RADIX-22 DIF FFT 설계

고속 디지털 신호처리를 위한 병렬 FFT 아키텍처

TEAM11 - 정민교 엄찬하 신상학 임재홍



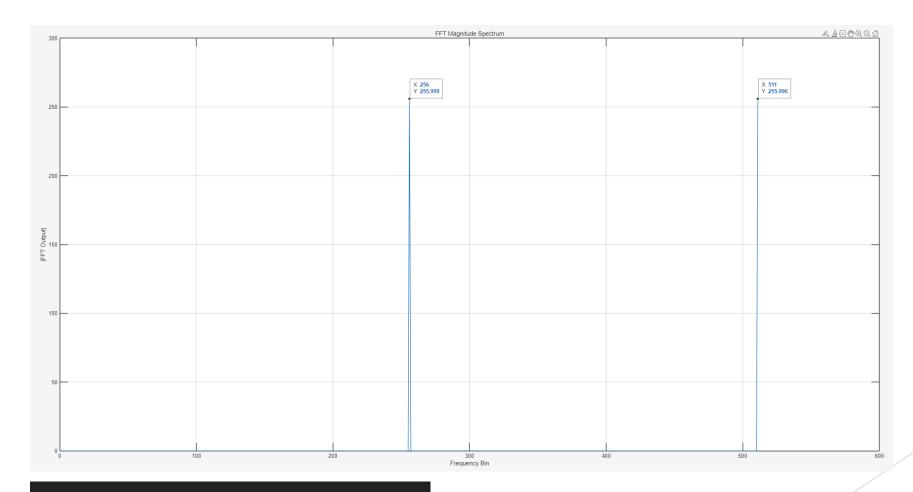
• 목차

1. Matlab 결과

2. Module Diagram

3. 논문 수식 설명

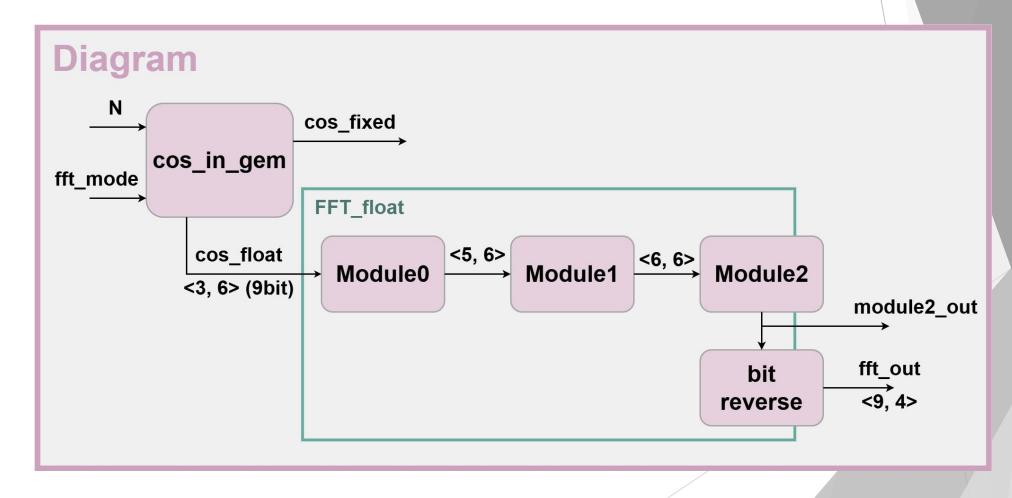
1. Matlab 결과



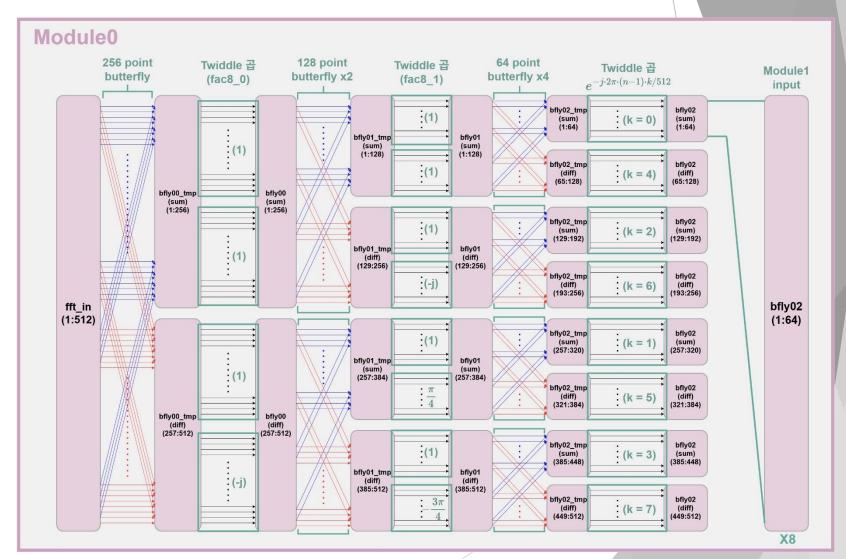
$$\cos(2\pi f_0 n/N) = rac{1}{2} \left(e^{j2\pi f_0 n/N} + e^{-j2\pi f_0 n/N}
