B.Tech. Project Report

Hardware Implementation of Random Access Preamble Detection for Narrowband IoT (NB-IoT) Systems

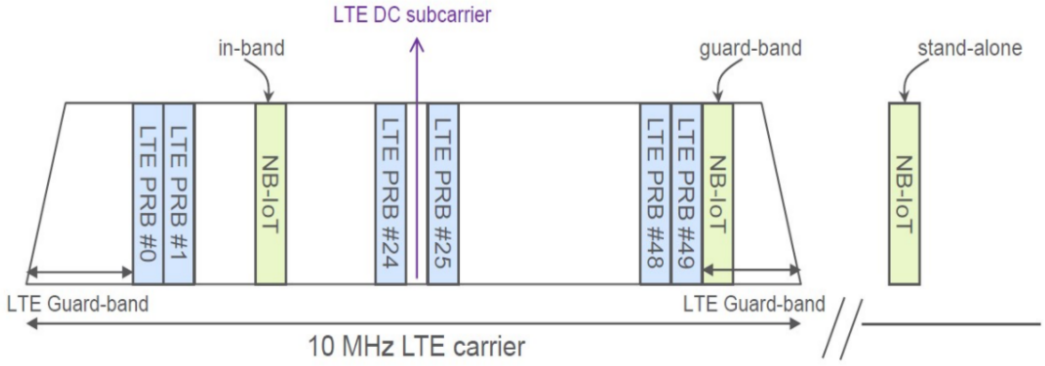
Rishabh Tripathi (180030036)

Introduction

Narrowband Internet of Things (NB-IoT) is a low-power wide-area (LPWA) technology developed by 3GPP to operate on top of existing LTE systems. It has been introduced with the goal to connect massive low-cost, low-complexity and long-life IoT devices with extended coverage. NB-IoT devices are designed to have more than 10 years of battery life. To ensure a long battery life of NB-IoT devices, 3GPP has introduced Power Saving Mode (PSM) and enhanced Discontinuous Reception (eDRX) modes. In the NB-IoT communication model, users send only low, infrequent, and delay-tolerant data. So, a massive number of devices can be catered simultaneously by a single cell. NB-IoT reuses the LTE design extensively, including the numerologies, downlink orthogonal frequency-division multiple-access (OFDMA), uplink single-carrier frequency-division multiple-access (SC-FDMA), channel coding, rate matching, interleaving, etc. This reduces the time required to develop full specifications.

Specifications

NB-IoT has a channel bandwidth of 200 kHz but occupies only 180 kHz. This is equal to one resource block in LTE (1 RB). The following operation modes are possible: Stand alone operation, Guard band operation, In-band operation.



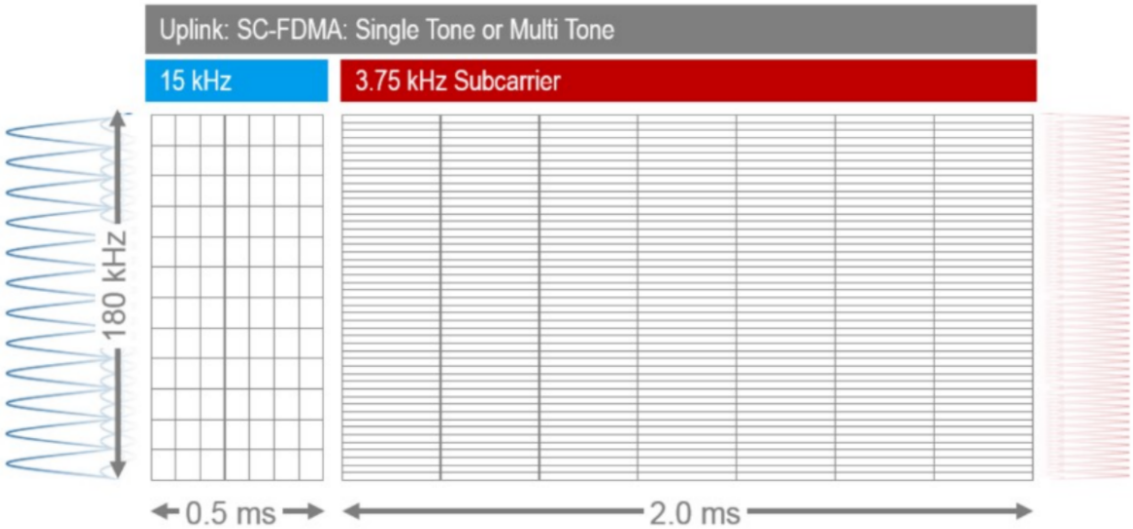
The downlink uses OFDMA with a carrier spacing of 15 kHz. NB-IoT uses only 12 carriers, which leads to an occupied bandwidth of 180 kHz. One slot consists of seven OFDMA symbols.

