ソフトウェア無線を用いた 重畳符号化通信の基礎実装と評価

山崎 景太[1] 猿渡 俊介[1] 渡辺 尚[2]

概要：重畳符号化は，2 つの端末に対する異なる 2 つのフレームを 1 つのフレームに重畳して送信することができるため，無線通信のスループット向上が期待されている．重畳前の 2 つのフレームをそれぞれファーストレイヤ，セカンドレイヤと呼ぶ．本稿では，BPSK を用いた重畳符号化の実装と $(0, \pi/2)$-BPSK を用いた重畳符号化の実装と評価を示す．実装では，ハードウェアとしてソフトウェア無線機である Universal Software Radio Peripheral N200 (USRP N200)，ソフトウェアとして USRP Hardware Driver (UHD) と C++を用いた．実装した重畳符号化の動作検証の結果，$(0, \pi/2)$-BPSK を用いた重畳符号化ではファーストレイヤとセカンドレイヤに割り当てられた電力が近い場合でも信号が分離できることを示す．

Software Implimentation and Evaluation of Superposition Coding with UHD and USRP-N200

Keita Yamazaki[1] Shunsuke Saruwatari[1] Takashi Watanabe[2]

1. はじめに

2010 年の総務省情報通信審議会報告によると，2017 年には 2007 年に比べ通信データ量が約 200 倍になるだろうと述べられている [1]．その予測を裏付けるかのように，2012 年の総務省情報通信白書によると，スマートフォンの増加から移動通信トラフィック量が年間約 2.2 倍に増加していることが示されている [2]．今後も増加し続ける新しいデバイスやサービスに対応するために，有限の電波資源を高効率に利用する無線 LAN 技術が求められている．これに向けて，無線資源を効率的に利用するための新しい無線 LAN 方式の基礎技術として，MIMO [3–6]，レートレス符号 [7–9]，逐次干渉除去 [10,11]，無線全二重方式 [12–15]，重畳符号化 [16–23] などが研究されている．

無線資源の効率的な利用に向け，本研究では重畳符号化に着目する．重畳符号化は，1 つのアクセスポイントから 2 つの端末にデータを送る際に，2 つの宛先への信号を重ねあわせて同時に送信する技術である．逐次干渉除去 [10,11] を用いることで信号が分離できるため，1 度の通信で 2 つの宛先に同時にデータを届けることができる．

重畳符号化の実用化に向け，重畳符号化をソフトウェア無線技術を用いて実装した例も報告されている [16,17]．しかしながら，既存の研究での実装例では変調方式がそれぞれ異なる上に得られている結果も異なるにも関わらず，実装の詳細が明らかになっていない．

このような観点から，本稿では，重畳符号化の基礎的な部分の実装の詳細を示す．具体的には，ソフトウェア無線のハードウェアプラットフォームである USRP と，USRP 用のドライバである UHD を用いて，C++で記述したソフトウェアで重畳符号化を実装する．重畳符号化のファーストレイヤとセカンドレイヤに割り当てた電力が近い場合に信号の分離を失敗する例として BSPK 同士を重畳・分離するソフトウェアを示す．ファーストレイヤとセカンドレイヤに割り当てた電力が近い場合でも信号の分離が成功する例として通常の BPSK と位相を $\pi/2$ ずらした BPSK とを重畳・分離するソフトウェアを示す．実装したソフトウェアを動作検証した結果，$(0, \pi/2)$-BPSK を用いた重畳符号化ではファーストレイヤとセカンドレイヤに割り当てられ

[1] 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
[2] 大阪大学情報科学研究科
Graduate School of Information Science and Technology, Osaka University

た電力が近い場合でも信号が分離できることを示す．

本稿の構成は以下の通りである．2 節では，関連研究として重畳符号化とソフトウェア無線技術について述べると共に，重畳符号化の実装での課題について述べる．3 節では，本稿での実装に用いたハードウェアとソフトウェアについて述べる．4 節では，BPSK を用いた重畳符号化の実装について，送信機と受信機の構成を示す．5 節では，$(0, \pi/2)$-BPSK を用いた重畳符号化の実装について述べる．6 節では，実装した BPSK を用いた重畳符号化と $(0, \pi/2)$-BPSK を用いた重畳符号化の動作検証を行う．最後に，7 節でまとめとする．

2. 関連研究

2.1 重畳符号化

重畳符号化は，アクセスポイントからフレームを送信する際に，2 つの異なる宛先を持つフレームを変調した信号に対して異なる電力を割り当て，重ね合わせて送信する技術である．各受信側では逐次干渉除去を用いてフレームを復調する．逐次干渉除去では，アクセスポイントが 2 つの端末から同時にフレームを受信した場合に，各端末とアクセスポイントの伝搬損失の差を利用することで各フレームを同時に復調する [11].