ight)$$

-> 중심 주파수: N/2 -> bin 256 = 256 / bin 511 = 256

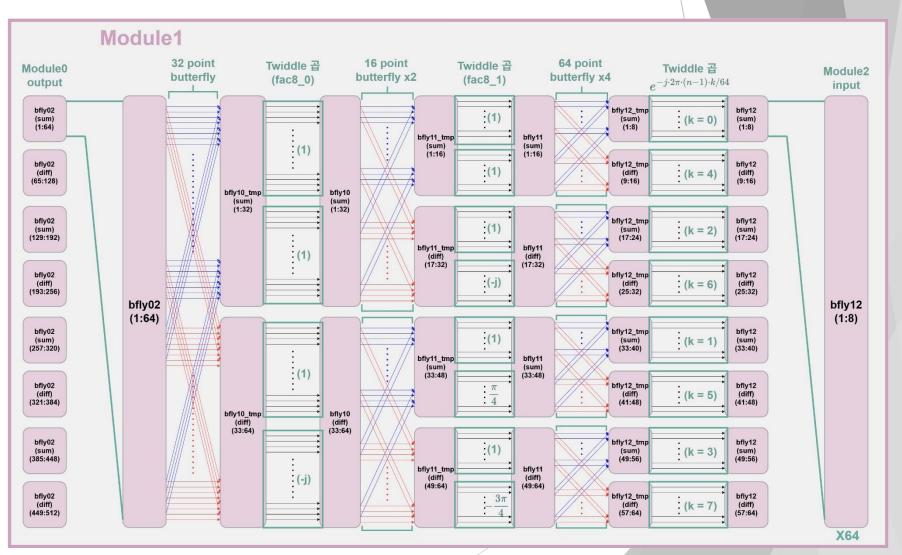
Module Diagram



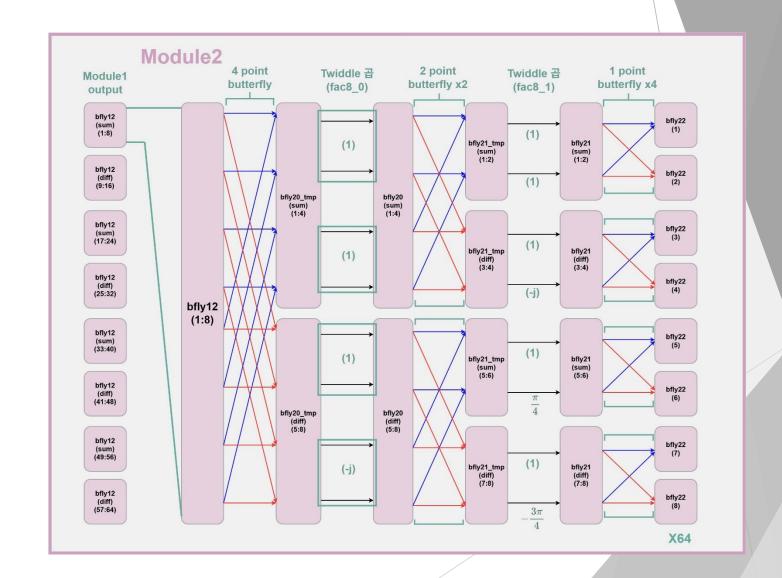
- Module Diagram
 - Module0



- Module Diagram
 - Module1



- Module Diagram
 - Module2



Twiddle Factor

1. DFT 정의식:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk}$$

3. 공통 항 정리:

$$egin{align} &= \sum_{m=0}^{N/2-1} \left[x(2m) W_{N/2}^{mk} + x(2m+1) W_{N/2}^{mk} W_N^k
ight] \ &= \sum_{n=0}^{N/2-1} W_{N/2}^{mk} \cdot \left[x(2m) + x(2m+1) W_N^k
ight]
onumber \end{split}$$

(여기서
$$lpha=x(2m)+x(2m+1)W_N^k$$
)

2. 짝수/홀수로 나눠서 전개:

$$=\sum_{m=0}^{N/2-1}x(2m)W_N^{2mk}+\sum_{m=0}^{N/2-1}x(2m+1)W_N^{(2m+1)k}$$

4. 요약 정리된 표현:

$$X_{ ext{even}}(k) = lpha, \quad X_{ ext{odd}}(k) = lpha W_N^k$$
 $X(k) = lpha + lpha W_N^k$

Twiddle Factor

$$W_N^k=e^{-jrac{2\pi k}{N}}$$
 \Longrightarrow

$$X(k) = \sum_{n=0}^{511} x(n) \cdot e^{-j2\pi kn/512}$$

Bit Reverse (K)

Factor	정	5
---------------	---	---

원래 인덱스 (10진)	원래 인덱스 (2진)	비트 리버스 (2진)	리버스된 인덱스 (10진)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

이름	길이	적용되는 위치	설명
fac8_0	4	step0_0, step1_0, step2_0	N=4 FFT 기준 twiddle 상수
fac8_1	8	step0_1, step1_1, step2_1	N=8 FFT 기준 twiddle 상수
twf_m0	512	step0_2	MSB 기준 кз 정렬용 위상 곱
twf_m1	64	step1_2	MSB 기준 K2 정렬용 위상 곱

Twiddle Factor

따라서 홀수는 Fac8_1,2를 진행하며 의미 없는 1이 곱해지며, 짝수는 의미 있는 위상 변화가 곱해진다.

fac8_0 = [1, 1, 1,
$$-j$$
];
fac8_1 = [1, 1, 1, $-j$, 1, 0.7071 $-0.7071j$, 1, $-0.7071-0.7071j$];

$$n=\left\langle rac{N}{2}n_1+rac{N}{4}n_2+n_3
ight
angle_N \quad ext{or} \quad n=n_3\cdot 4+n_2\cdot 2+n_1 \qquad \qquad k=\left\langle k_1+2k_2+4k_3
ight
angle_N$$

단계	변수	설명	비트위치
Stage1	N3, k3	가장 상위 비트	비트 8~6
Stage2	N2, k2	중간 비트	비트 5~3
Stage3	N1, k1	하위 비트	비트 2~0

Flow_0

Step	역할	입력 크기	출력 크기	B 용	비고
step0_0	512 → 256×2로 홀 짝 분할 (radix-4)	512	512	fac8_0(ceil(nn/128)) (길이 4)	N=4 FFT, MSB 초기 정 렬용
step0_1	256×2 → 128×4 구 조로 재조합	512	512	fac8_1(ceil(nn/64)) (길이 8)	N=8 FFT 중간 구조 설 계
step0_2	128×4 → 64×8, 마 지막 그룹화	512	512	twf_m0(nn) (K3 기 반 위상 곱)	K3 = MSB 정렬 적용

