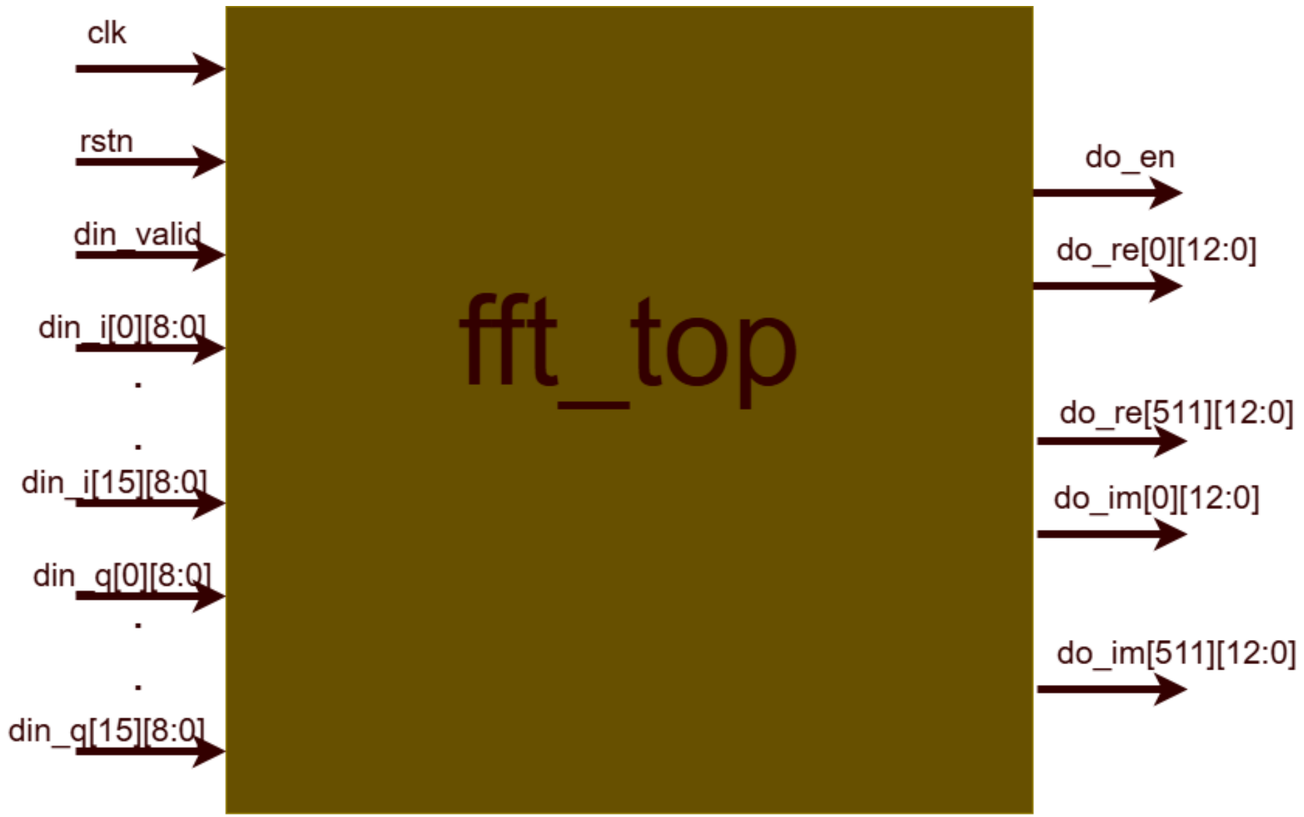
FFT (Fast Fourier Transform)

Hardware Block Diagram



FFT (Fast Fourier Transform)

FFT BLOCK DIAGRAM



SQNR(Signal-to-Quantization Noise Ratio)

양자화 오류의 정도를 나타내는 지표

SQNR이 높을수록 양자화 잡음이 적고, 원본 신호와 양자화된 신호 유사도 증가

SQNR과 양자화 잡음

양자화 잡음은 아날로그 신호를
디지털로 변환할 때 발생하는 오차

신호의 세부 사항이 소실되거나
오차가 발생하는 과정에서 발생

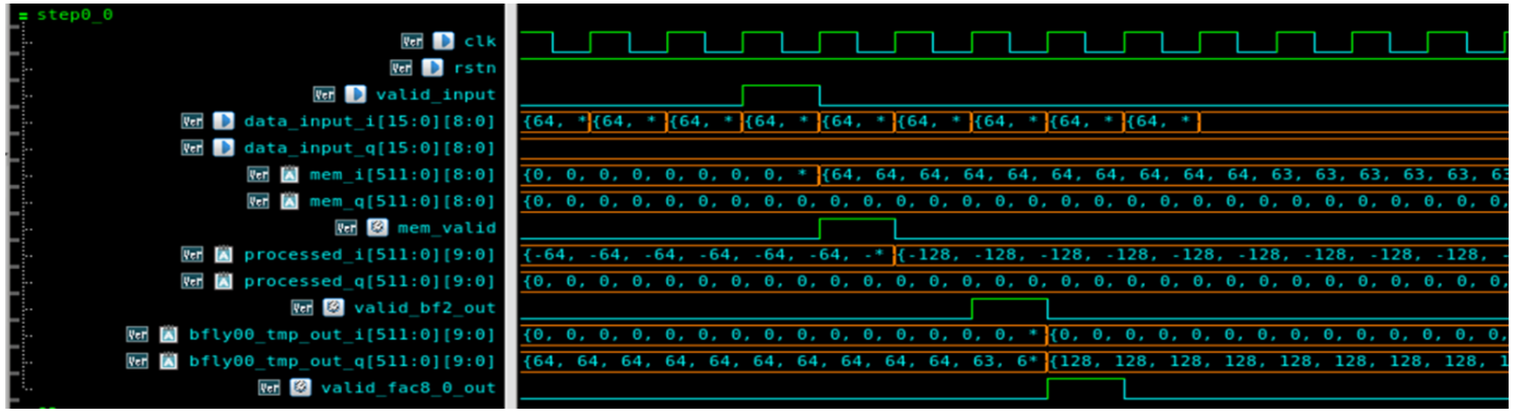
SQNR과 CBFP

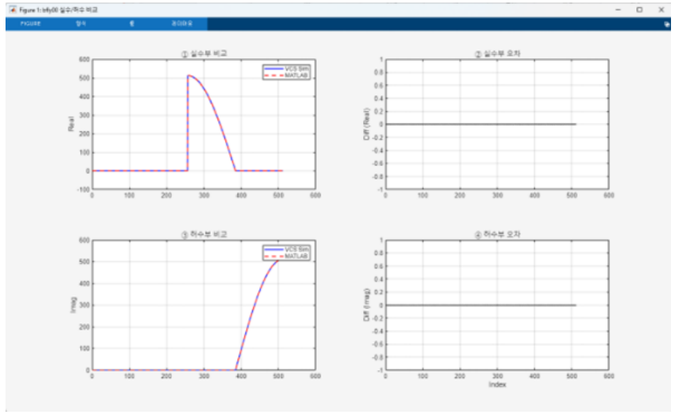
CBFP는 고정소수점 연산에서 발생하는
양자화 오류를 줄이기 위해 사용

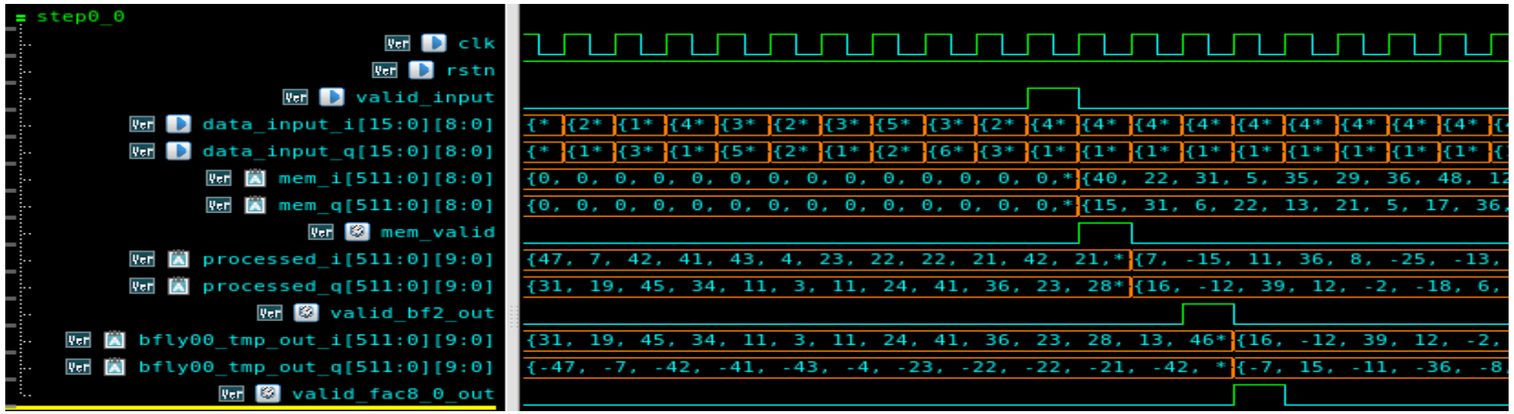
CBFP는 동적 비트폭 조정을 통해
양자화 오류를 줄이고, SQNR을 향상

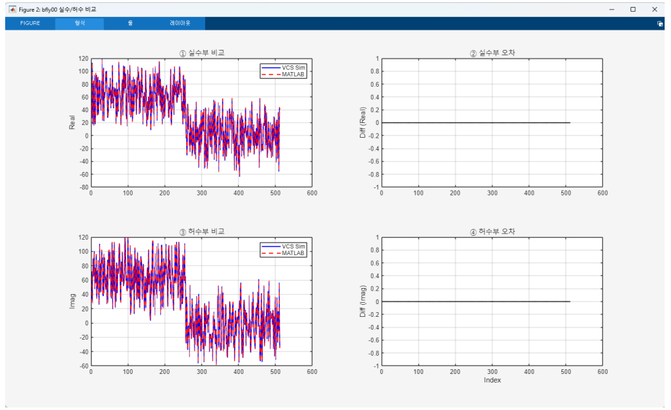
PART 2. Module Simulation & Verification