例として，図 1 に BPSK を用いた重畳符号化の信号ダイヤグラムを示す．重畳符号化では，2 つの信号に対して異なる電力が割り当てられる．一方には P_{1st} の電力が，もう一方には P_{2nd} の電力が割り当てられている．P_{1st} は P_{2nd} より大きい電力であるとする．重畳符号化では，高い電力が割り当てられた信号をファーストレイヤ，低い電力が割り当てられた信号をセカンドレイヤと呼ぶ．図 1 の左がファーストレイヤの信号ダイヤグラムであり，図 1 の中央がセカンドレイヤの信号ダイヤグラムである．ファーストレイヤとセカンドレイヤの信号を足し合わせることによって，図 1 右の重畳符号化の信号ダイヤグラムが生成される．

図 2 に重畳符号化の動作例を示す．図 2 では，アクセスポイント A が重畳符号化を用いて端末 B と端末 C に対して異なるフレームを同時に配送している．アクセスポイント A は，端末 B 宛てをファーストレイヤ，端末 C 宛てをセカンドレイヤとし，2 つの信号を重畳して 1 つの重畳フレームを作成して送信する．端末 B が重畳フレームを受け取ると，セカンドレイヤはノイズとして扱われるため，ファーストレイヤの端末 B 宛てのフレームのみが復調される．端末 C が重畳フレームを受け取ると，まず，ファーストレイヤである端末 B 宛てのフレームのみが復調される．端末 C は，受信した重畳フレームの信号から復調した端末 B のフレームの信号を取り除き，セカンドレイヤである端末 C 宛てのフレームを復調する．

N をノイズ，P_{1st} を端末 B 宛てフレームに割り当てる送信電力，P_{2nd} を端末 C 宛てのフレームに割り当てる送信電力，h_{AB} をアクセスポイント A と端末 B 間のチャネル，h_{AC} をアクセスポイント A と端末 C 間のチャネルとする．シャノン = ハートレーの定理 [24] により，アクセスポイントから端末 B への通信容量 C_{AB} は，セカンドレイヤの信号がチャネル h_{AB} を通ってノイズとなるため，次の式で表わされる．

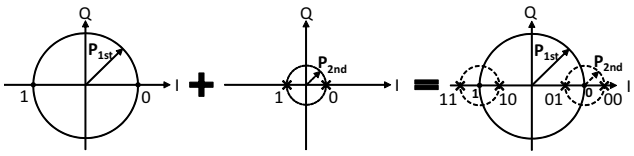

図 1 重畳符号化の信号ダイヤグラム

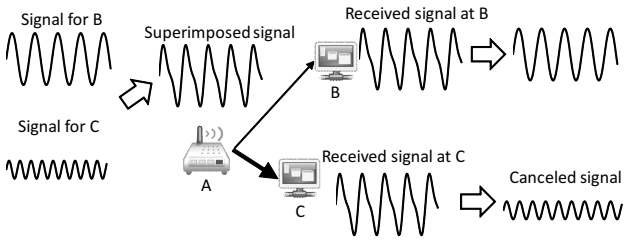

図 2 重畳符号化の動作例

$$C_{AB} = log\left(1 + \frac{P_{1st}|h_{AB}|^2}{P_{2nd}|h_{AB}|^2 + N}\right) \quad [bit/s/Hz] \quad (1)$$

また，アクセスポイントから端末 C への通信容量 C_{AC} は，ファーストレイヤの信号を完全に除去できたと仮定すると，次の式で表わされる．

$$C_{AC} = log\left(1 + \frac{P_{2nd}|h_{AC}|^2}{N}\right) \quad [bit/s/Hz] \quad (2)$$

このような重畳符号化の同時通信の特性を用いて通信品質を改善するための研究がなされている．重畳符号化の MAC プロトコルを考案する研究 [20–22] では，様々な制御フレームにより重畳符号化の持つ同時通信の特性を活かし，通信品質を改善している．他にも，重畳符号化をネットワークコーディングと組み合わせることで通信品質を改善する研究 [19]，重畳符号化をレートレス符号化と組み合わせることで通信品質を改善する研究 [7,8] などが挙げられる．また，重畳符号化の電力割り当てに焦点を置いて理論ベースでスループット性能を向上させている重畳符号化の論文として，[21–23] が挙げられる．しかしながら，これらの研究では，符号化に具体的にどの変調方式を使うかなどは抽象化して式 (1)，式 (2) に従って理想的に重畳符号化が実現できることを前提に，シミュレーションよる評価がなされている．

2.2 ソフトウェア無線機

新しい無線 LAN 技術をソフトウェア無線を用いて実証する研究が活発化している．無線 LAN を実証主体で研究することで，無線 LAN 技術の進化を加速することができる．