NB-IoT defines only QPSK modulation in the downlink. Each of these resource elements carries a complex value with values according to the modulation scheme.

In the uplink (UL), two different possibilities are defined. It can use either a single carrier or multiple carriers.

- Single-tone: each user only uses one subcarrier at a time, with 15 kHz or 3.75 kHz carrier spacing.
- Multitone: multiple subcarriers are simultaneously used by the same user, SC-FDMA with 15 kHz carrier spacing.

There are 7 SC-FDMA symbols within a slot. With a carrier spacing of 15 kHz, 12 carriers are available and for 3.75 kHz spacing 48 carriers are available.The symbol duration for 3.75 kHz subcarrier spacing has four times the duration compared to 15 kHz, which results in a slot length of 2 ms.



NPRACH

Random Access (RA) is the first connection between the UE and the base station and ensures that the UE can be identified and synchronized with the BS. The Random Access Channel (NPRACH) uses a single tone with frequency hopping. The preamble is based on symbol groups on a single subcarrier. Preamble transmission comprises 4 symbol groups transmitted without gaps. While each group uses a single subcarrier, it can hop across 12 subcarriers and each symbol group has a cyclic prefix (CP) followed by 5 symbols. The subcarrier spacing used for the transmission of NPRACH is equal to 3.75 kHz. This leads to 48 possible subcarriers that can be used to transmit a preamble over the 180 kHz bandwidth of the NB-IoT system. Two preamble formats are defined, format 0 and format 1, which differ in their CP length. The CP

length is 66.67s (format 0) for cell radius up to 10 km and 266.7s (format 1) for cell radius up to 40 km. In order to reduce the probability of collision between the preambles transmitted from different UEs in a cell, the mechanism of frequency hopping is used.

The choice of the subcarrier for preamble transmission is random and independent of the choice made by other devices. If more than one device from the same CE level selects the same initial subcarrier in the same PRACH occasion, it results in a collision. In order to reduce the probability of collision between the preambles transmitted from different UEs in a cell, the mechanism of frequency hopping is used. These frequencies f_i (in kHz) are pseudo-randomly chosen according to the formula

$$f_i = (n_{sc}^{RA}(i) + Kk_0 + \frac{1}{2}) \times 3.75,$$
$$n_{sc}^{RA}(i) \in \{0, 1, .., 47\}$$