Module0 – step0_0

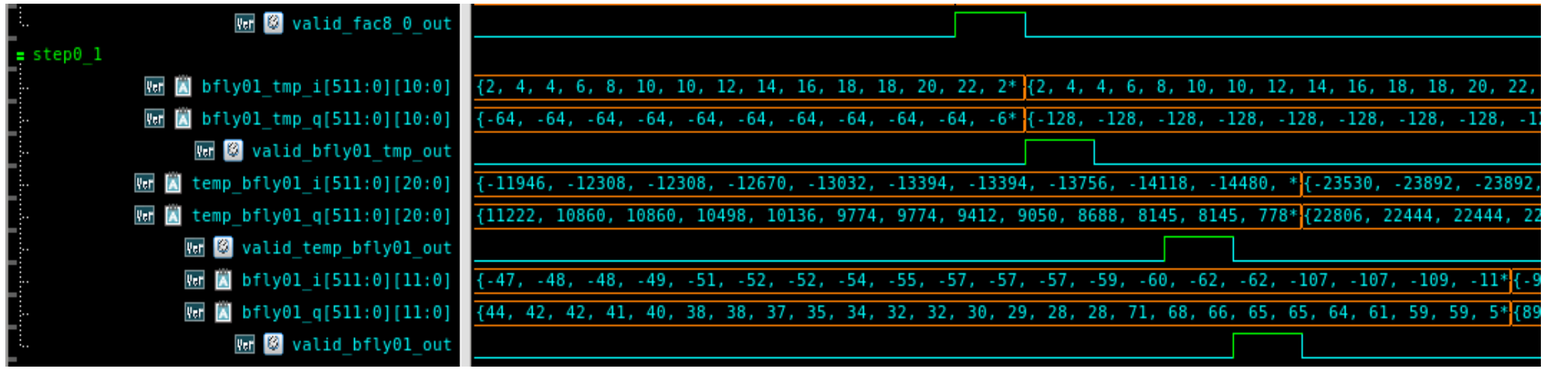


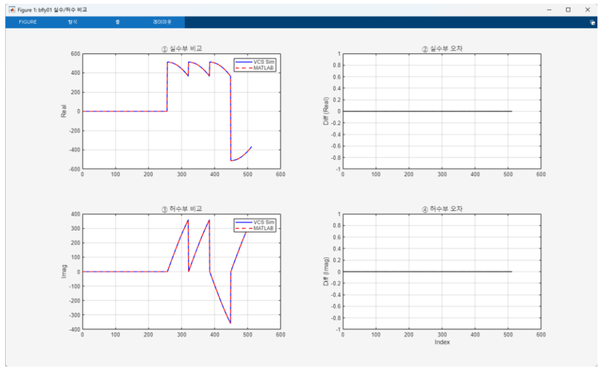


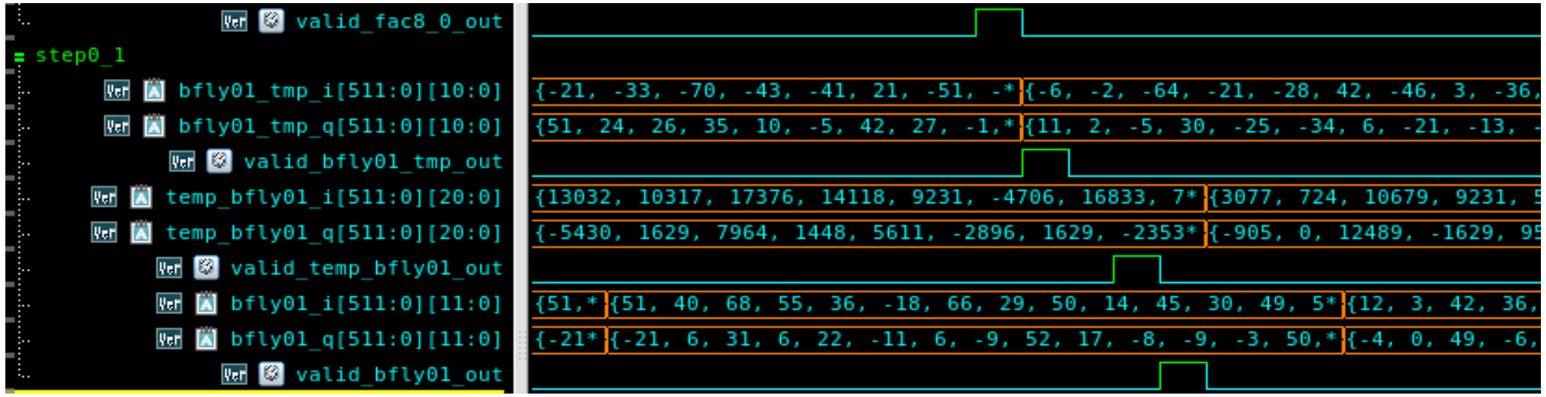


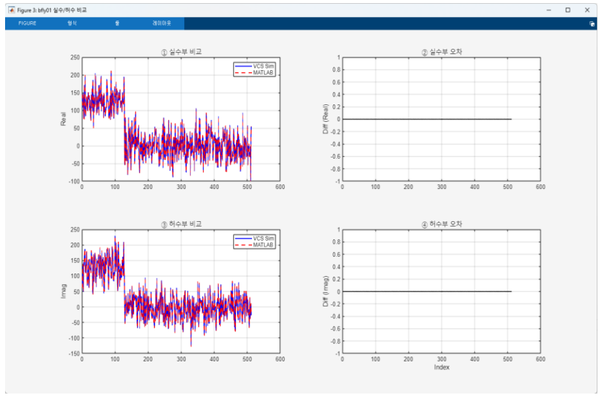


Module0 – step0_1

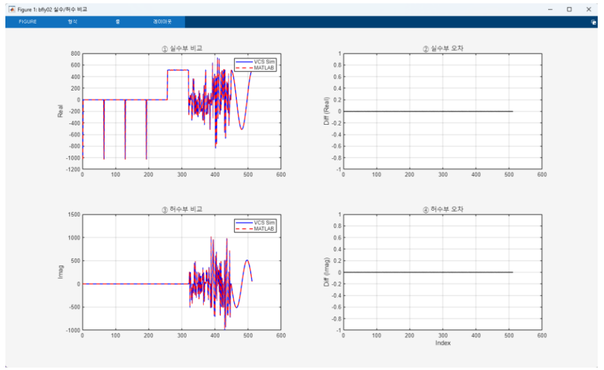


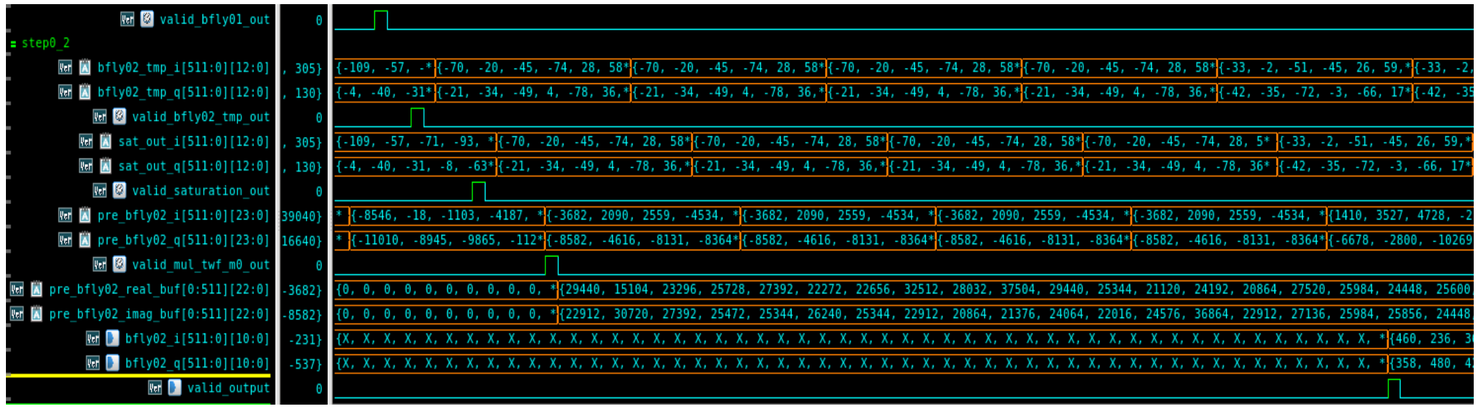


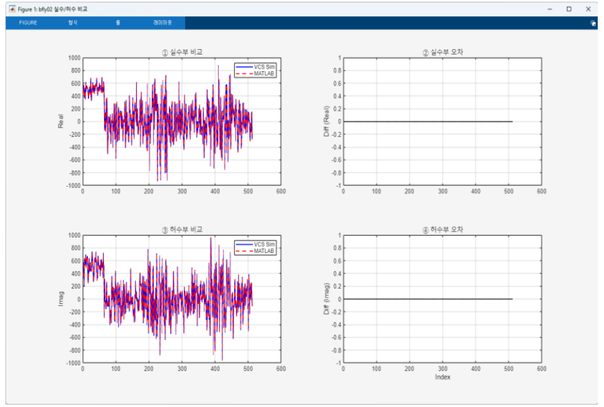




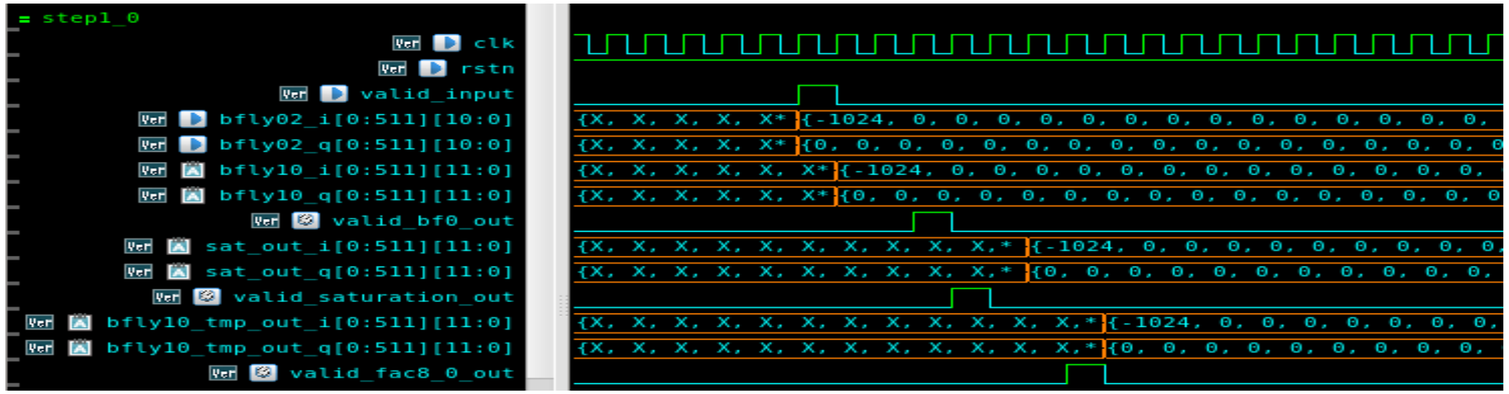
Module0 – step0_2

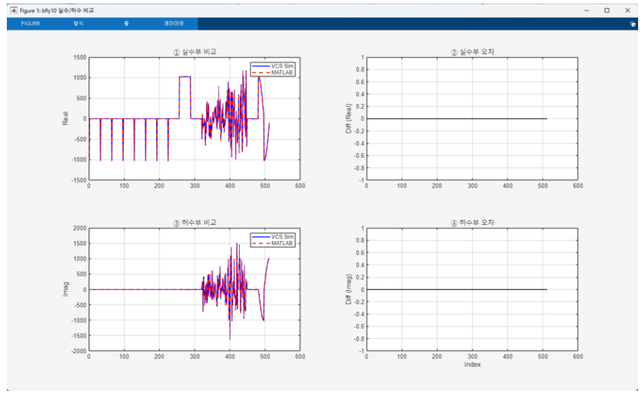