図 3 USRP N200

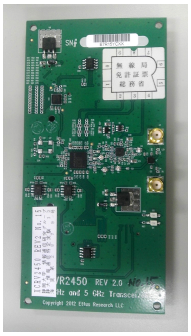

図 4 XCVR2450

1984 年に無線 LAN の最初のコンセプトが提示されてから [25,26]，無線通信の高速化に向けてシャノン限界に近づくべく変復調方式を中心に研究が進められてきた．近年ではレートレス符号化を用いた Spinal Code [9] といったシャノン限界に迫る性能の方式も実証されており，変復調方式単体の研究は成熟を迎えつつある．今後は MIMO [3,4,6] や全二重通信 [14,27] のようなハードウェアや干渉除去を組合わせた手法や，動画圧縮などのアプリケーション層と物理層を連携させた手法 [28,29] など複数の層に跨った手法が重要になってくると考えられる．

複数の層に跨った通信プロトコルの研究開発をする場合，ソフトウェア無線技術を用いて実証主体で研究を進めるアプローチが有効である．研究の専門領域自体も跨るため，必要となる知識も広範になり，抽象化したモデルを前提としたシミュレーション主体のアプローチではモデル間の齟齬を全て検証するのが困難だからである．特に無線通信では，アンプ，回路ノイズ，水晶の精度，環境，アンテナ，トラヒック，バッファサイズ，処理遅延など多くのパラメータが複雑に関わってくるため，研究者が扱っているモデルと実際の通信との乖離が大きい．

例えば，ZigZag Decoding [30] や CRMA [31] では，物理層の干渉除去と MAC 層の衝突回避が密接に関わった手法である．物理層の研究者から見ると物理モデルとして抽象化すると干渉除去を用いただけの手法に見える．MAC 層の研究者から見ると本当に実現可能なのか判断するのが困難であることが多い．これらの複数の層に跨った通信プロトコルを実証することは現実から離れて抽象的なモデルの上で議論することよりも実現可能性に関する説得力が高いだけでなく，新たなモデルを構築するのにも貢献する．なお，ZigZag Decoding や CRMA は USRP [32] や GNU Radio [33] を用いて実際に実装されている．

表 1 USRP N200 の諸元

interface		Gigabit Ethernet
FPGA		Xilinx Spartan 3A-DSP 1800
AD/DA	ADC	Dual 100 MS/s, 14-bit
	DAC	Dual 400 MS/s, 16-bit
SRAM		1MByte

2.3 重畳符号化の実装上の課題

重畳符号化においても，ソフトウェア無線機による実証が進んでいる．現在のところ，重畳符号化をソフトウェア無線に実装した研究として文献 [8,16,17] が挙げられる．しかしながら，文献 [8,16,17] によって重畳符号化の全てが完成されたわけではないにも関わらず，実装の詳細が示されていない．

文献 [16,17] では，USRP と BPSK，QPSK，16QAM を用いた重畳符号化を時分割多重方式で実証している．文献 [16] によれば，ファーストレイヤとセカンドレイヤに割り当てている電力が近ければ近いほどパケットエラーレート（PER: Packet Error Rate）が悪化する．ファーストレイヤとセカンドレイヤに割り当てらる電力が 1 対 1 の場合にはほとんど通信ができない．しかしながら，式 (1)，式 (2) によれば，ファーストレイヤとセカンドレイヤに割り当てられる電力が 1 対 1 の場合でも通信可能なはずである．

一方で，文献 [8] では全く異なる結果が得られている．文献 [8] では，レートレス符号化である Strider [7] を前提とした重畳符号化によって高いスループットを実現する MAC プロトコルである AutoMAC が提案されている．USRP 上に実装してテストベッドでの評価によって実証されている．しかしながら，重畳符号化ではファーストレイヤとセカンドレイヤに割り当てる電力によって性能が変わるにも関わらず，文献 [8] でファーストレイヤとセカンドレイヤに割り当てられている電力は 1 対 1 に固定されている．

それに対して文献 [23] では，式 (1)，式 (2) を前提として，ファーストレイヤとセカンドレイヤに割り当てる最適な電力の値がトラヒックや端末の位置によって異なることが示されている．文献 [23] ではトラヒックや端末の位置によって電力と送信バッファサイズを動的に調整することで高いスループットを実現している．しかしながら，文献 [23] での評価は計算機シミュレーションであり，実現方法は示されていない．