Flow_1

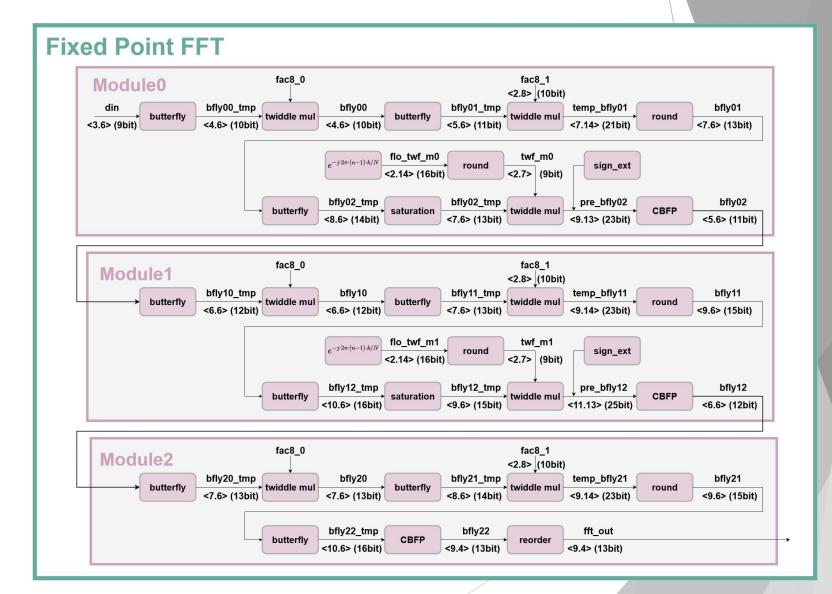
Step	역할	입력 크기	출력 크기	Twiddle Factor 적 용	비고
step1_0	64점 그룹 → 32점 butterfly	512 = 8×64	512 = 8×64	fac8_0(ceil(nn/16)) (길이 4)	
step1_1	32점 그룹 → 16점 butterfly	512 = 16×32	512 = 16×32	fac8_1(ceil(nn/8)) (길이 8)	
step1_2	16점 그룹 → 8점 butterfly	512 = 32×16	512 = 32×16	twf_m1(nn) (K2 기 반 위상 곱)	K2 = MSB 정렬 적용

Flow_2

Step	역할	입력 크기	출력 크기	Twiddle Factor 적 용	비고
step2_0	8점 그룹 → 4점 butterfly	512 = 64×8	512 = 64×8	fac8_0(ceil(nn/2)) (길이 4)	
step2_1	4점 그룹 → 2점 butterfly	512 = 128×4	512 = 128×4	fac8_1(nn) (길이 8, 고정값 적용)	
step2_2	2점 butterfly (최종 단계)	512 = 256×2	512	없음	최종 결과 완성됨

2. fixed point FFT

Module Diagram



3. CBFP

Real / Imgn

```
% CBFP(Convergent Block Floating Point) stage0
cnt1_re = zeros(1,8);
cnt1_im = zeros(1,8);
for ii=1:8
  for jj=1:64
    tmp1_re=mag_detect(real(pre_bfly02(64*(ii-1)+jj)),23);
    tmp1_im=mag_detect(imag(pre_bfly02(64*(ii-1)+jj)),23);
% 블록 내에서 실수부/허수부 시프트 카운트의 최소값을 계속 갱신
% (가장 큰 절댓값에 해당하는 가장 작은 시프트 카운트를 찾는 과정)
    temp1_re=min_detect(jj,tmp1_re,cnt1_re(ii));
    temp1_im=min_detect(jj,tmp1_im,cnt1_im(ii));
    cnt1_re(ii)=temp1_re;
    cnt1_im(ii)=temp1_im;
end
end
```