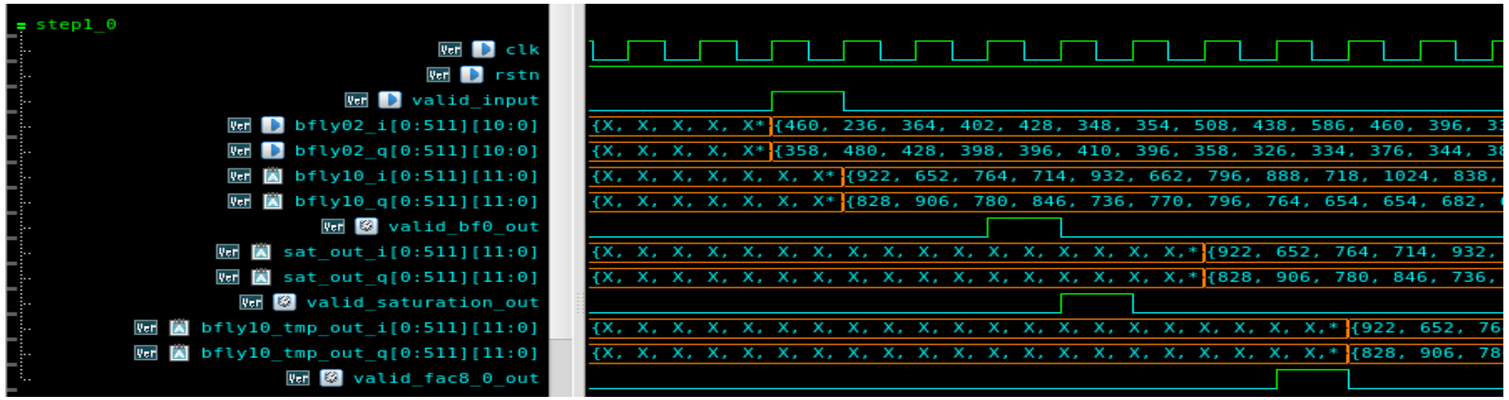


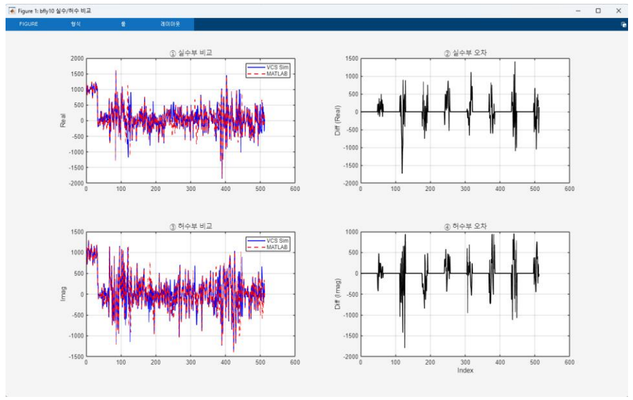


Module1 – step1_0

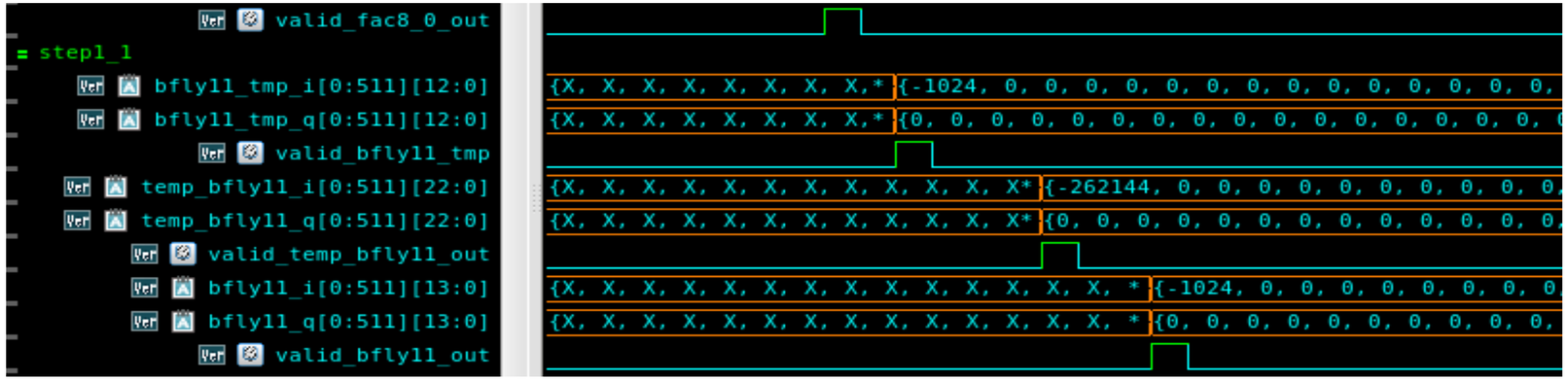


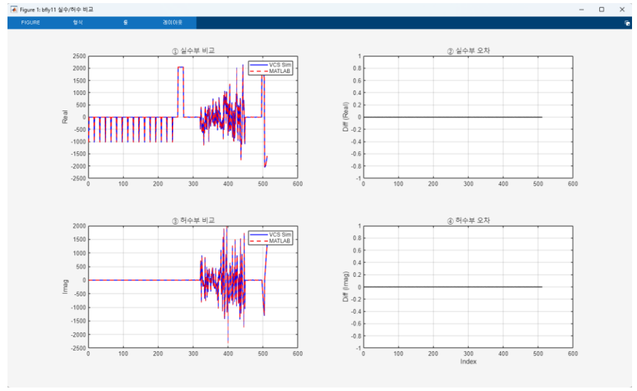


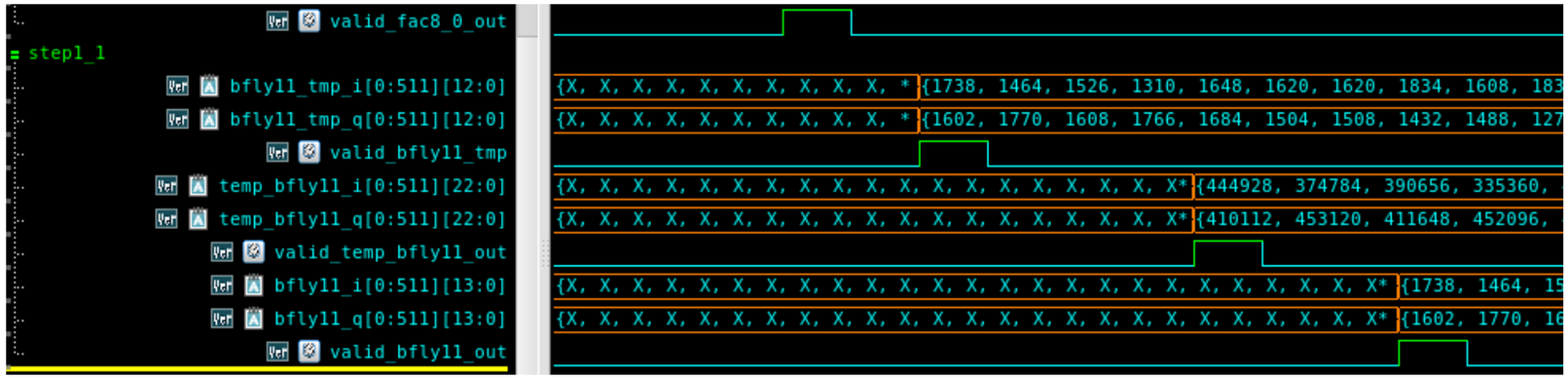


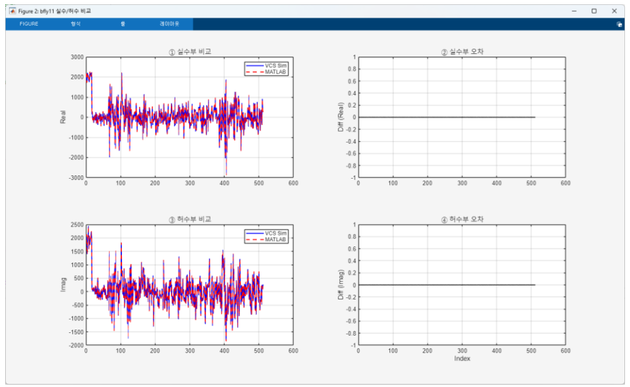


Module1 – step1_1

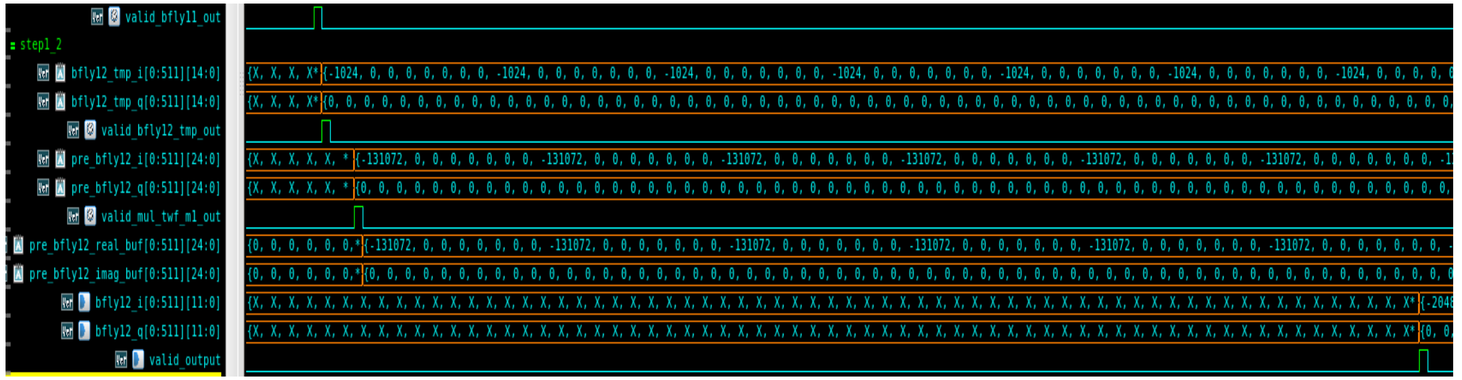


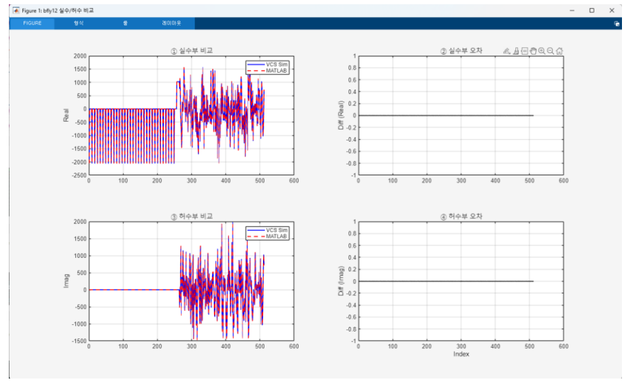