3. 実装環境

2.3 節に示したように，重畳符号化の性能は実現方法に強く依存するにも関わらず，これまでの研究では実装の詳細が示されていない．他の技術者，研究者が再現可能な形で実装の詳細を明らかにすることは，重畳符号化を実用化に近づけるために必須である．このような観点から，本稿では，重畳符号化の基礎的な部分をソフトウェア無線技術を用いて実装した上で，実装の詳細を示す．

3.1 ハードウェア: USRP N200

重畳符号化を実装する機器として，ソフトウェア無線機

である USRP N200 を用いた．USRP は文献 [30,34–37] など多くの研究で利用されているソフトウェア無線機である．

図 3 に USRP N200 の外観を示す．USRP N200 は，本体基板に周波数ごとに対応したドーターボードを接続することで，DC∼3GHz や 4.9GHz∼5.85GHz 間の周波数の変更が可能である．周波数の細かな設定はソフトウェアで設定する事も可能となっている．表 1 に USRP N200 の諸元を示す．外部インタフェースとしてギガビットイーサネット，FPGA として Xilinx 社の Spartan 3A-DSP 1800，14 [bit]・100 [MS/s] の AD 変換器，16 [bit]・400 [MS/s] の DA 変換器，1 [MBytes] の SRAM を備えている．USRP N200 では，ギガビットイーサを介して FPGA にファームウェアやプログラムを書き込むこともできる．

図 4 に，本稿の実装で用いるの XCVR2450 を示す．XCVR2450 は 2.4GHz 帯と 5GHz 帯に対応したドーターボードであり，2.3 [GHz]∼2.9 [GHz] と 4.9 [GHz]∼5.85[GHz] の送受信が可能である．XCVR2450 では，100 [mW] で送信する事が可能であるため，電波法の規定で技術基準適合証明などの取得が必要となる．本稿では，特定実験試験局の免許（免許番号：海実第 2878∼2879 号）を取得し，送信実験を行った．

3.2 ソフトウェア: UHD と C++

USRP を扱う際には，オープンソースの信号処理ライブラリである GNU Radio を用いることが一般的である．GNU Radio は C++で記述された信号処理ブロックを Python で接続することで無線機を実装することが可能なソフトウェアフレームワークである．C++で記述された信号処理ブロックは，それぞれ入力と出力のインタフェースを持ち，SWIG というラッパを用いて Python から呼び出せる形に加工されている．

しかしながら，次の 2 つの理由により，今回は C++で作成した自作の信号処理ライブラリと UHD を用いて実装を行った．1 つ目の理由は，GNU Radio で採用されている入出力ブロックモデルで記述されたソフトウェアを他の研究者や開発者が利用する場合に，読解が困難であるからである．入出力ブロックモデルは，GNU Radio に限らず，LabView や Matlab Simulink でも採用されている．複数の入出力ブロックを接続して 1 つの処理を完成させるというプログラム形態は初学者にとっては直感的に理解しやすいというメリットがある．一方で，大規模な処理を記述する場合，どのブロックとどのブロックが接続されているのか，どの順番に処理が実行されるのかを理解するのが困難になるという問題が発生する．

2 つ目の理由は，Python での実行は処理速度が遅いからである．本稿で実装する重畳符号化は最終的には文献 [23] などの MAC 層の研究でも利用することを視野に入れている．GNU Radio では，Python から C++で作成された信号処理ブロックを接続して物理層の処理をする．物理層の処理のみに限定した場合には実質的には C++で処理しているのと変わらない性能が得られる．一方で，MAC 層の処理までを GNU Radio で実装しようとした場合，送信タイミングの制御は入出力ブロックモデルでは記述するのが困難であるため，Python で記述せざるを得ない．結果として，MAC 層で求められる送信タイミングの制御が困難となる．

```
#include <uhd/utils/safe_main.hpp>
#include <uhd/usrp/multi_usrp.hpp>
#include <stdio.h>

int UHD_SAFE_MAIN(int argc, char *argv[]){
    return EXIT_SUCCESS;
}
```

図 5 C++のソースコード

図 5 に UHD を用いて USRP を動作させるための最小限の C++ のソースコードを示す．UHD を用いて自作の信号処理プログラムと USRP を動作させるためには，まず，`multi_usrp.hpp` と `safe_main.hpp` をインクルードする．次に，メイン関数である `int UHD_SAFE_MAIN(int argc, char **argv)` に処理を記述する．

4. UHD と C++による BPSK を用いた重畳符号化通信の実装

2 節で述べたように，重畳符号化では実装形態によってはファーストレイヤとセカンドレイヤに割り当てられた電力が近いときにファーストレイヤとセカンドレイヤの信号の分離ができなくなる [17]．本節では，信号の分離ができない重畳符号化の例としてファーストレイヤとセカンドレイヤが共に BPSK の場合の実装を示す．