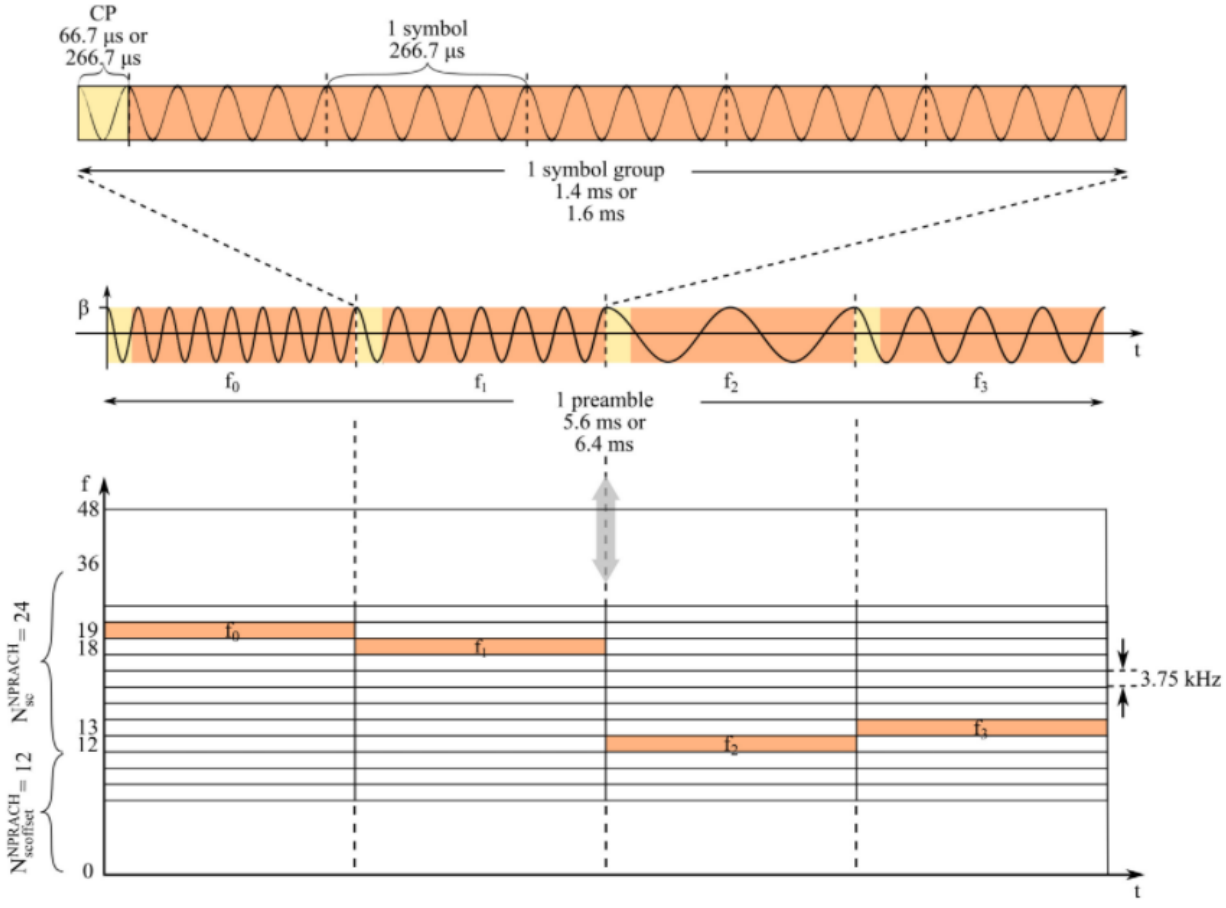


Fig. 14. Time and frequency representation of one NPRACH preamble where $N_{scoffset}^{NPRACH} = 12$, $N_{sc}^{NPRACH} = 24$, and $n_{init} = 7$.

NPRACH Detector

The baseband equivalent digital signal for the random access preamble transmission can be written as:

$$s[n;m] = \frac{\sqrt{E}}{N} \sum_{k} S[k;m] e^{j2\pi \frac{k}{N} n}, \quad n = -N_{cp}, \ldots, N-1$$

where s[n; m] is n − th sample of m − th symbol group and S[k; m] denotes symbol on k − th subcarrier during m − th symbol group.