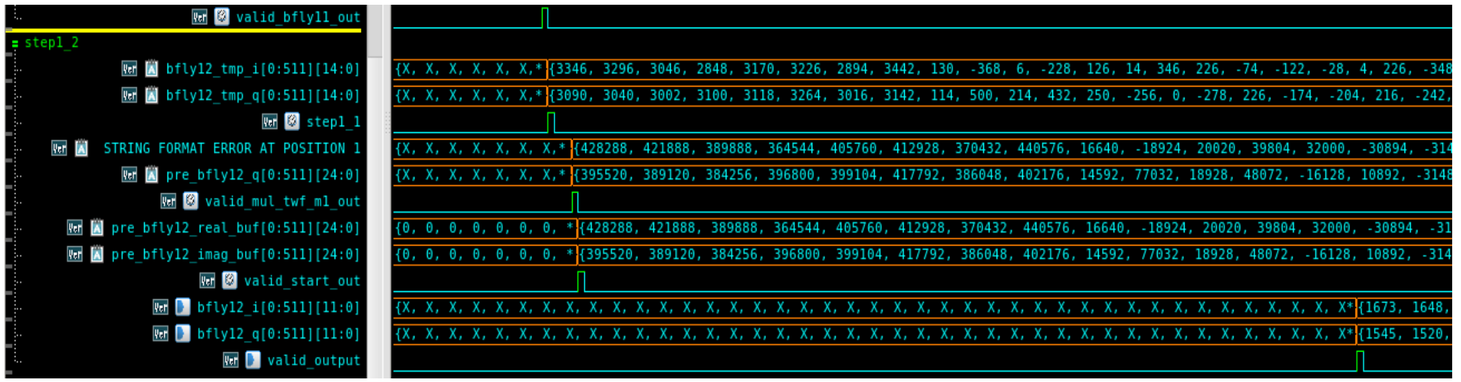


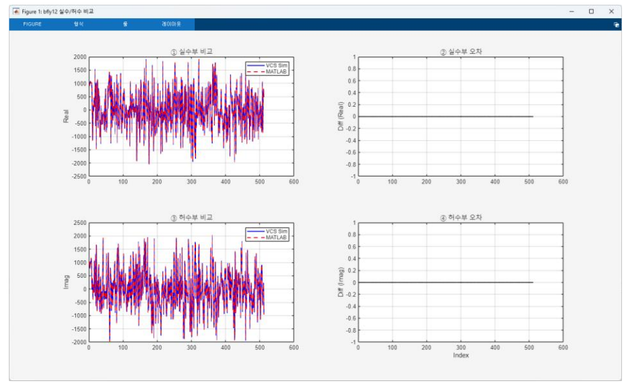


Module1 – step1_2

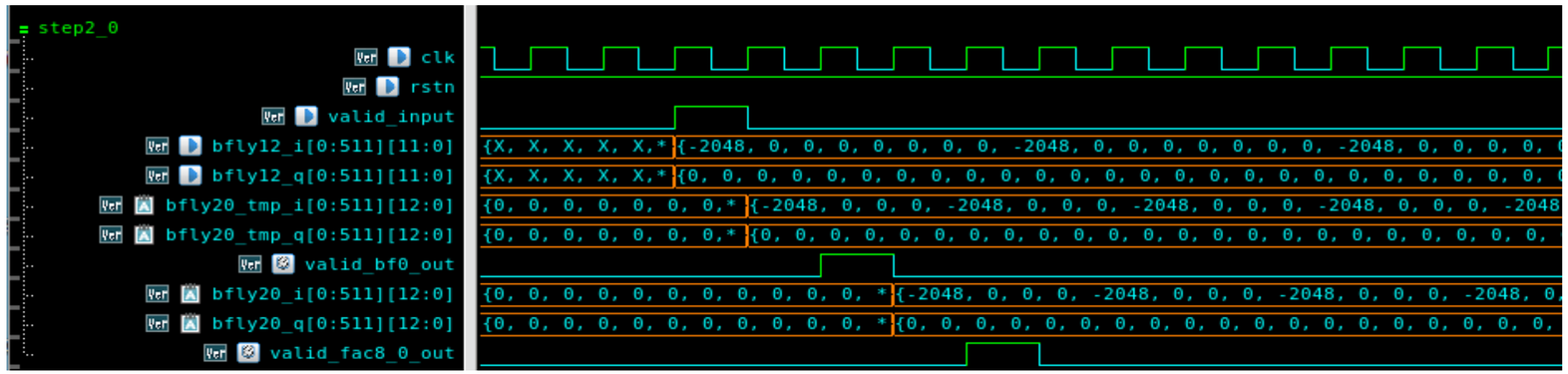


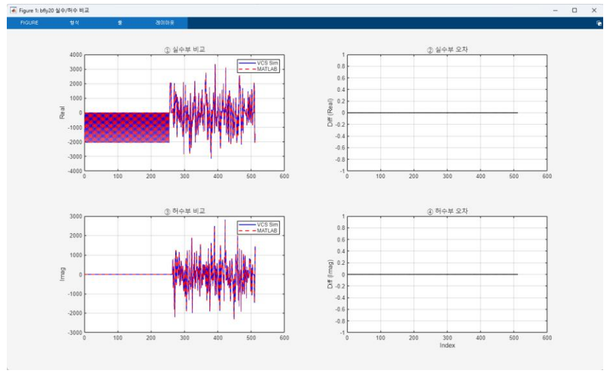


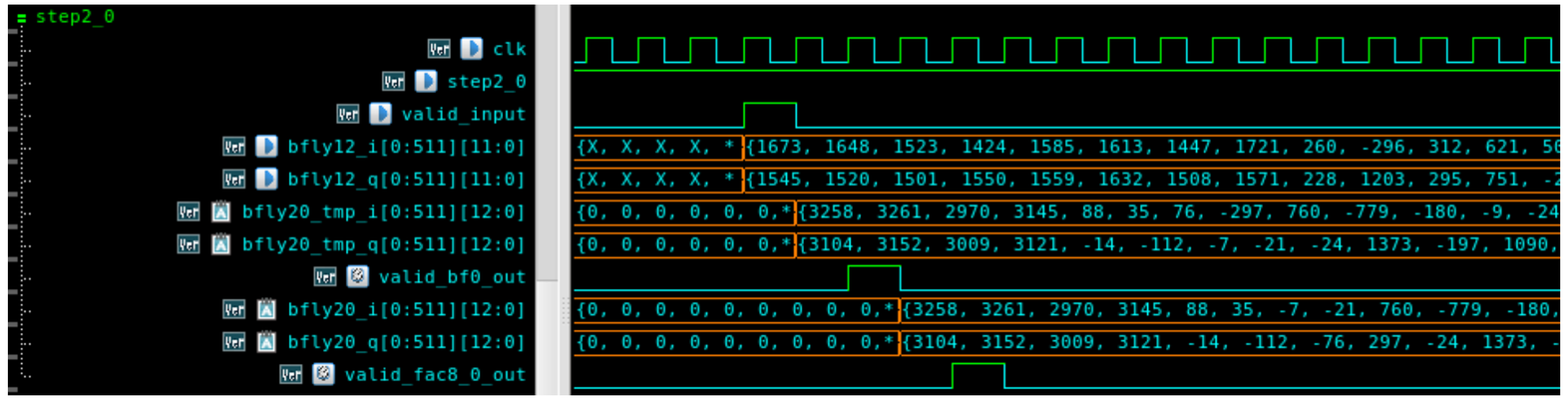


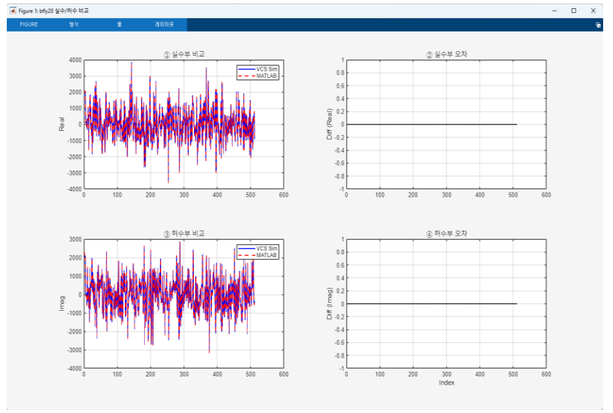


Module2 – step2_0

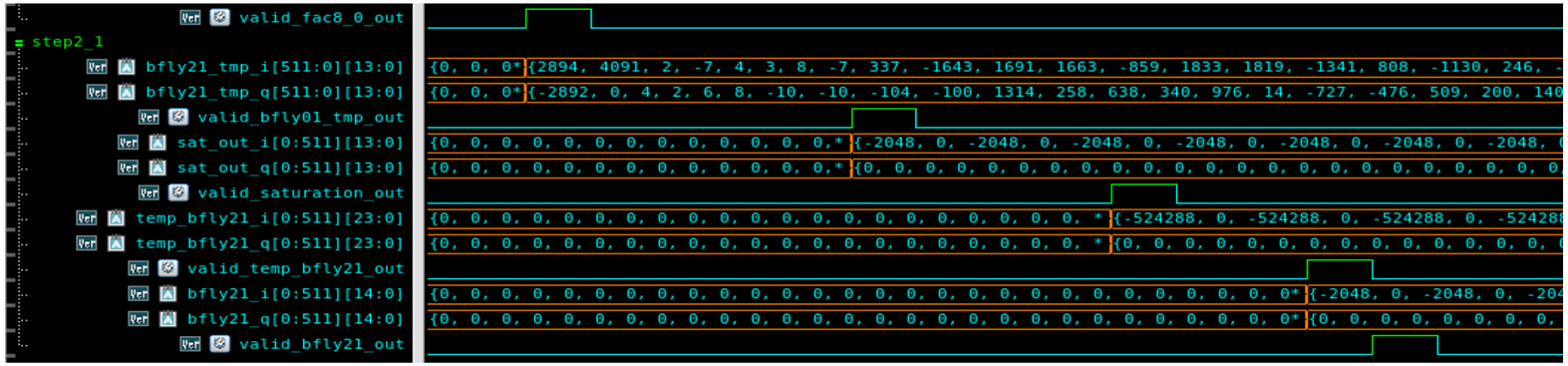


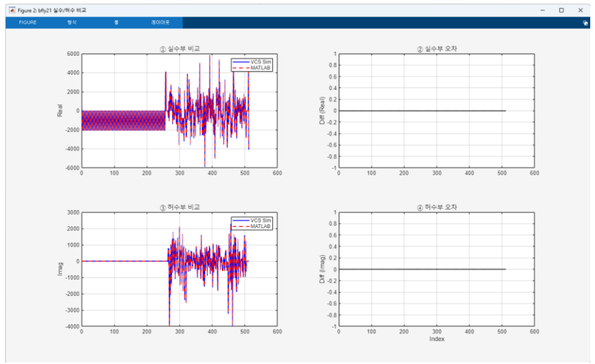


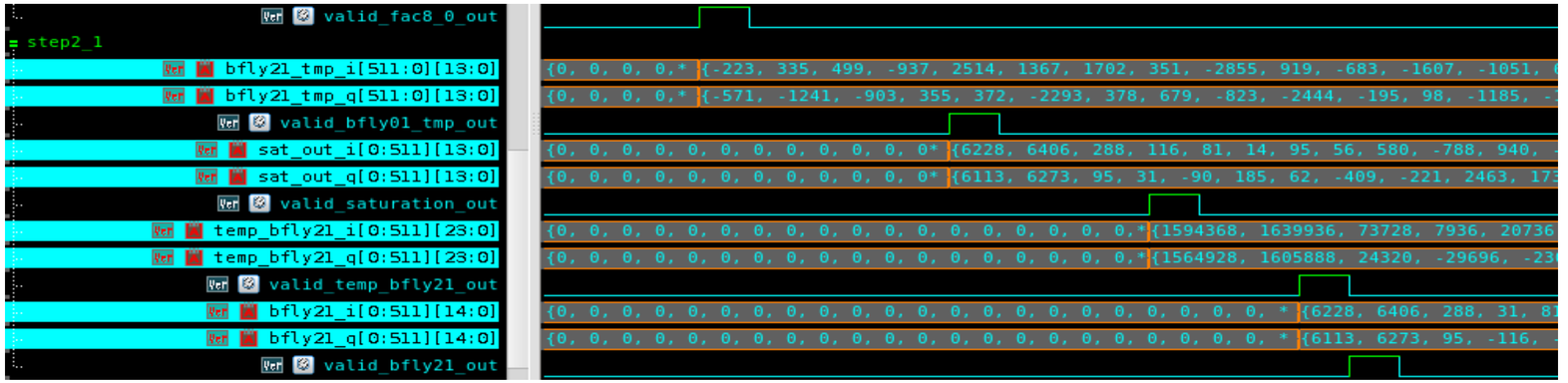