4.1 全体像

図 6 に実装の全体像を示す．実装には送信機と受信機として USRP N200 を 2 台，PC を 2 台使用する．送信側では PC でファーストレイヤとセカンドレイヤの信号を重畳し，重畳信号を送信側の USRP を介して送信する．受信側では，USRP を介して受信した重畳信号をそのままデコードすることでファーストレイヤを得て，デコードしたファーストレイヤから再度エンコードしてファーストレイヤの信号を作成する．受信した重畳信号からファーストレイヤ信号を減算してセカンドレイヤ信号を得て，セカンドレイヤ信号をデコードすることでセカンドレイヤを得る．

4.2 フレーム構成

図 7 にフレーム構成を示す．ヘッダ部は入力信号が I/Q 共に 0 の入力無しが 4 [Bytes]，プリアンブルが 4 [Bytes]，パケットの始まりを示すスタートコード 4 [Bytes] で構成さ

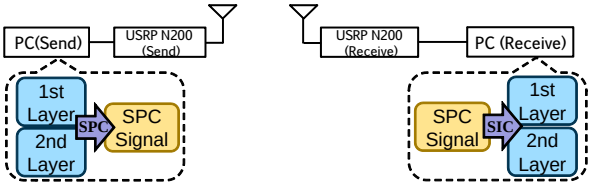

図 6 実装の全体像

入力無し 4Byte	プリアンブル 4Byte	スタートコード 4Byte	ペイロード 1472Byte
-1-1-1-1-1-1-1-1 × 4	0xFF 0xFF 0xFF 0xFF	0xCC 0xCC 0xAA 0xAA	random

図 7 フレーム構造

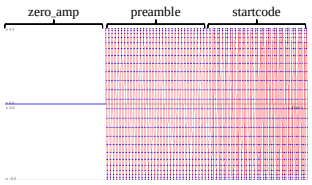

図 8 ヘッダ部の信号

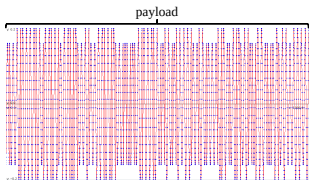

図 9 ペイロード部の重畳信号

れ，ペイロード部は 1472 [Bytes] とし，合計 1484 [Bytes] で構成した．作成される信号のサンプル数は入力なしが 4×8 サンプル，スタートコード部が 4×8 サンプル，ペイロード部が 1472×8 サンプル，合計 1484×8 サンプルである．

4.3 送信機

Algorithm 1 に，送信機の動作を示す．送信機では大きく分けてヘッダ信号の作成，重畳信号の作成，送信の 3 つの動作を行う．*Algorithm* 1 の 1 行目から 9 行では，1 つ目の動作であるヘッダ信号の作成を表している．図 8 にヘッダ信号を示す．図 7 に記述されている入力無し，プリアンブル，スタートコードのビット列 (*header_bit*) を用いてヘッダ信号 (*header_signal*) を作成する．I/Q 共に全て 0 の入力無しには，−1 を便宜的に割り当てている．*header_bit* の値に従って BPSK 信号を *header_signal* に代入する．*header_bit* が 0 ならば 1 の信号を，1 ならば −1 の信号を，−1 ならば 0 を *header_signal* に代入する．

Algorithm 1 の 10 行目から 26 行目は，2 つ目の動作である重畳信号の作成を表している．図 9 に重畳した信号を示す．*Algorithm* 1 の 10 行目から 16 行目では，BPSK 信号に P_{1st} を乗算することで，P_{1st} の振幅のファーストレイヤ信号 (*1st_layer_signal*) を作成する．*Algorithm* 1 の 17 行目から 23 行目では，BPSK 信号に P_{2nd} を乗算することで，P_{2nd} の振幅のセカンドレイヤ信号 (*2nd_layer_signal*) を作成する．*Algorithm* 1 の 24 行目から 26 行目は，*1st_layer_signal* と *2nd_layer_signal* を加算することでペイロード部となる重畳信号 (*payload_signal*) を作成する．