For the narrow random access hopping range that is not larger than 45 kHz(12 subcarriers) the channel can be sufficiently modeled using single channel coefficient h[n; m] at the n − th sample of the m − th symbol group

$$h[n;m] = a[m]\delta[n],$$

The spectrum of the received signal is given as follows,

$$\begin{aligned}\tilde{y}[i;m] =& B(\Delta f, D) a[m] u[m] e^{j2\pi\Delta f(m(N_{cp}+\xi N)+iN))} \\ & \times e^{-j2\pi \frac{\Omega(m)}{N} D} + \tilde{v}[i;m].\end{aligned}$$

$$B(\Delta f, D) = \sqrt{E} \frac{sin(N\pi\Delta f)}{N sin(\pi\Delta f)} e^{j2\pi\Delta f(\frac{N-1}{2} - D)}$$

PREAMBLE FORMAT 0: $N_{cp} = N/4$

PREAMBLE FORMAT 1: $N_{cp} = N$

where D is time of arrival (ToA) to be estimated and $D \in [0, N_{CP} - 1]$, Δf is the residual carrier frequency offset (CFO) and $\tilde{v}[i;m]$ is a complex AWGN with zero mean and variance N_0.

The problem at the base station is to detect the presence of the active UEs and for each detected UE, estimate its ToA and CFO parameters accurately and efficiently. we make the simplifying approximation of a block fading model: the channel does not change in a block of Q symbol groups but change independently over the blocks. Here Q is chosen such that L/Q is an integer. Now, ToA and residual CFO can be jointly estimated as,

$$\begin{aligned}(D^*, \Delta f^*) &= \arg\max_{D,\Delta f} J(D, \Delta f) \\ &= \arg\max_{D,\Delta f} \sum_{g=0}^{L/Q-1} |J_g(D, \Delta f)|^2\end{aligned}$$

$$J_g(D, \Delta f) = \sum_{m=gQ}^{(g+1)Q-1} \sum_{i=0}^{\eta-1} \tilde{y}[i;m] u^*[m] e^{-j2\pi\Delta f(m(N_c p+\eta N)+iN)} e^{j2\pi \frac{\Omega(m)}{N} D}.$$

We can express the above equation in the form of 2D DTFT,

$$W_g[p,q] = \sum_{n=0}^{M_1-1} \sum_{k=0}^{M_2-1} w_g[n,k] e^{-j2\pi \frac{n}{M_1} p} e^{-j2\pi \frac{k}{M_2} q}$$

$$w_g[n,k] = \begin{cases} z[i;m] & if n = (m - gQ)(\eta + 1) + i, k = \Omega(m); \\ 0 & \text{otherwise.} \end{cases}$$

$$z[i;m] = \tilde{y}[i;m] u^*[m].$$

Now, the point of maximum correlation is found as follows and ToA and CFO are estimated,

$$(p^*, q^*) = \arg\max_{p,q} \tilde{J}(p,q)$$

$$where, \quad \tilde{J}[p,q] = \sum_{g=0}^{L/Q-1} |W_g[p,q]|^2$$

$$\Delta f^* = \begin{cases} \frac{1}{NM_1} p^* & if p^* < \frac{M_1}{2}; \\ \frac{1}{NM_1}(p^* - M_1) & \text{otherwise.} \end{cases} \quad D^* = \begin{cases} -\frac{N}{M_2} q^* & if p^* < \frac{M_2}{2}; \\ -\frac{N}{M_2}(q^* - M_2) & \text{otherwise.} \end{cases}$$

The above algorithm to find ToA and CFO requires the computation of 2D DTFT using the 2D FFT algorithm. For implementing the NPRACH detector on a DSP Board, we need to enable a hardware accelerator for FFT computation. Since, FFT computation is the most costly step in the entire process, the project focussed on enabling and efficient usage of a FFT hardware accelerator.