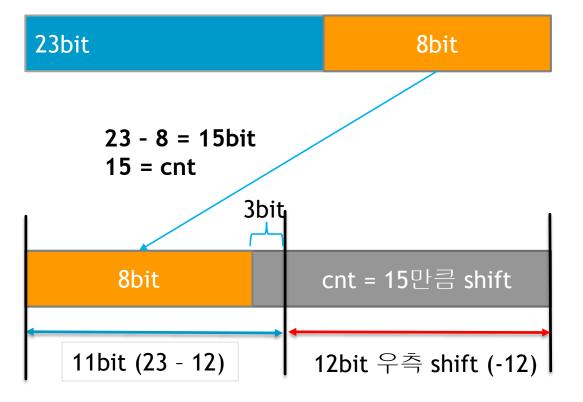
```
% 각 블록에 대해 실수부와 허수부의 시프트 카운트 중 더 작은 값으로 통일
for ii=1:8
                              % 실수부 시프트 카운트가 더 작거나 같으면 그대로 유지
if (cnt1_re(ii)<=cnt1_im(ii))</pre>
 cnt1_re(ii)=cnt1_re(ii);
 cnt1_re(ii)=cnt1_im(ii);
                              % 허수부 시프트 카운트가 더 작으면 살수부 시프트 카운트를 허수부로 맞춤 (더 보수적으로)
for ii=1:8
if (cnt1_im(ii)<=cnt1_re(ii))</pre>
                              % 허수부 시프트 카운트가 더 작거나 같으면 그대로 유지
  cnt1_im(ii)=cnt1_im(ii);
  cnt1_im(ii)=cnt1_re(ii);
                              % 실수부 시프트 카운트가 더 작으면 허수부 시프트 카운트를 실수부로 맞춤 (더 보수적으로)
for ii=1:8
for jj=1:64
 % 결정된 블록별 시프트 카운트를 전체 512개 데이터에 대해 복사
  index1_re(64*(ii-1)+jj)=cnt1_re(ii);
  index1_im(64*(ii-1)+jj)=cnt1_im(ii);
```