Algorithm 1 の 27 行目と 28 行目は，3 つ目の動作である送信を表している．作成したヘッダ信号 (*header_signal*),

Algorithm 1 送信機のアルゴリズム

1: **for** $i = 0$ to *header_bit* size **do**
2: **if** *header_bit*[i] is 0 **then**
3: *header_signal*[i] ← 1
4: **else if** *header_bit*[i] is 1 **then**
5: *header_signal*[i] ← -1
6: **else if** *header_bit*[i] is −1 **then**
7: *header_signal*[i] ← 0
8: **end if**
9: **end for**
10: **for** $i = 0$ to *1st_layer_bit* size **do**
11: **if** *1st_layer_bit*[i] is 0 **then**
12: *1st_layer_signal*[i] ← P_{1st} · 1
13: **else**
14: *1st_layer_signal*[i] ← P_{1st} · -1
15: **end if**
16: **end for**
17: **for** $i = 0$ to *2nd_layer_bit* size **do**
18: **if** *2nd_layer_bit*[i] is 0 **then**
19: *2nd_layer_signal*[i] ← P_{2nd} · 1
20: **else**
21: *2nd_layer_signal*[i] ← P_{2nd} · -1
22: **end if**
23: **end for**
24: **for** $i = 0$ to *1st_layer_bit* size **do**
25: *payload_signal*[i]
← *1st_layer_signal*[i] + *2nd_layer_signal*[i]
26: **end for**
27: send(*header_signal*)
28: send(*payload*)

```
std::string args = "IP address";
uhd::usrp::multi_usrp::sptr usrp =
                    uhd::usrp::multi_usrp::make(args);
usrp->set_tx_rate(samplingrate);
usrp->set_tx_freq(frequency);
usrp->set_tx_gain(tx_gain, 0);
uhd::stream_args_t stream_args("fc64");
uhd::tx_streamer::sptr tx_stream =
                      usrp->get_tx_stream(stream_args);
uhd::tx_metadata_t md;
md.start_of_burst = false;
md.end_of_burst = false;
md.has_time_spec = false;

// send
tx_stream->send(&tb_wave_stream_header.front(),
                tb_wave_stream_header.size() , md);
tx_stream->send(&tb_wave_stream_spc.front(),
                tb_wave_stream_spc.size() , md);
```

図 10 最小限の送信コード

ペイロード信号 (*payload_signal*) を USRP から送信する．図 10 に送信を行うための最小限のソースコードを示す．まず，対象となる USRP に割り当てられた IP アドレスを引数として `multi_usrp.hpp` 内の `multi_usrp::make` 関数を呼び出すことで USRP オブジェクト (`usrp`) を作成する．USRP オブジェクトを介して，サンプリングレート，周波数，ゲインを設定する事ができる．送信するデータの型情報として `fc64` (complex 型の double) を引数として USRP

Algorithm 2 受信機のアルゴリズム

```
1: while recv_sample ← recv() do
2:     start_position
         ← convolution of recv_sample and startcode_signal
3:     if start_position ≤ 0 then
4:         for i = 0 to 1472 · 8 do
5:             spc_signal[i] ← recv_sample[start_position + i]
6:         end for
7:         phase correction of spc_signal
8:         packet_decode(spc_signal)
9:     end if
10: end while
```

オブジェクトの `tx_stream` 関数を実行することで，送信ストリームオブジェクト (`tx_stream`) が作成される．送信信号を引数として送信ストリームオブジェクトの `send` 関数を実行することで USRP から送信信号が送信される．送信時に `tx_metadata_t` を用いることで，USRP に対して信号の送信・終了，送信時刻などの情報を設定することもできる．

4.4 受信機

Algorithm 2 に受信機の動作を示す．受信機では大きく分けて受信フレームの切り出し，受信フレームのデコードの 2 つの動作を行う．

受信フレームの切り出し

受信フレームの切り出しは，受信信号の取得，畳込み演算，信号補正の 3 つから構成される．*Algorithm* 2 の 1 行目は，切り出し内の 1 つ目の動作である受信信号の取得を表している．USRP から受信信号 (*recv_sample*) を取得する．図 12 に USRP から受信した信号を，図 11 に受信を行うための最小限のソースコードを示す．送信側と同様，対象となる USRP に割り当てられた IP アドレスを引数として `multi_usrp.hpp` 内の `multi_usrp::make` 関数を呼び出すことで USRP オブジェクト (`usrp`) を作成する．USRP オブジェクトを介して，サンプリングレート，周波数，ゲイン，帯域幅を設定する事ができる．受信するデータの型情報として `fc64` (complex 型の double) を引数として USRP オブジェクトの `rx_stream` 関数を実行することで，受信ストリームオブジェクト (`rx_stream`) が作成される．あらかじめ `get_max_num_samps` 関数を用いて `rx_stream` の最大サンプル数を取得しておき，受信信号を格納する配列を引数として受信ストリームオブジェクトの `recv` 関数を実行することで USRP から受信信号を受け取る．受信には，受信したい数値データ配列の開始アドレスと受信したい配列の大きさを `rx_metadata_t` を用いて指定することができる．*Algorithm* 2 の 2 行目で，*recv_sample* が 1 パケット (1484×8 サンプル) の時に次の動作に移る．

