FFTをRX621とRX220で試した
N=4x(2^p)x(3^q)x(5^r)x(7^s)

目的:

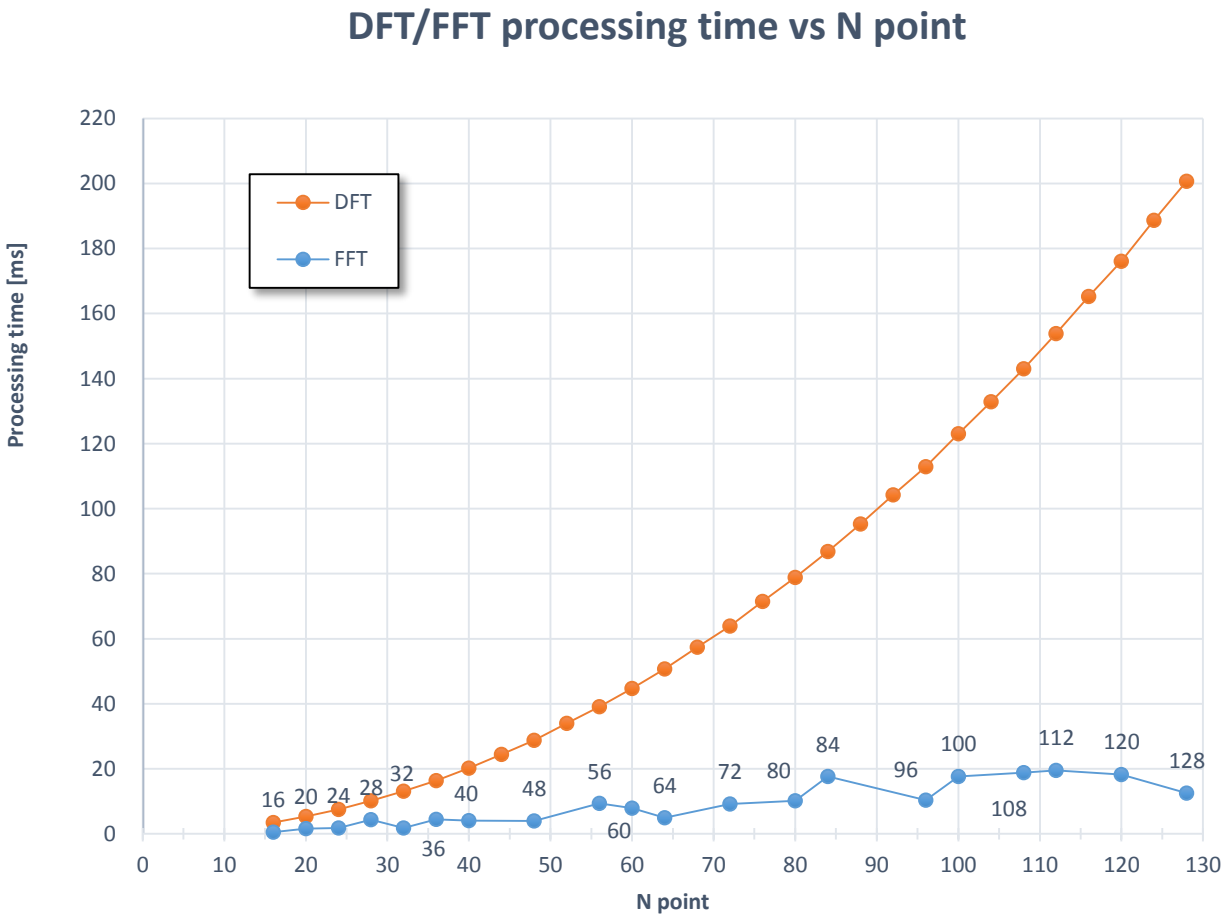
- RXマイコンでのFFT処理のパフォーマンスを把握する。
- 2のべき乗以外のNにおけるFFTを実装してみる。
- 複素数入力でない実数専用のFFTの実装はどうなるのか？をみる。
  →バタフライ演算は片肺になる
- FFTの本質である再帰性をそのまま再帰関数として実装する。
- 以上によりFFTをよく理解する。