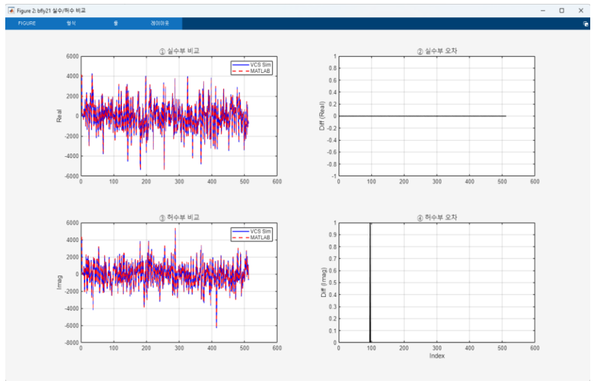


Module2 – step2_1

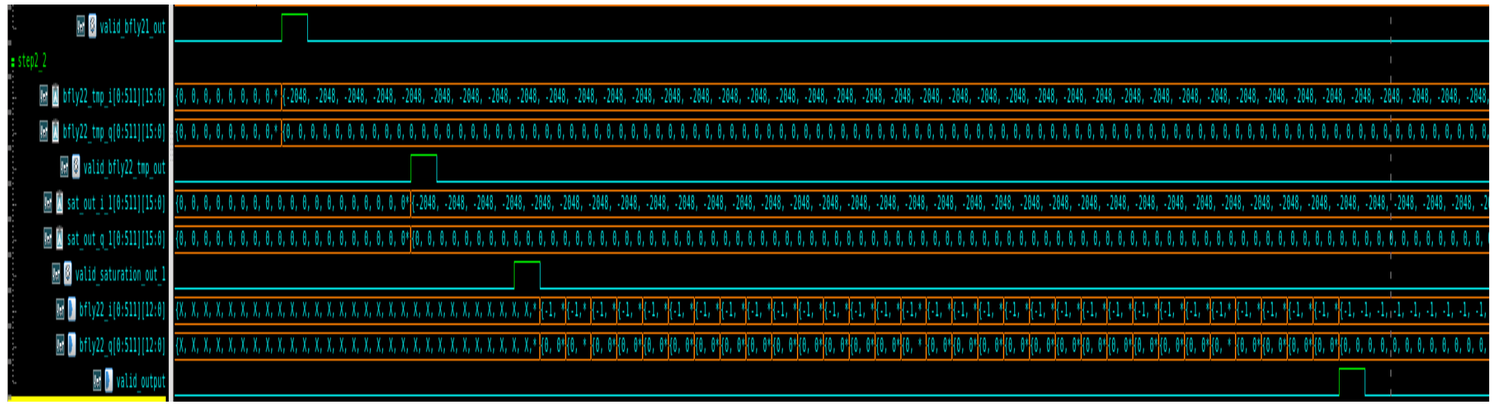


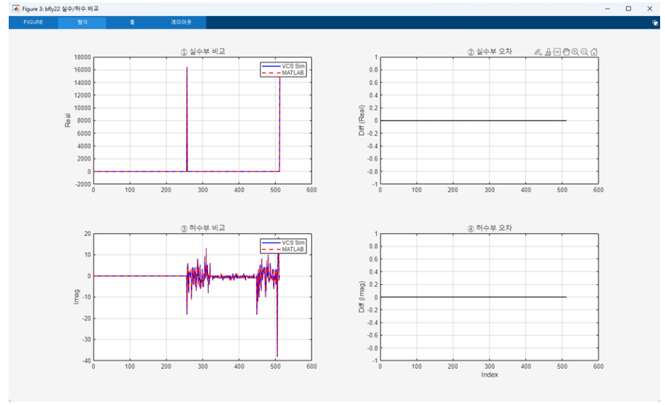


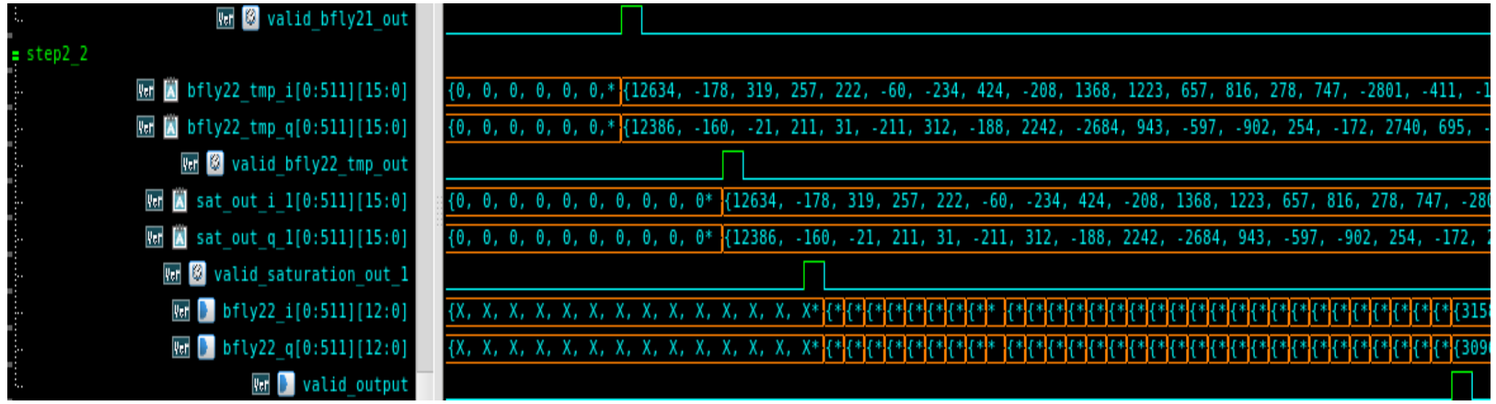


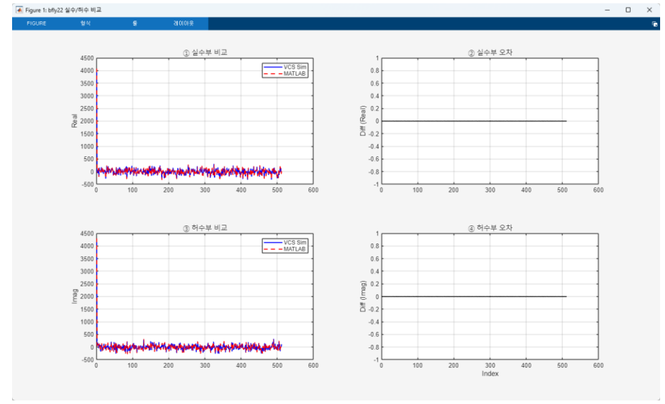


Module2 – step2_2



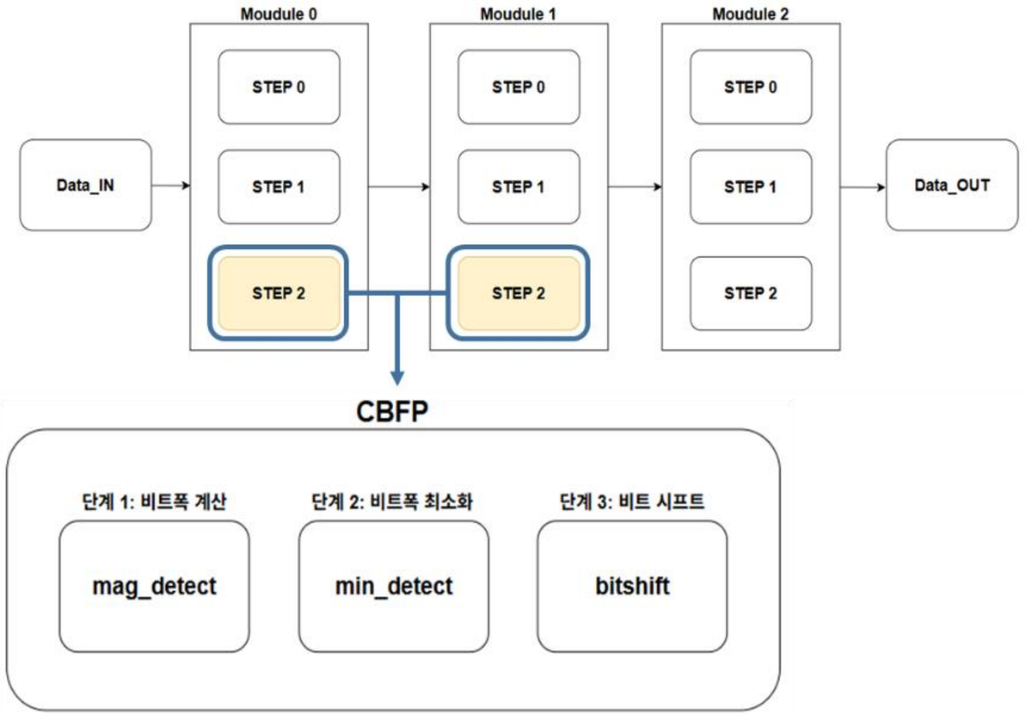






PART 3. CBFP Simulation & Verification

CBFP

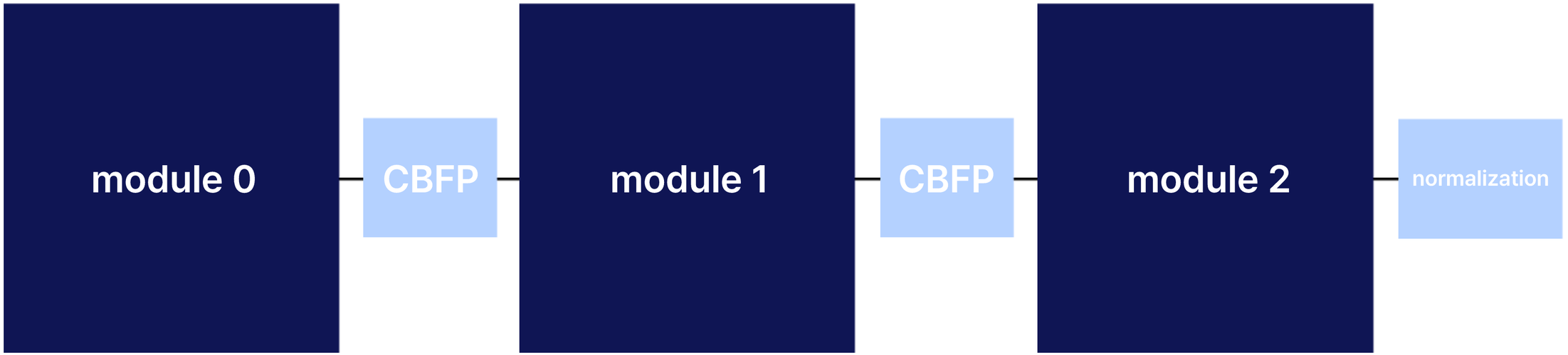


목적: 고정소수점 연산에서 공동 스케일링을 적용해 동적 범위 확장 및 오버플로 방지.