Algorithm 2 の 3 行目は，切り出し内の 2 つ目の動作である畳み込み演算を表している．*recv_sample* とスタートコード信号 (*startcode_signal*) の畳込み演算から，開始地点を発見する．式 (3) に行った畳込み演算を，図 13 に結果を示す．図 13 から分かるように，スタートコードのある位置でピークが確認できる．

```
std::string args = "IP address";
uhd::usrp::multi_usrp::sptr usrp =
                    uhd::usrp::multi_usrp::make(args);
usrp->set_rx_rate(samplingrate);
usrp->set_rx_freq(frequency, 0);
usrp->set_rx_gain(rx_gain, 0);
usrp->set_rx_bandwidth(bandwidth);
uhd::stream_args_t stream_args("fc64");
uhd::rx_streamer::sptr rx_stream =
                usrp->get_rx_stream(stream_args);
uhd::stream_cmd_t stream_cmd(
      uhd::stream_cmd_t::STREAM_MODE_START_CONTINUOUS);
usrp->issue_stream_cmd(stream_cmd);
const size_t recvsize = rx_stream->get_max_num_samps();
std::vector<std::complex<float> > recv_sample(recvsize);
uhd::rx_metadata_t md;

// recv
rx_stream->recv(&recv_sample.front(),
                recv_sample.size(), md);
```

図 11 最小限の受信コード

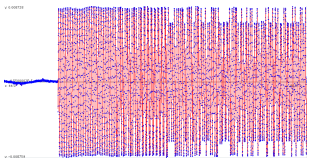

図 12 受信信号

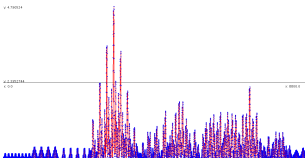

図 13 畳込み演算の結果

$$c_i = \sum_{j=1}^{32} \left(s_{recv_{i+j}} \cdot s_{start_j} \right) \tag{3}$$

c_i は畳込み演算の結果を，i はサンプル数を，s_{recv} は受信信号を，s_{start} はスタートコード信号を表す．c_i が最大値をとる i を開始地点 (*start_position*) に代入する．c_i の最大値がスレッショルドに達さなかった場合には *start_position* に -1 を代入する．*Algorithm* 2 の 4 行目で *start_position* が 0 以上の場合には，*Algorithm* 2 の 5 行目から 7 行目で，発見した開始地点を用いて *recv_signal* から重畳信号 (*spc_signal*) を抜き出す．

Algorithm 2 の 8 行目は，切り出し内の 3 つ目の動作である信号補正を表している．信号補正では，プリアンブル信号を用いて位相オフセットの補正を行う．プリアンブル信号の中央値の偏角を取ることで，位相のズレを見つけ出し補正を行う．式 (4)，式 (5) に補正をかける式を示す．

$$s'_{spc_{real}} = |s_{spc}| \cdot \cos\left(\arctan s_{spc} - \arctan s_{pre}\right) \tag{4}$$

$$s'_{spc_{imag}} = |s_{spc}| \cdot \sin\left(\arctan s_{spc} - \arctan s_{pre}\right) \tag{5}$$

$s'_{spc_{real}}$ と $s'_{spc_{imag}}$ はそれぞれ補正後の重畳信号の実数値と虚数値を示しており，s_{pre} はプリアンブル信号の中央値を示している．

Algorithm 3 パケットデコードのアルゴリズム

```
1: for i = 0 to spc_signal size do
2:     spc_phase[i] ← arctan(spc_signal[i])
3:     if spc_phase[i] > π/2 then
4:         1st_layer_bit[i] ← 0
5:     else
6:         1st_layer_bit[i] ← 1
7:     end if
8: end for
9: for i = 0 to 1st_layer_bit size do
10:    if 1st_layer_bit[i] is 0 then
11:        1st_layer_signal[i] ← P_1st · 1
12:    else if 1st_layer_bit[i] is 1 then
13:        1st_layer_signal[i] ← P_1st · -1
14:    end if
15: end for
16: normalize spc_signal scale
17: for i = 0 to spc_signal do
18:    2nd_layer_signal[i] ← spc_signal[i] − 1st_layer_signal[i]
19: end for
20: for i = 0 to 2nd_layer_signal size do
21:    2nd_layer_phase[i] ← arctan(2nd_layer_signal[i])
22:    if 2nd_layer_phase[i] > π/2 then
23:        2nd_layer_bit[i] ← 0
24:    else
25:        2nd_layer_bit[i] ← 1
26:    end if
27: end for
```