Specification

- Boards
  - AKIZUKI AE-RX220
    - RAM: 16kByte
    - Clock: 20MHz (Xtal 20MHz)
    - FPUなし
  - AKIZUKI RX621
    - RAM: 96kByte
    - Clock: 96MHz (Xtal 12MHz x 8)
    - FPU(単精度)あり
- DFT, FFT
  - Input: real only
  - Output: 0 to Nyquist frequency
  - Precision
    - double 8byte
    - float 4byte

DFT/FFT on AKI-RX220(20MHz, 16kB)

N=4x(2^p)x(3^q)x(5^r)x(7^s)に対応した。



N	Radix
128	8x16
120	8x3x5
112	7x16
108	4x3x3x3
100	4x5x5
96	2x3x16
84	4x7x3
80	2x8x5
72	3x3x8
64	4x16
60	4x3x5
56	7x8
48	3x16
40	8x5
36	4x3x3
32	2x16
28	7x4
24	3x8
20	4x5
16	16

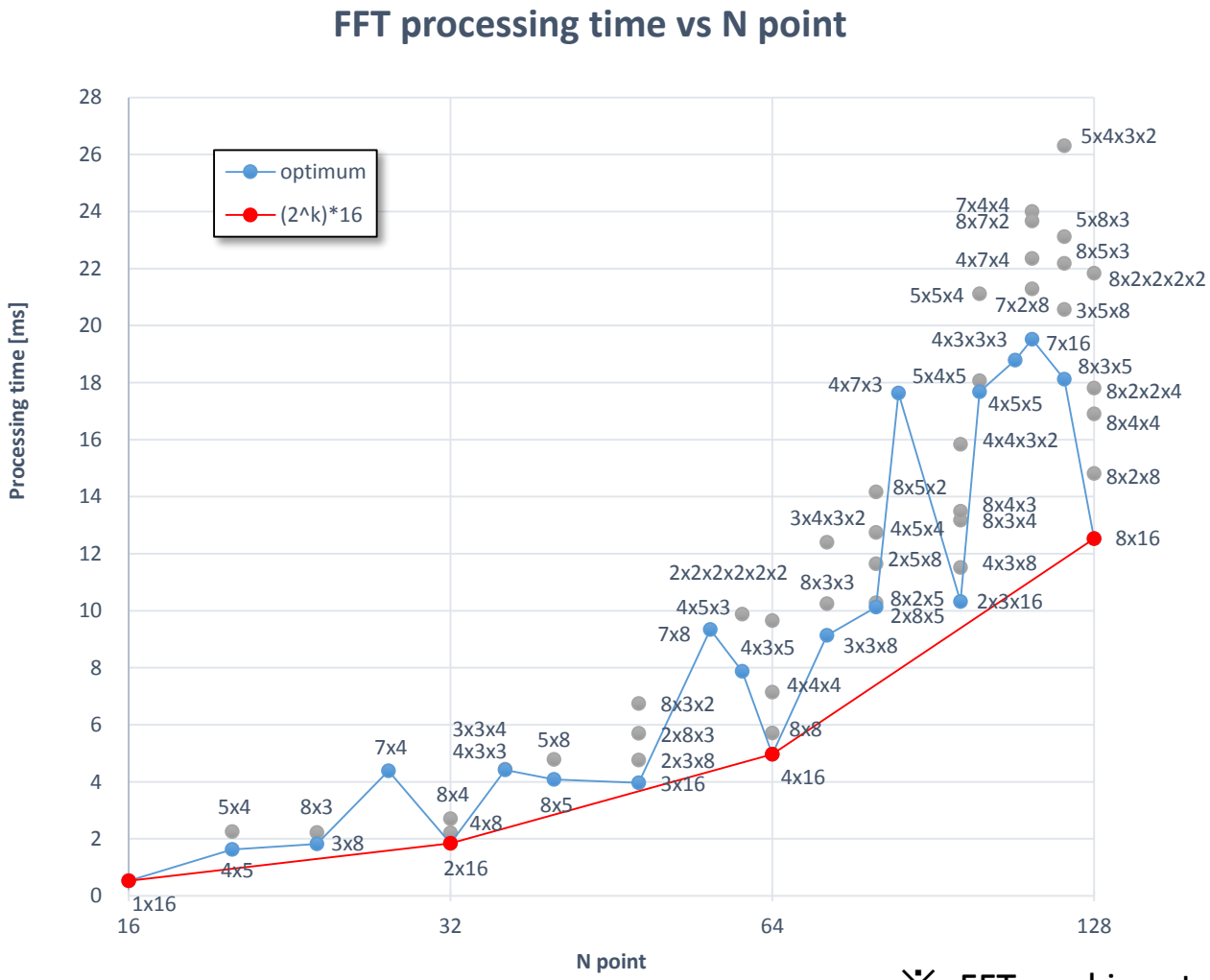
※: FFT: real input only

Note: cos/sin table をメモリ節約するため1/4周期ずらして兼用しているので、Nは4の倍数であることが必要。

Radix optimization

- AKI-RX220(20MHz, 16kB)

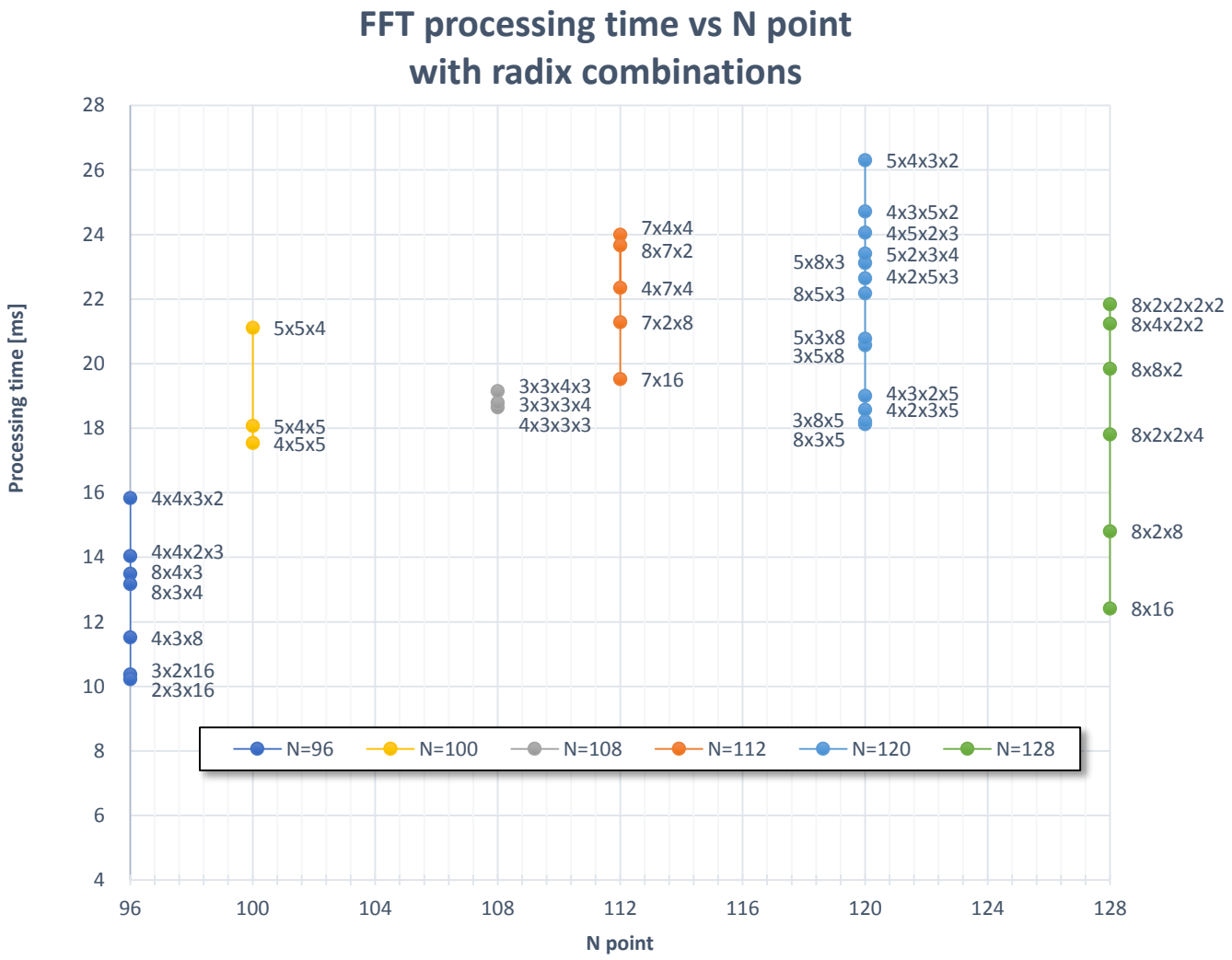
・中間ステージと終端ステージのRadixの組み合わせにより速度が変わる。
・N=(2^k)x16 は特に良い速度。バタフライ演算での計算簡略化が働くため。および終端ステージRadix=16を作成してあるため。