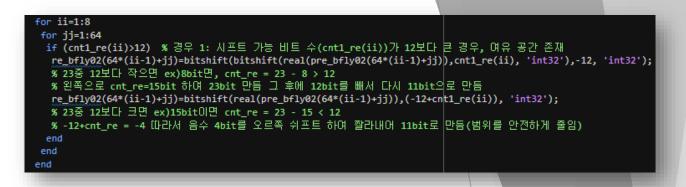


3. CBFP

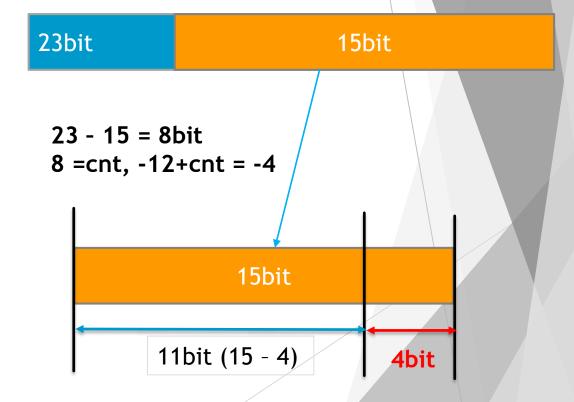
bit shift

Cnt1_re>12



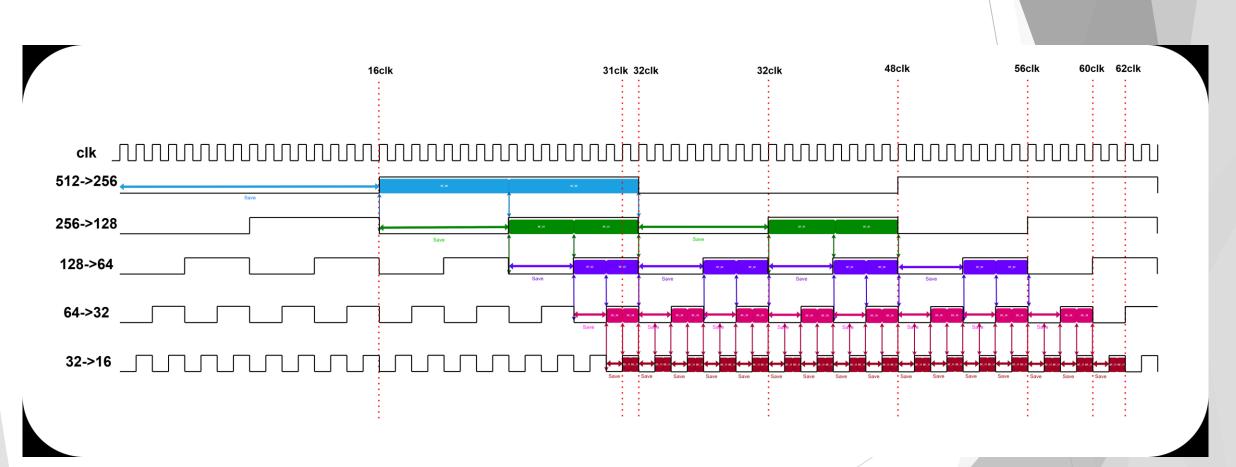


Cnt1_re<=12



3. Timing

timing graph



3. Timing

fac	c8 0	fac8_1	twf m0	fac8 0	fac8_1
49000			Ī		
(1), fft_in (17) , fft_in	(17), fft_in (25), fft	(25), fft_in (29)	mt_in (29) mt_in (31) m	(31), ffLin (32)	fft_in (32), fft_in (33), fft_in
(2), fft_in (18), fft_in	[0:15] [0	15] (26), (15) (30)	[0:15] [0:15] [0	:15] [0:15]	(0.15) (0.15) (0.15) (0.15) (1.17) (1
(3) fft_in (19) fft_in	[0:15]	15] [0:15] In (27) fft_in (3.1)	[0:15] [0:15] [0	:15] [0:15]	(0:15) (0:15) (0:15) (0:15) (1:5) (1
(4) fft_in (20) fft_in	[0:15] / / [0	15] [0:15] [32]	[0:15] [0	:15] t_in (34) fft_in (34)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(5), fft_in (21) [0:15]	(21) fft_in (29) fff	15] [0:15] in (29) fft_in (29)	[0:15] [0:15] [ft_in (33) [ft]	:15] t_in (35), fft_in (36)	[0:15] [0:15] [0:15] [0:15] fft_in (37), fft_in
(6), fft in (22) [0:15]	(22) fft_in (30) / fft	15] [0:15] [0:17] [17]	fft_in (34) fft_in (34) ff	t_in (36) fft_in (36)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(7), fft_in (23) (0.15) (0.15) (0.15) (0.15)	(23) fft_in $\sqrt{(3,1)}$ fft	15] [0:15] in (31), fft_in (31)	fft_in (35) fft_in (33) ff	t_in (37), fft_in (38)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(8), fft_in (24) fft_in (0:15)	(24), fft_in (32) , fft [0:15]	in (32), fft_in (32)	fft_in (36), fft_in (34) ff	t_in (38), fft_in (38)	fft_in (39), fft_in (40), fft_in (0:15)
(9), fft_in (25) fft_in [0:15]	(25), fft_in (25) fft	_in (33), fft_in (37)	fft_in (37), fft_in (37) fft_in (37)	Lin (39), fft_in (40)	fft_in (40), fft_in (41), fft_in (0:15)
(10) fft_in (26) fft_in (0:15)	(26), fft_in (26) fft	in (34), fft_in (38) 15] [0:15]	fft_in (38), fft_in (38) ff	t_in (40), fft_in (40)	fft_in (41), fft_in (42), fft_in (0:15)
(11) fft_in (27) fft_in (27) (27) (27) (27) (27) (27) (27) (27)	ro:151 //// ro	in (35), fft_in (39)	(0.15)	:15] [0:15]	fft_in (42), fft_in (43), fft_in [0:15]
(12) fft_in (28) fft_in (0:15) (13) fft_in (29) fft_in	[0:15] / /	in (36), fft_in (40) 15] [0:15] in (37), fft_in (37)	[0:15] [0:15] [0	:15] [0:15]	fft_in (43), fft_in (44), fft_in [0:15] [0:15] [0:15] fft_in fft_in (44), fft_in (45), fft_in
(13) IT_IN (29) IT_IN (10:15) (14) IT_IN (30) IT_IN (10:15)	[0:15] / /	15] (38), (15) (38)	[0:15] [0:15] [0	:15] [0:15]	[0:15] [0:15] [0:15] [0:15] [1:1] [1
[0:15] [0:15] [0:15] (15) fft_in (3,1) fft_in	[0:15] // [0	15] [0:15] In (39) fft_in (39)	[0:15] [0:15]	:15] [0:15]	(0:15) (0:15) (0:15) (0:15) (1:5) (1
[0:15] [0:15] [0:15] [0:15] [0:15]	[0:15]	15] [0:15] (40) fft_in (40)	[0:15] [0:15] [0:15] [ft_in (42) ft	:15] t_in (46), fft_in (46)	[0:15] [0:15] [0:15] [0:15] [0:15] [1
(17) fft_in (17) fft_in [0:15]	(33) [0:15] [0 (41) [fft]	15] [0:15] in (41) fft_in (45)	[0:15] [0:15] [ft_in (45) ft	:15] t_in (47) fft_in (48)	[0:15] [0:15] [0:15] [0:15] [fft_in (49) fft_in