DSP Board

The board used for implementation purposes is the TMS320C5515 digital signal processor (DSP). It contains a high-performance, low-power DSP to efficiently handle tasks required by portable audio, wireless audio devices, industrial controls, software defined radio, fingerprint biometrics, and medical applications. The DSP consists of the following primary components:

- A C55x CPU and associated memory
- FFT hardware accelerator
- Four DMA controllers and external memory interface
- Power management module
- A set of I/O peripherals that includes I2S, I2C, SPI, UART, Timers, EMIF, 10-bit SAR ADC, LCD Controller, USB 2.0

FFT Hardware Accelerator (HWAFFT)

The C55x CPU includes a tightly coupled FFT accelerator that communicates with the C55x CPU through the use of the coprocessor instructions. The main features of this hardware accelerator are:

- Supports 8- to 1024-point (powers of 2) complex-valued FFTs.
- Internal twiddle factor generation for optimal use of memory bandwidth and more efficient programming.
- Basic and software-driven auto-scaling feature provides good precision versus cycle count trade-off.
- Single-stage and double-stage modes enable computation of one or two stages in one pass, and thus better handle the odd power of two FFT widths.
- Is 4 to 6 times more energy efficient and 2.2 to 3.8 times faster than the FFT computations on the CPU.

The software interface to the HWAFFT is handled through a set of coprocessor instructions that, when decoded by the coprocessor, perform initialization, load/store, and execution operations on the HWAFFT coprocessor. C-callable functions are provided that contain the necessary sequences of coprocessor instructions for performing FFT/ IFFT computations in the range of 8 to 1024 points. Additionally, an optimized out-of-place bit-reversal function is provided to perform the complex vector bitreversal required by Radix-2 FFT computations. To conserve on-chip RAM these functions have been placed in the on-chip ROM of the DSP.

The input and output vectors of the HWAFFT contain complex numbers. Each real and imaginary part is represented by a two's complement, 16-bit fixed-point number. The most significant bit holds the number's sign value, and the remaining 15 are fraction bits (S16Q15 format). The range of each number is $[-1, 1-(1/2)^{15}]$. Real and imaginary parts appear in an interleaved order within each vector:

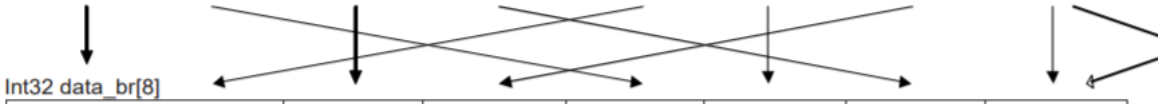
Int32 CMPLX_Vec32[N] = …(N = FFT Length)

Real[0]	Imag[0]	Real[1]	Imag[1]	Real[2]	Imag[2]
Bit31,....., Bit16, Bit15,....., Bit0		Bit31,....., Bit16, Bit15,. ....., Bit0		Bit31,....., Bit16, Bit15,....., Bit0	

Before computing the FFT/IFFT on the HWAFFT, the input buffer must be bit-reversed to facilitate a Radix-2 DIT computation. This function contains optimized assembly that executes on the CPU to rearrange the Int32 elements of the input vector by placing each element in the destination vector at the index that corresponds to the bit-reversal of its current index.

Int32 data[8]

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Index = 000	001	010	011	100	101	110	111



Int32 data_br[8]

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Index = 000	100	010	110	001	101	011	111

Challenges and Solutions

- The memory location of the arrays passed into the HWAFFT functions should not be in the heap memory, it should be within the DARAM. For this purpose a buffer array has to be maintained in the DARAM for each of the input parameters of hwafft_br() and all hwafft() functions.
- Strict requirements are placed on the address of the bit-reversal destination buffer. This buffer must be aligned in RAM such that log2(4 * N) zeros appear in the least significant bits of the byte address (8 bits), where N is the FFT Length.
  Code snippet used:

```
#define ALIGNMENT 2*N
// ALIGNS data_br_buf to an address with log2(4*N) zeros in the
// least significant bits of the byte address
#pragma DATA_SECTION(data_br _buf, "data_br_buf");
// Allocation to Section: "data_br _buf : > DARAM" in Linker CMD File
#pragma DATA_ALIGN (data_br_buf, ALIGNMENT);
Int32 data_br_buf[N = FFT Length];
Int32 * data_br = data_br_buf;
```