※: FFT: real input only

Radix optimization for N=96 to128

- AKI-RX220(20MHz, 16kB)



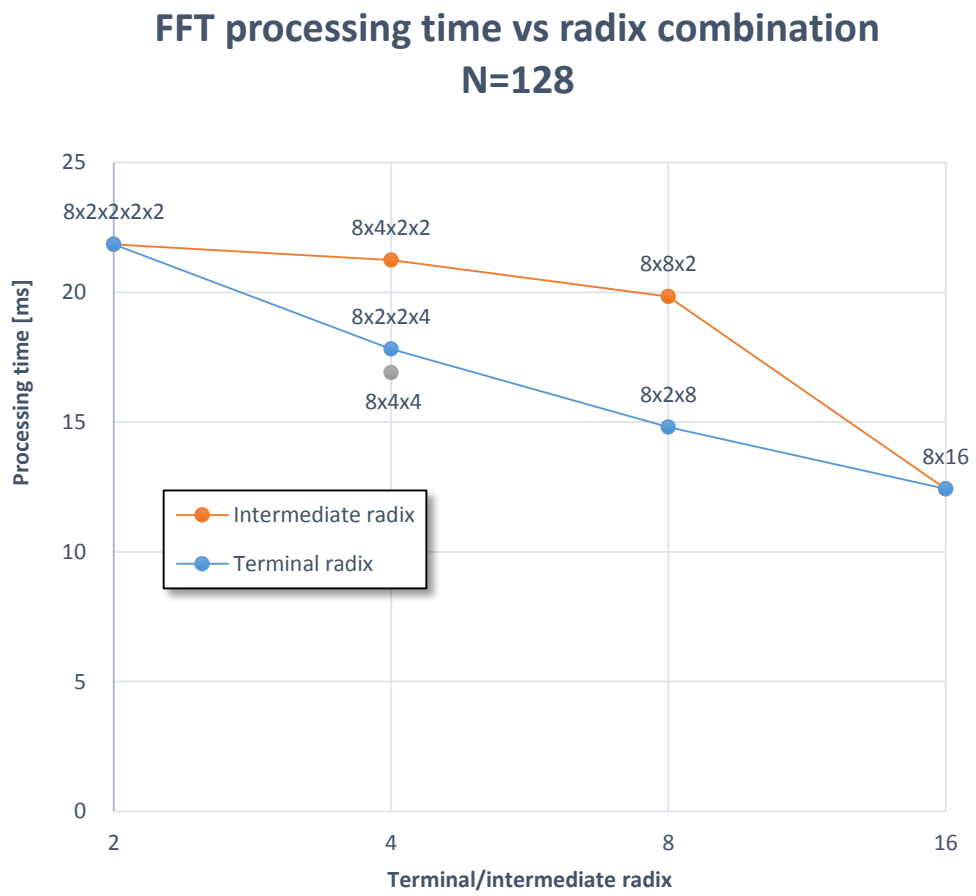
Kazuyuki Hirooka, 2016/5/14

※: FFT: real input only