(18) fft in (18) fft in	(34) fft_in (42) / fft	15] [0:15] (46)	fft_in (46) fft_in (46) ff	t_in (48), fft_in (48)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:16]
(19) fft_in (19) fft_in (19) fft_in (19)	(35) fft_in (43) / fft	15] [0:15] [15] [0:15] [15] [15] [15] [15] [15] [15]	fft_in (47), fft_in (45) f	t_in (49) fft_in (50)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(20) fft_in (20) fft_in (0:15)	(36), fft_in \(44)\/, fft	(44), fft_in (48)	fft_in (48) fft_in (46) ff	t_in (50), fft_in (50)	(51), (51), (52), (6.15) (6.15) (6.15)
(21) fft_in (21) fft_in (0:15)	(37), fft_in (45) fft	_in (45), fft_in (45)	fft_in (49) fft_in (49) fft	t_in (51) fft_in (52)	fft_in (52), fft_in (53), fft_in (0:15)
(22) fft_in (22) fft_in (0:15)	[0:15]	in (46), fft in (46)	fft_in (50), fft_in (50) ff	t_in (52), fft_in (52)	fft_in (53), fft_in (54), fft_in (0:15)
(23) fft_in (25) fft_in (25) fft_in (25)	TO:151 ///// TO	in (47), fft_in (47)	(0.15)	:15] [0:15]	fft_in (54), fft_in (55), fft_in (0:15) (0:15)
(24) fft_in (24) fft_in (0:15) (25) fft_in (25) fft_in (25)	[0:15]	in (48), fft_in (48) 15] [0:15] in (49), fft_in (53)	[0:15] [0:15] [0	:15] [0:15]	fft_in (55), fft_in (56), fft_in (56), fft_in (515) (515) (515) (515)
(25) ff_in (25) ff_in (25) (26) ff_in (26) ff_in (26) ff_in (26)	[0:15]	in (49), ff_in (53) 15] [0:15] in (50), ff_in (54)	[0:15] [0:15] [0	:15] [0:15]	fft_in (56), fft_in (57), fft_in (515) [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
[0:15] [0:15] [0:15] [0:15] [0:17] [0:18]	[0:15] / / / / / [0	15] (51), (51), (55)	[0:15]	:15] [0:15]	[0:15] [0:15] [0:15] [0:15] [fft_in (58), fft_in (59), fft_in
(28) fft_in (28) (28) (0:15)	(44) fft_in (44) fft	15] [0:15] _in (52), fft_in (56)	[0:15] [0:15] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1	:15] t_in (58) fft_in (58)	[0:15] [0:15] [0:15] fft_in (60) fft_in
[0:15] [0:15] (29) fft_in (29) fft_in	(45) fft_in (45) fff	15] [0:15]in (53) fft_in (53)	[0:15] [0:15] [0:17] [17] [18] [18] [18] [18] [18] [18] [18] [18	:15] t_in (59) fft_in (60)	[0:15] [0:15] [0:15] fft_in (61) fft_in
(30) fft in (30) fft in	(46) fft_in //(46) fft	15] [0:15] In (54) fft_in (54)	fft_in (58) fft_in (58) ff	:15] t_in (60) fft_in (60)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(31) (31) (0.15) (0.15) (0.15) (0.15)	(47)_ fft_in // (47) \ fft	15] [0:15] [0:15] [1] [0:15] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1	fft_in (59) fft_in (57) ff	t_in (61) fft_in (62)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
(32) (0:15) (0:15) (0:15) (15) (15) (15) (15) (15) (15) (15) ((48) fft_in (48) fft	in (56), fft_in (56)	fft_in (60) fft_in (58) ff	t_in (62) fft_in (62)	[0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15] [0:15]
			[55]		

3. FFT architecture

module 0