방식: 블록 단위로 입력 데이터의 최소 Headroom 탐색 → Shift 스케일링 적용

장점: 정밀도 유지 + 하드웨어 효율성 확보, FFT 등 대규모 연산에서 메모리와 연산자 절약

CBFP



8개 블록
블록당 data 64개

64개 블록
블록당 data 8개

CBFP

MSB == 1

```
if ((d
    case
        23'b11111111111111111111111: headroom <= 22;
        23'b1111111111111111111111?: headroom <= 21;
        23'b111111111111111111111??: headroom <= 20;
        23'b11111111111111111111???: headroom <= 19;
        23'b1111111111111111111????: headroom <= 18;
        23'b111111111111111111?????: headroom <= 17;
        23'b11111111111111111??????: headroom <= 16;
        23'b1111111111111111???????: headroom <= 15;
        23'b111111111111111????????: headroom <= 14;
        23'b11111111111111?????????: headroom <= 13;
        23'b1111111111111??????????: headroom <= 12;
        23'b111111111111???????????: headroom <= 11;
        23'b11111111111????????????: headroom <= 10;
        23'b1111111111?????????????: headroom <= 9;
        23'b111111111??????????????: headroom <= 8;
        23'b11111111???????????????: headroom <= 7;
        23'b1111111????????????????: headroom <= 6;
        23'b111111?????????????????: headroom <= 5;
        23'b11111??????????????????: headroom <= 4;
        23'b1111???????????????????: headroom <= 3;
        23'b111????????????????????: headroom <= 2;
        23'b11?????????????????????: headroom <= 1;
        23'b1??????????????????????: headroom <= 0;
    en
    ov
end el
```

MSB == 0

```
    case
        23'b00000000000000000000000: headroom <= 22;
        23'b00000000000000000000000?: headroom <= 21;
        23'b00000000000000000000000??: headroom <= 20;
        23'b00000000000000000000000???: headroom <= 19;
        23'b00000000000000000000000????: headroom <= 18;
        23'b00000000000000000000000?????: headroom <= 17;
        23'b00000000000000000000000??????: headroom <= 16;
        23'b00000000000000000000000???????: headroom <= 15;
        23'b00000000000000000000000????????: headroom <= 14;
        23'b00000000000000000000000?????????: headroom <= 13;
        23'b00000000000000000000000??????????: headroom <= 12;
        23'b00000000000000000000000???????????: headroom <= 11;
        23'b00000000000000000000000????????????: headroom <= 10;
        23'b00000000000000000000000?????????????: headroom <= 9;
        23'b00000000000000000000000??????????????: headroom <= 8;
        23'b00000000000000000000000???????????????: headroom <= 7;
        23'b00000000000000000000000????????????????: headroom <= 6;
        23'b00000000000000000000000?????????????????: headroom <= 5;
        23'b00000000000000000000000??????????????????: headroom <= 4;
        23'b00000000000000000000000???????????????????: headroom <= 3;
        23'b00000000000000000000000????????????????????: headroom <= 2;
        23'b00000000000000000000000?????????????????????: headroom <= 1;
        23'b00000000000000000000000??????????????????????: headroom <= 0;
        default: headroom <= 23;
    endcase
    over <= 1;
```

구현내용

과정	설명	구현 방법
mag_detect	headroom을 찾는 단계	MUX(case)
min_detect	최솟값을 찾는 단계	조합 논리
shift	스케일링 단계	shift 연산자(<<< , >>>)

CBFP

BLOCK

12bit

Decimal	Binary	headroom
-20	1111 1110 1100	6
-10	1111 1111 0110	7
0	0000 0000 0000	11
10	0000 0000 1010	7
20	0000 0001 0100	6
30	0000 0001 1110	6
40	0000 0010 1000	5
50	0000 0011 0010	5

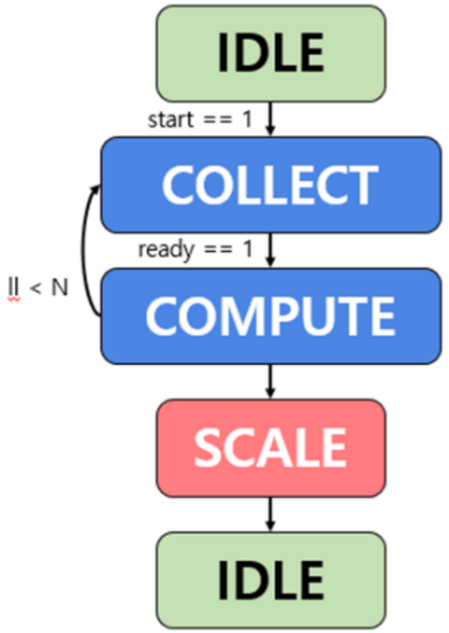
CBFP

7bit

Binary
1 10 1100
1 11 0110
0 00 0000
0 00 1010
0 01 0100
0 01 1110
0 10 1000
0 11 0010

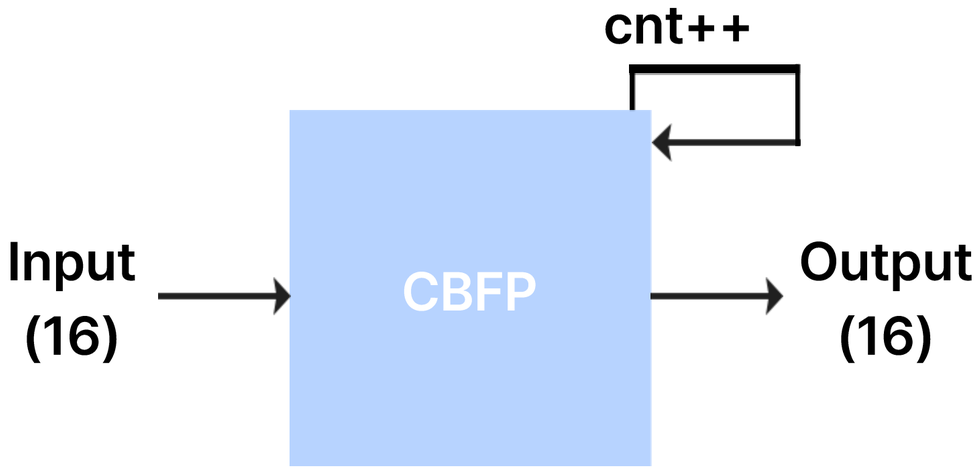
CBFP02 23bit를 11bit로 스케일링 하기

CBFP12 25bit를 12bit로 스케일링 하기



512 / FSM, CNT

V1

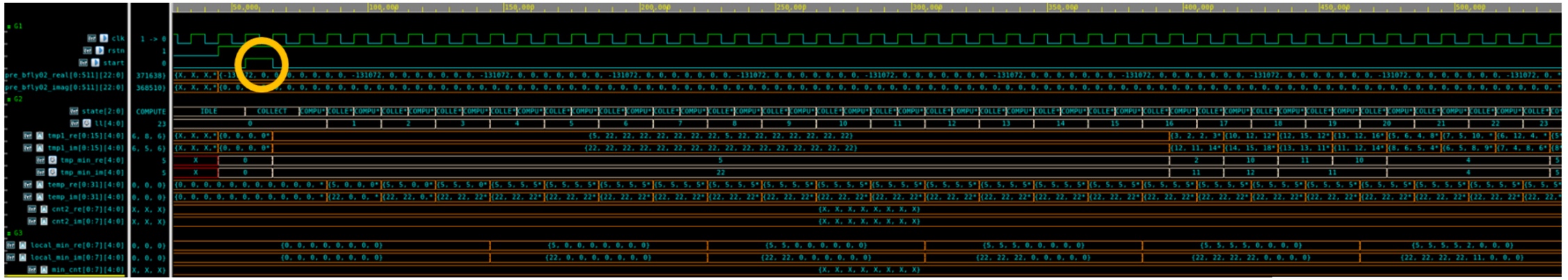


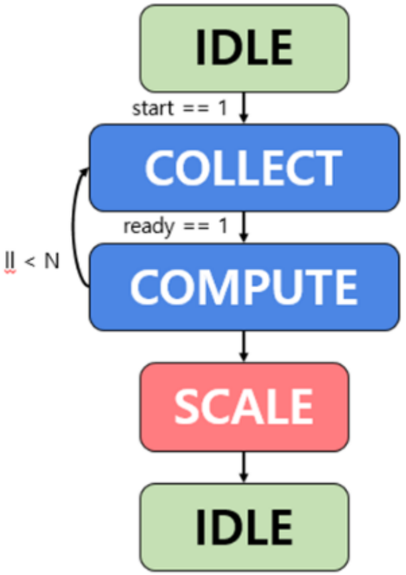
16 / CNT

V2

512 / FSM, CNT → Input: cosine(pre_bfly02)