Radix optimization: terminal vs intermediate

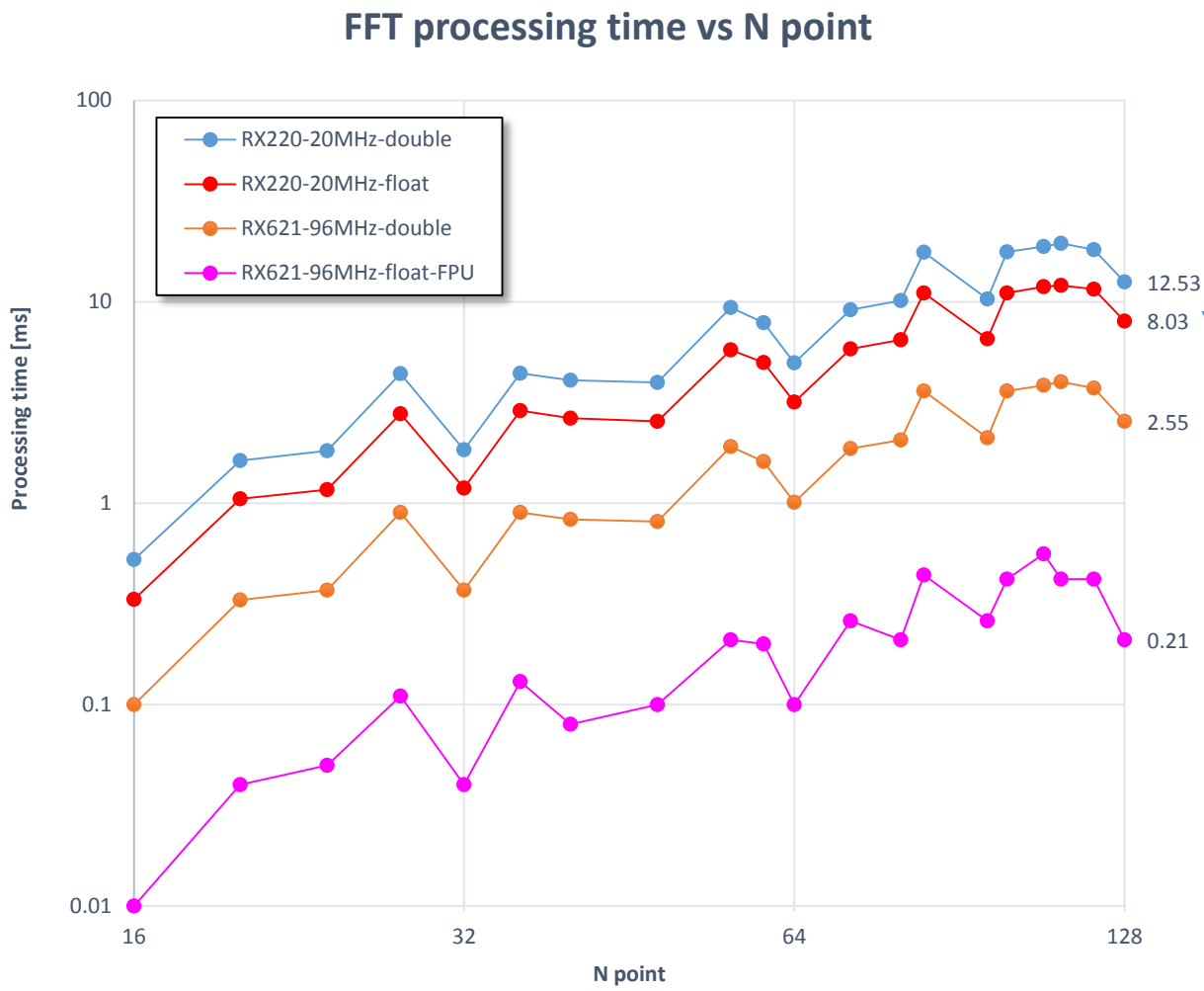
- AKI-RX220(20MHz, 16kB)

Terminal radix is more important than intermediate for improving processing time in this case.