- The HWAFFT output does not remain active after the boot process. By the time the bootloader releases control to the user code, all peripheral clocks will be off and all domains in the ICR (Idle Configuration Register), except the CPU domain, will be idled. For this purpose, to enable the HWAFFT the ICR must be reconfigured at the beginning of the user program. Code snippet used is as follows:

```
*(ioport volatile unsigned *)0x0001 = 0x000E;
asm(" idle"); // must add at least one blank before idle in " ".
```

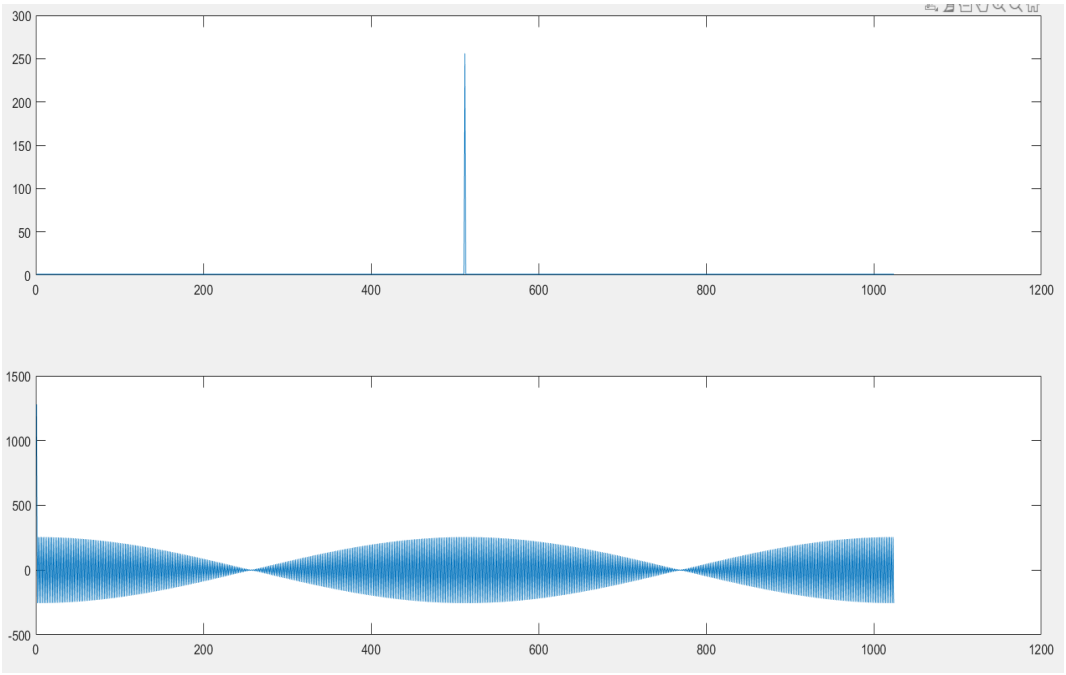
  The ICR has the following field description:

Bit	Field	Value	Description
15-10	Reserved	0	Reserved.
9	HWAI		FFT hardware accelerator idle control bit.
		0	Hardware accelerator remains active after execution of an IDLE instruction.
		1	Hardware accelerator is disabled after execution of an IDLE instruction.
8	IPORTI		Instruction port idle control bit. The IPORT is used for all external memory instruction accesses.
		0	IPORT remains active after execution of an IDLE instruction.
		1	IPORT is disabled after execution of an IDLE instruction.
7	MPORTI		Memory port idle control bit. The memory port is used for all DMA, LCD DMA, and USB CDMA transactions into on-chip memory.
		0	MPORT remains active after execution of an IDLE instruction.
		1	MPORT is disabled after execution of an IDLE instruction.
6	XPORTI		I/O port idle control bit. The XPORT is used for all CPU I/O memory transactions.
		0	XPORT remains active after execution of an IDLE instruction.
		1	XPORT is disabled after execution of an IDLE instruction.
5	DPORTI		Data port idle control bit. The data port is used for all CPU external memory data accesses.
		0	DPORT remains active after execution of an IDLE instruction.
		1	DPORT is disabled after execution of an IDLE instruction.
4-1	IDLECFG	0111b	Idle configuration bits. You must always set bit 1, 2 and 3 to 1 and bit 4 to 0 before executing the idle instruction.
0	CPUI		CPU idle control bit.
		0	CPU remains active after execution of an IDLE instruction.
		1	CPU is disabled after execution of an IDLE instruction.