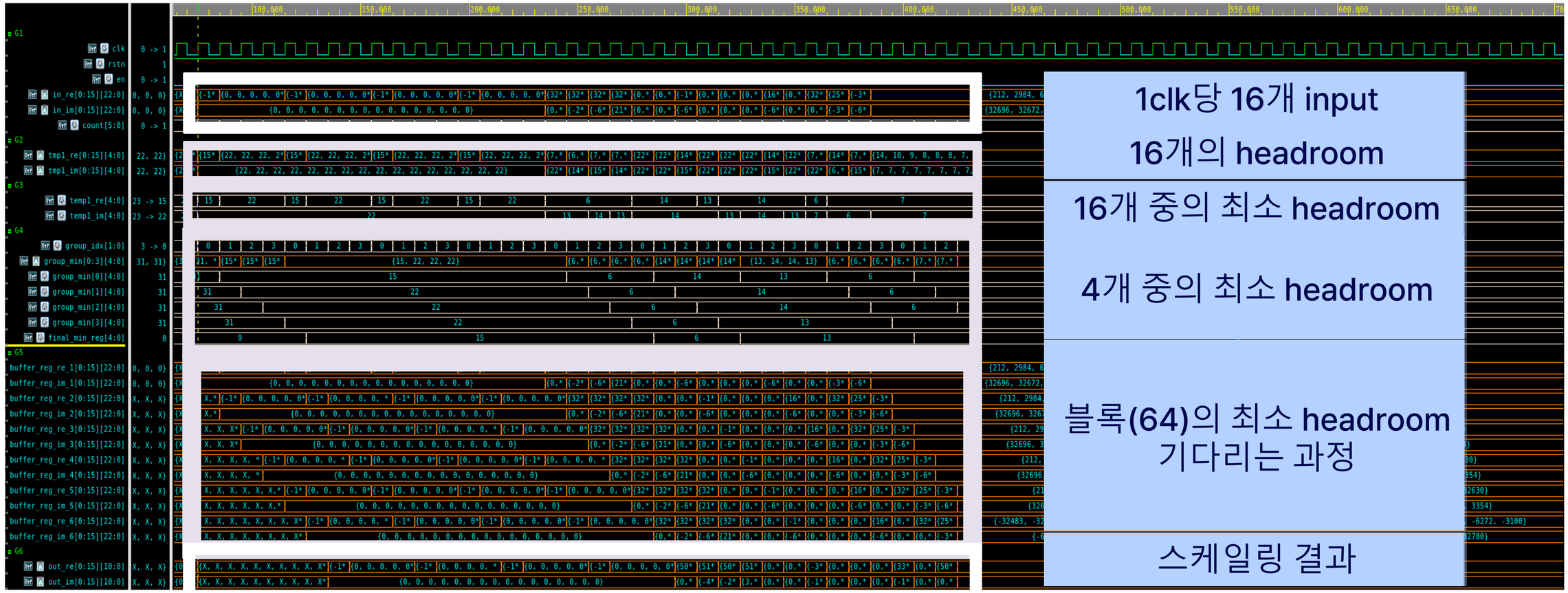
16개씩 처리





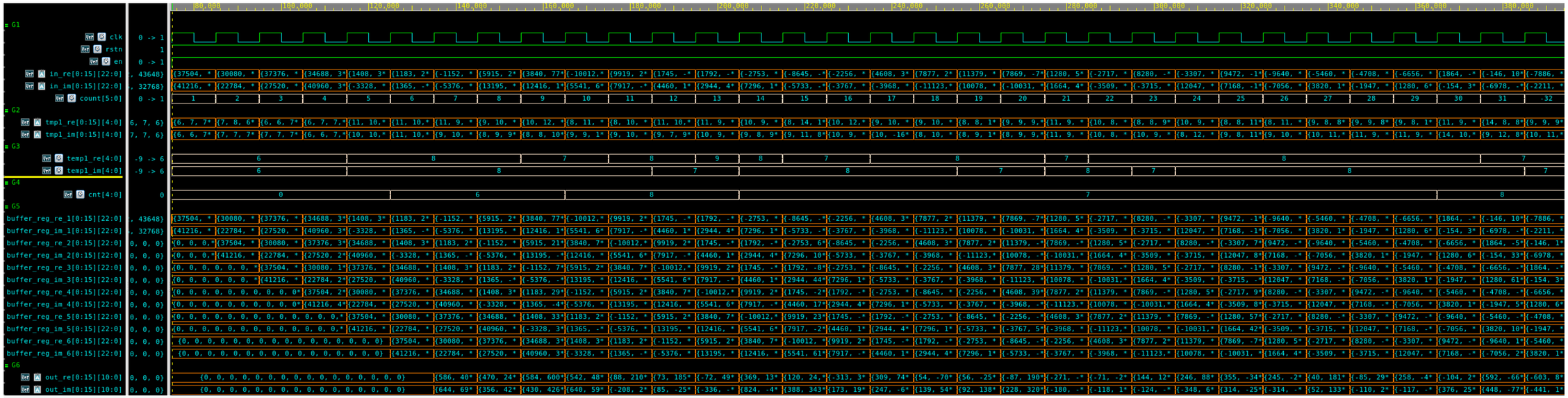
Latency: 68clk

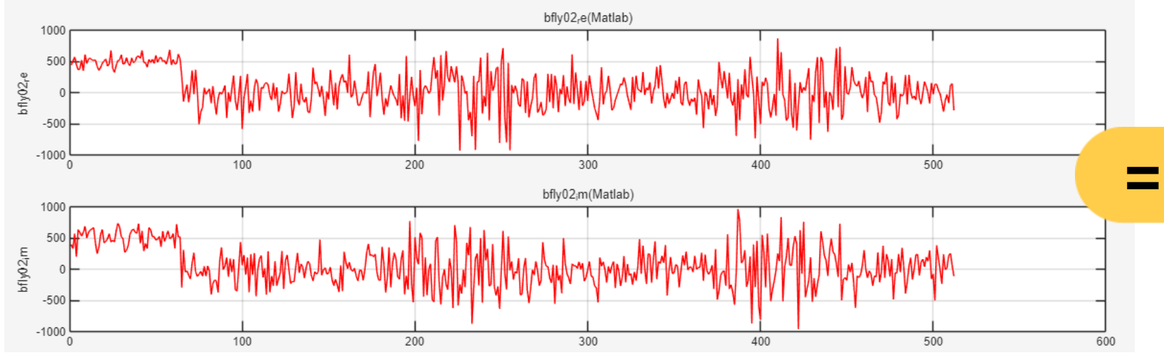
16 / CNT → Input: cosine(pre_bfly02)



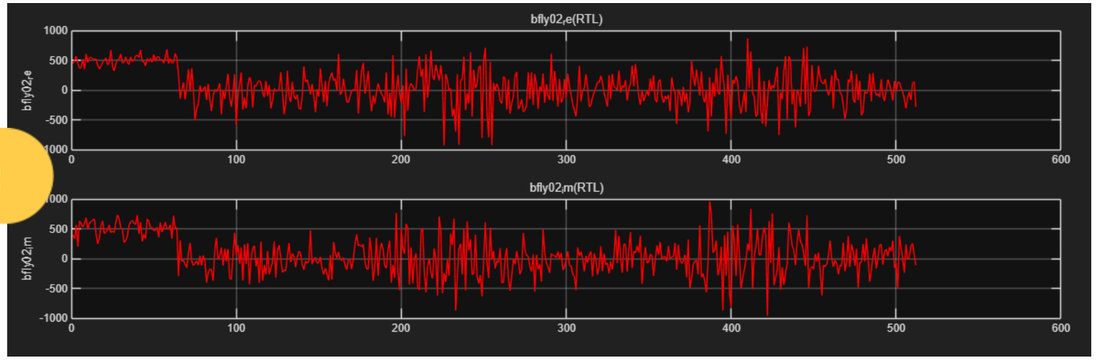
Latency: 38clk

16 / CNT → Input: random(pre_bfly02)

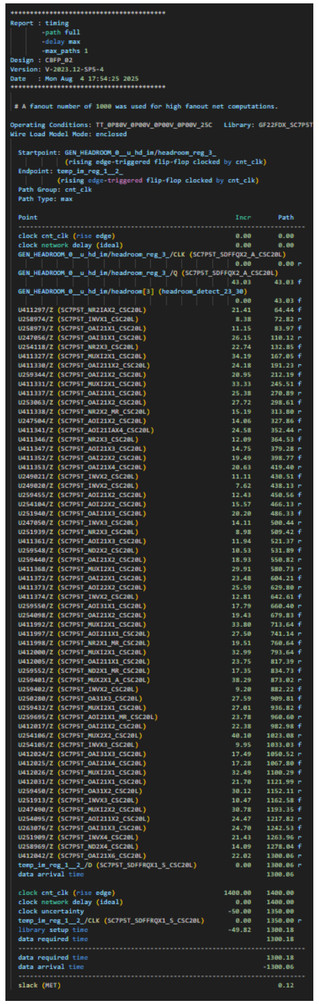




=



CBFP02 AREA

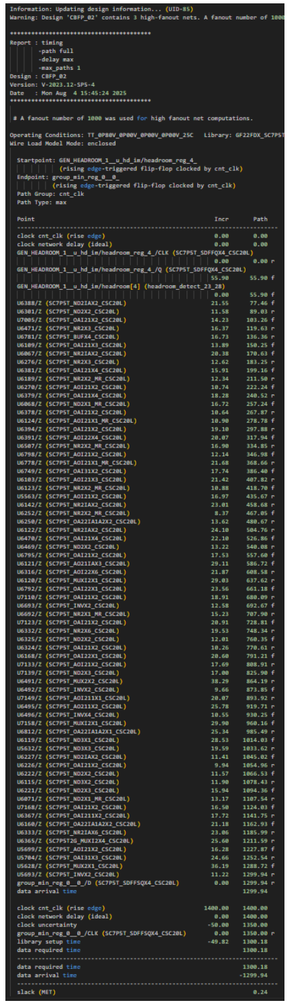


```
GF22FDX_SC7P5T_116CPP_BASE_CSC20L_TT_0P80V_0P00V_0P00V_0P00V_0P00V
PP_BASE_CSC20L_TT_0P80V_0P00V_0P00V_0P00V_25C.db)

Number of ports:                     41236
Number of nets:                     241535
Number of cells:                    216783
Number of combinational cells:      204709
Number of sequential cells:          11974
Number of macros/black boxes:            0
Number of buf/inv:                   48747
Number of references:                  270

Combinational area:           64211.568428
Buf/Inv area:                  8959.956063
Noncombinational area:        19120.442858
Macro/Black Box area:             0.000000
Net Interconnect area:     undefined  (No wire load specified)

Total cell area:              83332.011285
Total area:                undefined
```

512 / FSM, CNT

```
GF22FDX_SC7P5T_116CPP_BASE_CSC20L_TT_0P80V_0P00V_0P00V_0P00V_0P00V
PP_BASE_CSC20L_TT_0P80V_0P00V_0P00V_0P00V_25C.db)

Number of ports:                      2272
Number of nets:                      15898
Number of cells:                     14085
Number of combinational cells:        9013
Number of sequential cells:           5002
Number of macros/black boxes:            0
Number of buf/inv:                    2296
Number of references:                  182

Combinational area:            3681.561600
Buf/Inv area:                   938.625587
Noncombinational area:         9503.879788
Macro/Black Box area:             0.000000
Net Interconnect area:     undefined  (No wire load specified)

Total cell area:              13185.441388
Total area:                undefined
```

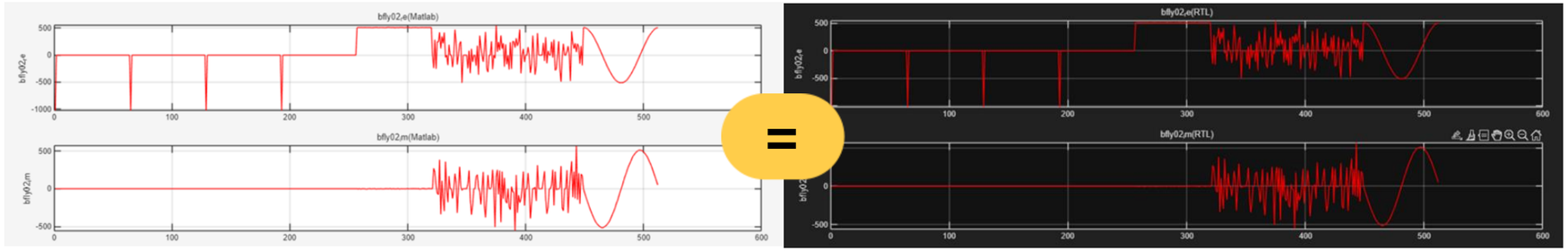
16 / CNT



CBFP 결과

CBFP02

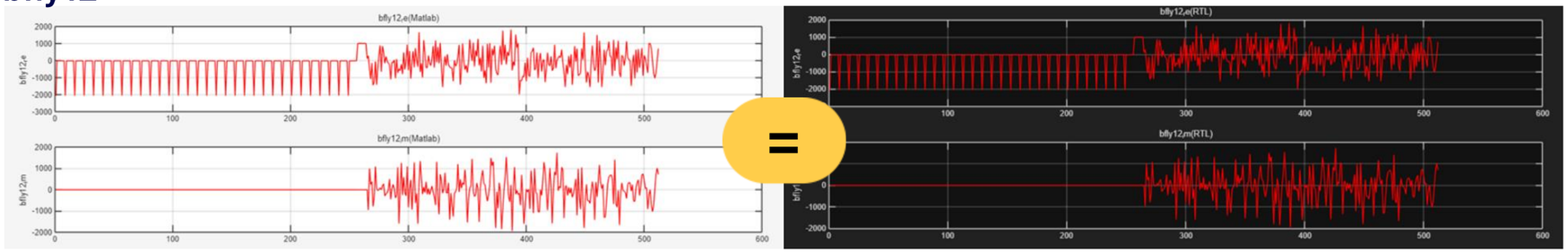
bfly02 (cosine)



<Fixed Model> <RTL>

CBFP12

bfly12



<Fixed Model> <RTL>

PART 4. Synthesis

Setup Timing Check

- SQNR (Set up Timing 체크에서 slack 0.54ps로 타이밍 요구사항 만족(MET)

```
clock cnt_clk (rise edge)                                  1400.00    1400.00
clock network delay (ideal)                                   0.00    1400.00
clock uncertainty                                           -50.00    1350.00
u1_module0/u1_cbfp02/temp_im_reg_0__1_/CLK (SC7P5T_SDFFRQX1_S_CSC20L)
                                                              0.00    1350.00 r
library setup time                                          -53.39    1296.61
data required time                                                    1296.61
-------------------------------------------------------------------------------
data required time                                                    1296.61
data arrival time                                                    -1296.07
-------------------------------------------------------------------------------
slack (MET)                                                             0.54
```

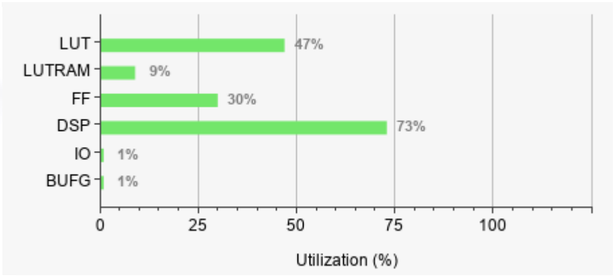
Cell Count & Area Report

- 합성된 회로는 총 2,579,028,72의 셀 면적을 가지며, 약 264만 개의 셀로 구성

```
Library(s) Used:

    GF22FDX_SC7P5T_116CPP_BASE_CSC20L_TT_0P80V_0P00V_0P00V_0P00V_25C (Fi

Number of ports:                      1110800
Number of nets:                       3617970
Number of cells:                      2648318
Number of combinational cells:        1575626
Number of sequential cells:           1061152
Number of macros/black boxes:               0
Number of buf/inv:                     342339
Number of references:                       6

Combinational area:             656223.876991
Buf/Inv area:                    56956.743001
Noncombinational area:         1922804.843500
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:               2579028.720491
Total area:                 undefined
```