※: FFT: real input only

FFT processing time comparison:
- clock frequency
- double, float and FPU



double->float(FPUなし）の速度向上、1.6倍。
ただし、精度劣化と引き換え。

クロック周波数比96MHz/20MHz=4.8にほぼ一致して速度が向上

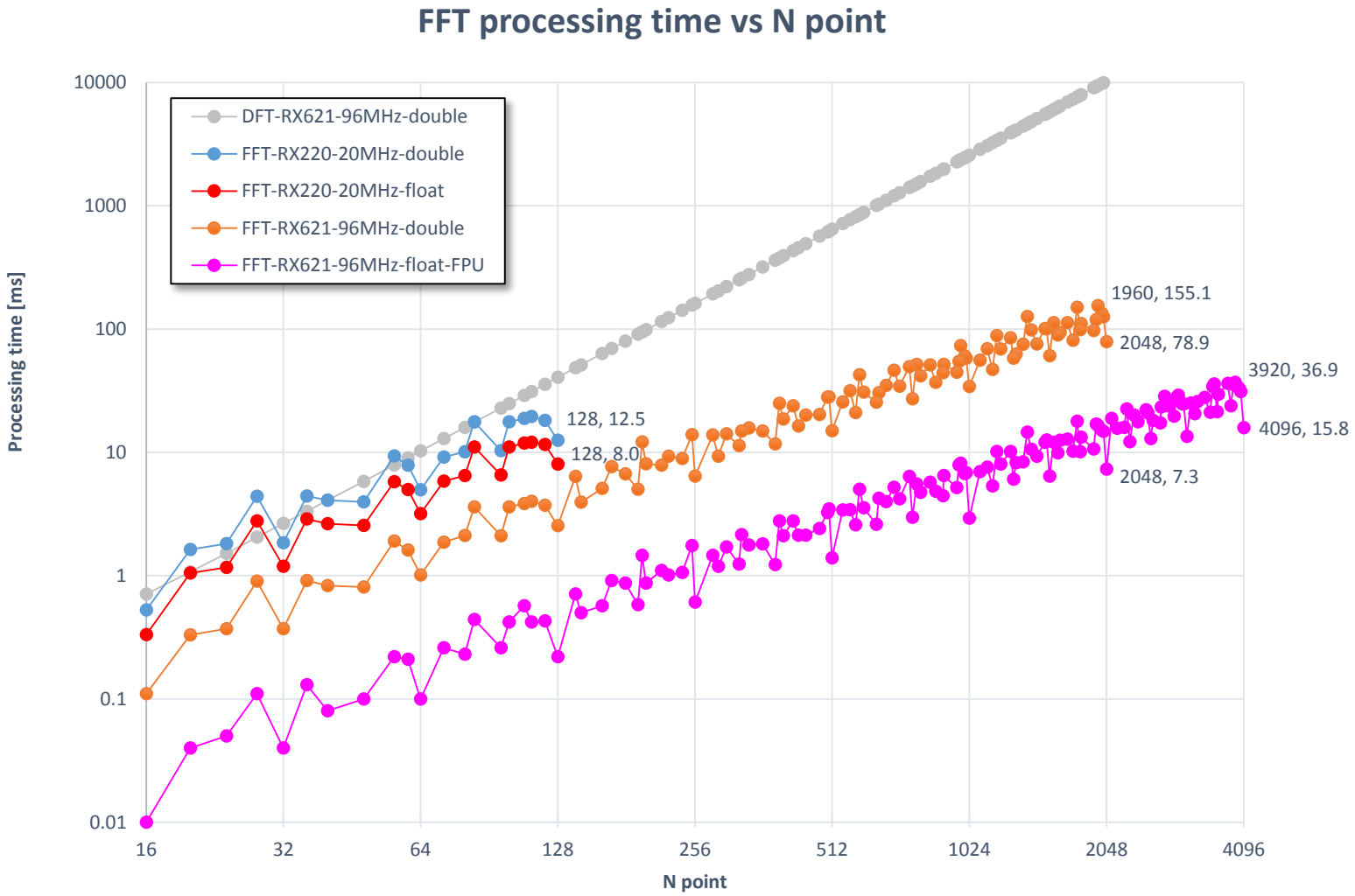
RX621-96MHzにおけるdouble->float(FPU) の効果、12倍。
(float演算割合の補正なし）