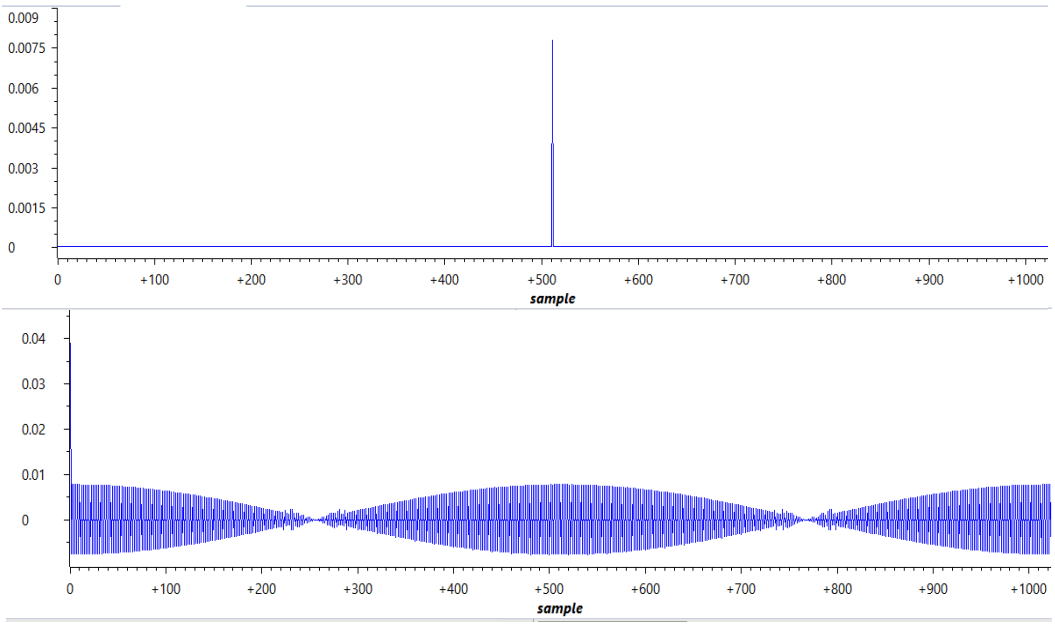
Results

The currently used DSP board does not have sufficient resources to support the entire NPRACH detection system. Therefore, for testing purposes, we generate all results using some dummy inputs and compare with corresponding MATLAB results.