Synthesis 현재 상황

Alveo U200 Data Center Accelerator Card

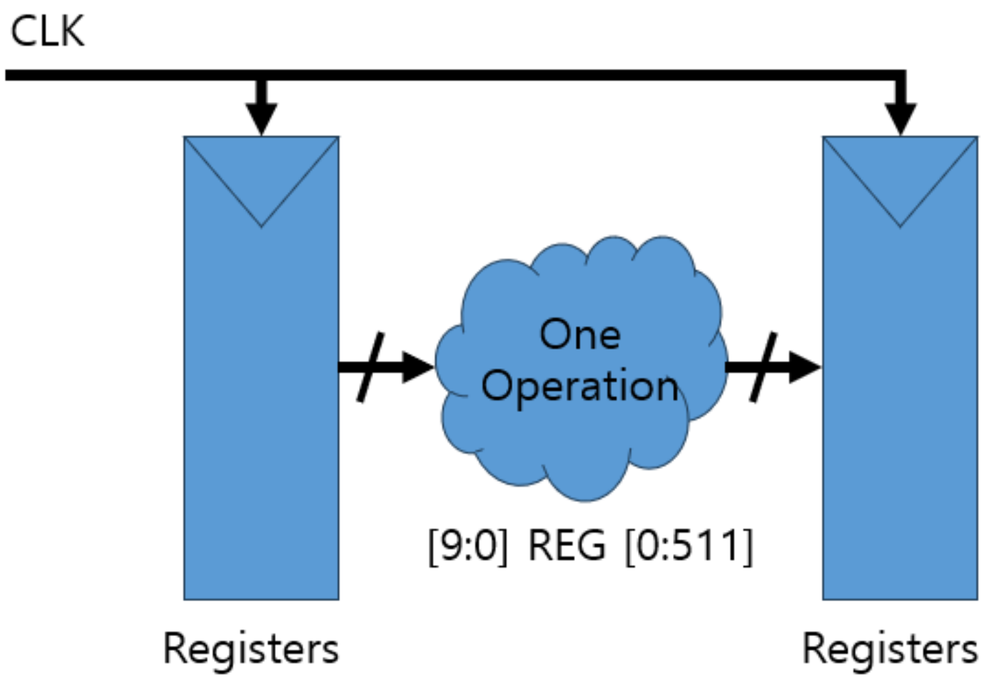



```
# Clock constraint
create_clock -name sys_clk -period 10 [get_ports clk]

# IOSTANDARD 설정 (PACKAGE_PIN은 지정 안 함)
set_property IOSTANDARD LVCMOS18 [get_ports clk]
```

Timing	
Worst Negative Slack (WNS):	-0.548 ns
Total Negative Slack (TNS):	-600.758 ns
Number of Failing Endpoints:	3199
Total Number of Endpoints:	1705469

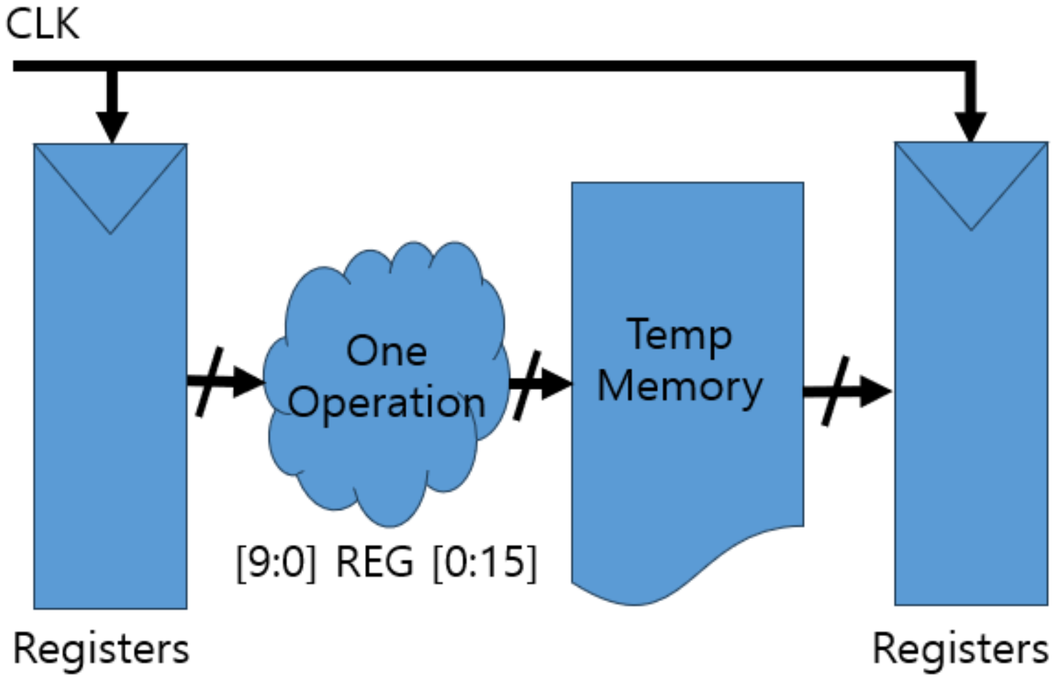
FFT Module0 간 비교

V1



Operates all 512 Points all at once

V2



Latency Increases at steps pass

FFT Module0 간 비교

	V1(512)	V2(16)	%
Area	914317	114247	87.5% 감소
LUT	338854	26673	92.1% 감소
Registers	294883	37203	87.3% 감소
Setup Time	1.52 (Slack Met)	28.66 (Slack Met)	
Latency	17	54	68% 증가

PART 5. Trouble Shooting

Indexsum 계산 수정

문제점

- Saturation 처리 없음
- 비트 폭이 정확하게 처리되지 않음

수정

- 조합 논리 변경
- Saturation 처리 추가

수정 전 코드

```
always_ff @(posedge clk or negedge rstn) begin
    if (!rstn) begin
        for (int kk = 0; kk < 512; kk++) begin
            indexsum_re_reg[kk] <= '0;
            indexsum_im_reg[kk] <= '0;
        end
    end else if (valid_in) begin
        for (int kk = 0; kk < 512; kk++) begin
            indexsum_re_reg[kk] = index1_re[kk] + index2_re[kk];
            indexsum_im_reg[kk] = index1_im[kk] + index2_im[kk];
        end
    end
end
```

수정된 코드

```
always_comb begin
    for (int kk = 0; kk < 512; kk++) begin
        indexsum_re_reg[kk] = index1_re[kk] + index2_re[kk];
        indexsum_im_reg[kk] = index1_im[kk] + index2_im[kk];
        // Saturate if the result exceeds 6-bit range
        if (indexsum_re_reg[kk] > 6'h3F) begin
            indexsum_re_reg[kk] = 6'h3F; )
        end
        if (indexsum_im_reg[kk] > 6'h3F) begin
            indexsum_im_reg[kk] = 6'h3F;
        end
    end
end
```

Shift 연산 음수 값 처리

문제점

- Shift 연산에서 음수 값 처리 부분에서 문제 발생
- Bit shift 연산을 양수만 처리하는 알고리즘 설계

수정

- Shift 할 값이 양수일 경우 왼쪽 shift 연산을 하고, 음수일 경우 오른쪽 shift 연산을 수행하도록 수정

```
//
shift_val_re[i] = 9 - indexsum_re[i];
if (indexsum_re[i] >= 6'd23)
    re_bfly22_pipe[i] <= '0;
else if (shift_val_re[i] > 0)
    re_bfly22_pipe[i] <= bfly22_tmp_re[i] <<< shift_val_re[i];
else
    re_bfly22_pipe[i] <= bfly22_tmp_re[i];
```



```
//
shift_val_re[i] = 9 - indexsum_re[i];
if (indexsum_re[i] >= 6'd23)
    re_bfly22_pipe[i] <= '0;
else if (shift_val_re[i] > 0)
    re_bfly22_pipe[i] <= bfly22_tmp_re[i] <<< shift_val_re[i];
else if (shift_val_re[i] < 0)
    re_bfly22_pipe[i] <= bfly22_tmp_re[i] >>> -shift_val_re[i];
else
    re_bfly22_pipe[i] <= bfly22_tmp_re[i];
```

Shift 연산 음수 값 처리

문제

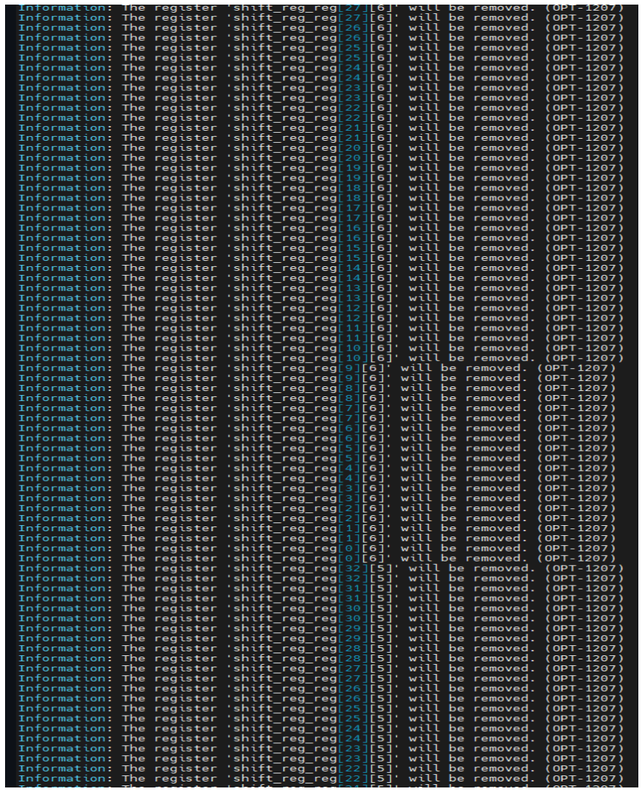
- 상수로 저장할 값을 logic 타입으로 선언하여 합성중 최적화 과정에서 꼭 필요한 레지스터가 제거되는 현상 발생

- 이로 인해 합성이 모두 진행되었음에 도 불구하고, 합성 결과는 비어있게 됨.

```
logic signed [COEFF_WIDTH-1:0] coeffs [0:TAP_NUM-1] = '{
    16'sd0,  -16'sd1,  16'sd1,  16'sd0,  -16'sd1,  16'sd2,  16'sd0,  -16'sd2,
    16'sd2,  16'sd0, -16'sd6, 16'sd8, 16'sd10, -16'sd28, -16'sd14, 16'sd111,
    16'sd196, 16'sd111, -16'sd14, -16'sd28, 16'sd10, 16'sd8, -16'sd6, 16'sd0,
    16'sd2,  -16'sd2,  16'sd0,  16'sd2,  -16'sd1,  16'sd0,  16'sd1,  -16'sd1,
    16'sd0
};
```

수정

- logic이 아닌 parameter로 변경하여 해결



PART 6. 고찰

고찰

- 비트 폭 불일치 문제
  모듈 인스턴스 및 시뮬레이션 과정에서 신호 비트 폭 차이로 디버깅 어려움 발생
  -> 초기 데이터 폭 정의의 중요성 인식

- 설계 스타일 차이, 팀원 간 코딩 스타일 불일치로 통합 과정에서 추가 조율 필요
  -> 코딩 컨벤션 확립 필요성 확인

- 요구사항 분석 보완
  초기 스펙 해석 부족으로 설계 방향 설정 지연 -> 요구사항 정리 및 문서화 강화 필요

감사합니다.