FPUなしのfloat の効果1.6倍を割り引き、さらにfloat演算割合を70%と仮定すると、FPUそれ自体の効果は(12/1.6)/0.7=11倍と計算される。

※: FFT: real input only

FFT performance of RX621, RX220

- RX621: 96kB, 96MHz, float-FPU
- RX220: 16kB, 12MHz, FPU none



※: FFT: real input only

Radix combination used in RX621

N	Radix	N	Radix	N	Radix	N	Radix
4096	8x8x4x16	2160	5x3x3x3x16	972	4x3x3x3x3x3	288	2x3x3x16
4032	4x7x3x3x16	2100	4x7x5x3x5	960	4x5x3x16	280	8x7x5
4000	5x5x5x2x16	2048	8x8x2x16	900	4x5x3x3x5	256	8x2x16
3920	7x7x5x16	2016	7x2x3x3x16	896	8x7x16	252	4x7x3x3
3888	4x4x3x3x3x3x3	2000	5x5x5x16	864	2x3x3x3x16	240	5x3x16
3840	8x5x2x3x16	1960	8x7x7x5	840	8x7x3x5	224	7x2x16
3780	4x7x3x3x3x5	1944	4x2x3x3x3x3x3	800	5x5x2x16	216	3x3x3x8
3600	5x5x3x3x16	1920	8x5x3x16	784	7x7x16	200	8x5x5
3584	8x4x7x16	1800	8x5x3x3x5	768	8x2x3x16	196	7x7x4
3528	7x7x3x3x8	1792	8x7x2x16	756	4x7x3x3x3	192	4x3x16
3500	4x7x5x5x5	1764	4x7x7x3x3	720	5x3x3x16	180	4x3x3x5
3456	8x3x3x3x16	1728	4x3x3x3x16	700	4x7x5x5	168	7x3x8
3360	7x5x2x3x16	1680	7x5x3x16	672	7x2x3x16	160	5x2x16
3240	8x3x3x3x3x5	1620	4x3x3x3x3x5	648	3x3x3x3x8	144	3x3x16
3200	8x5x5x16	1600	4x5x5x16	640	8x5x16	140	4x7x5
3136	4x7x7x16	1568	7x7x2x16	600	8x5x3x5	128	8x16
3072	8x8x3x16	1536	8x4x3x16	588	4x7x7x3	120	8x3x5
3024	7x3x3x3x16	1512	7x3x3x3x8	576	4x3x3x16	112	7x16
3000	8x5x5x3x5	1500	4x5x5x3x5	560	7x5x16	108	4x3x3x3
2940	4x7x7x3x5	1440	5x2x3x3x16	540	4x3x3x3x5	100	4x5x5
2916	4x3x3x3x3x3x3	1400	8x7x5x5	512	8x4x16	96	2x3x16
2880	4x5x3x3x16	1372	7x7x7x4	504	7x3x3x8	84	4x7x3
2800	7x5x5x16	1344	4x7x3x16	500	4x5x5x5	80	5x16
2744	7x7x7x8	1296	3x3x3x3x16	480	5x2x3x16	72	3x3x8
2700	4x5x3x3x3x5	1280	8x5x2x16	448	4x7x16	64	4x16
2688	8x7x3x16	1260	4x7x3x3x5	432	3x3x3x16	60	4x3x5
2592	2x3x3x3x3x16	1200	5x5x3x16	420	4x7x3x5	56	7x8
2560	8x4x5x16	1176	7x7x3x8	400	5x5x16	48	3x16
2520	8x7x3x3x5	1152	8x3x3x16	392	7x7x8	40	8x5
2500	4x5x5x5x5	1120	7x5x2x16	384	8x3x16	36	4x3x3
2400	5x5x2x3x16	1080	8x3x3x3x5	360	8x3x3x5	32	2x16
2352	7x7x3x16	1024	8x8x16	336	7x3x16	28	7x4
2304	8x2x3x3x16	1008	7x3x3x16	324	4x3x3x3x3	24	3x8
2268	4x7x3x3x3x3	1000	8x5x5x5	320	4x5x16	20	4x5
2240	4x7x5x16	980	4x7x7x5	300	4x5x3x5	16	16

END