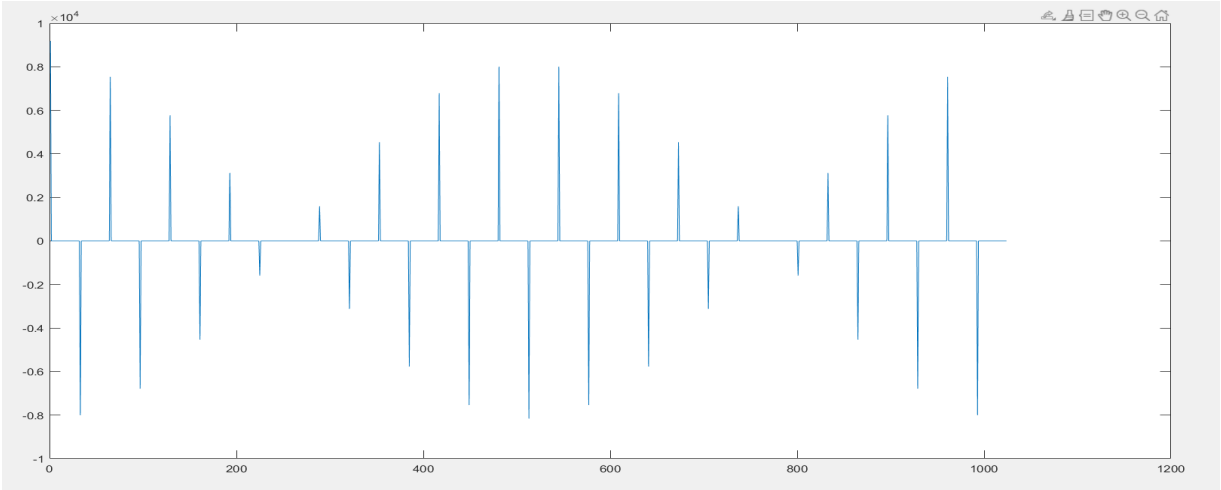
- We generate a delta function of 1024 length in MATLAB and in CCS to test and compare the results of the FFT built-in MATLAB and the HWAFFT co-processor.
  MALTAB input and FFT output:



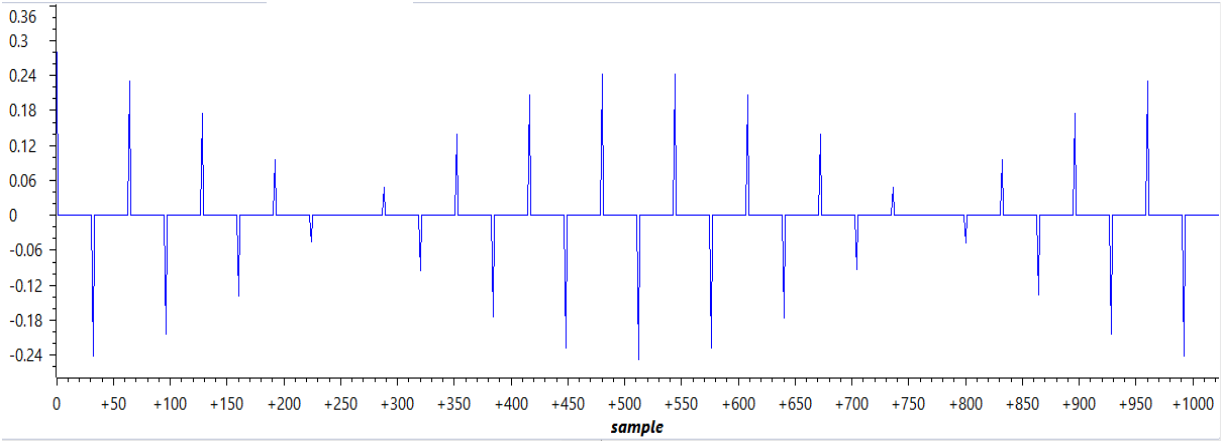
HWAFFT input and FFT output:



- We also compare 2D FFT results similarly. The input here is a 2D array where each row has a delta function exactly like above.
  MATLAB output:



HWAFFT output:



References

1. https://www.researchgate.net/publication/344237842_A_Tutorial_on_NB-IoT_Physical_Layer_Design
2. https://www.ti.com/lit/ug/sprufx5e/sprufx5e.pdf
3. https://www.ti.com/lit/an/sprabb6b/sprabb6b.pdf
4. https://www.ti.com/lit/er/sprz308d/sprz308d.pdf
5. J. Hwang, C. Li, and C. Ma. Efficient detection and synchronization of superimposed nb-iot nprach preambles. IEEE Internet of Things Journal, 6(1):1173–1182, 2019.