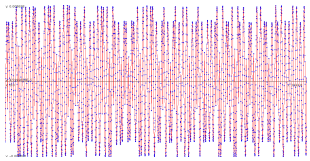

図 14 受信した重畳信号

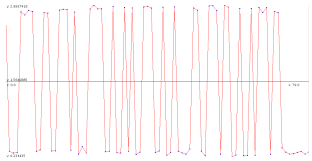

図 15 重畳信号の位相

受信フレームのデコード

Algorithm 2 の 9 行目は，パケットのデコードを表している．*Algorithm* 3 にデコードの動作を示す．受信フレームのデコードは，ファーストレイヤのビット判定，ファーストレイヤ信号の復元，振幅調整，セカンドレイヤのビット判定の 5 つから構成される．*Algorithm* 3 の 1 行目から 8 行目は，デコード内の 1 つ目の動作であるファーストレイヤのビット判定を表している．図 14 に受信した重畳信号を，図 15 に重畳信号の位相を示す．*Algorithm* 3 の 2 行目は，*spc_signal* の偏角を重畳信号の位相 (*spc_phase*) に代入している．*Algorithm* 3 の 3 行目から 7 行目は，*spc_phase* の絶対値が $\pi/2$ より大きければビット 0，$\pi/2$ 以下であればビット 1 と判定し，ファーストレイヤのビット列 (*1st_layer_bit*) に代入している．

Algorithm 3 の 9 行目から 15 行目は，デコード内の 2 つ目の動作であるファーストレイヤ信号の復元を表している．図 16 に復元したファーストレイヤ信号を示す．取得した *1st_layer_bit* を利用してファーストレイヤ信号 (*1st_layer_signal*) を復元する．BPSK 信号に P_{1st} を乗算

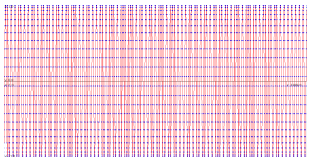

図 16 ファーストレイヤの信号

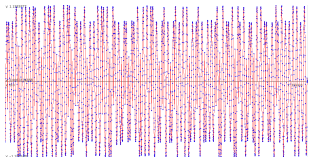

図 17 正規化した重畳信号

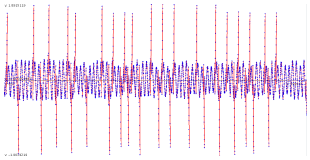

図 18 セカンドレイヤの信号

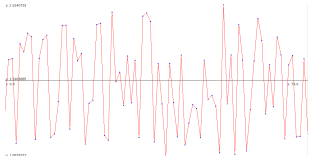

図 19 セカンドレイヤの位相

することで，P_{1st} の振幅の *1st_layer_signal* を復元する．

Algorithm 3 の 16 行目は，デコード内の 3 つ目の動作である振幅調整を表している．振幅調整では，*spc_signal* の正規化を行う．正規化は，*spc_signal* の絶対値の平均値で *spc_signal* を除算する．図 17 に正規化した重畳信号を示す．式 (6)，式 (7) に正規化の計算を示す．

$$r_{avg} = \frac{\sum_{i=0}^{l_{spc}} |s'_{spc_i}|}{l_{spc}} \tag{6}$$

$$s''_{spc_i} = \frac{s'_{spc_i}}{r_{avg}} \tag{7}$$

r_{avg} は *spc_signal* の絶対値の平均値を，l_{spc} は重畳信号のサンプル数を，s'_{spc_i} は位相オフセットを補正した重畳信号を，s''_{spc_i} は正規化後の重畳信号を示している．

Algorithm 3 の 17 行目から 27 行目は，デコード内の 5 つ目の動作であるセカンドレイヤのビット判定を表している．図 18 にセカンドレイヤ信号を，図 19 にセカンドレイヤの位相を示す．*Algorithm* 3 の 17 行目から 19 行目は，正規化した *spc_signal* から *1st_layer_signal* を減算し，セカンドレイヤ信号 (*2nd_layer_signal*) に代入している．*Algorithm* 3 の 21 行目は，*2nd_layer_signal* の偏角をセカンドレイヤ信号の位相 (*2nd_layer_phase*) に代入している．*Algorithm* 3 の 22 行目から 26 行目は，*2nd_layer_phase* の絶対値が $\pi/2$ より大きければビット 0，$\pi/2$ 以下であればビット 1 と判定し，セカンドレイヤのビット列 (*2nd_layer_bit*) に代入している．

5. UHD と C++による $(0, \pi/2)$-BPSK を用いた重畳符号化通信の実装

重畳符号化の実装論文 [18] に示されているように，電力割り当てによっては信号ダイヤグラム上の点が重なり，重畳符号化での通信性能が著しく劣化する場合が存在する．例として，式 1，2 の P_{1st}，P_{2nd} の電力割り当てが $P_{1st} = P_{2nd}$ の時を考えてみる．$P_{1st} = P_{2nd} = P$ と置いたとき，2 節の式 2 に従うと，基地局に近い方の端末に対する通信容量 C_{AC} は以下のように表すことができる．

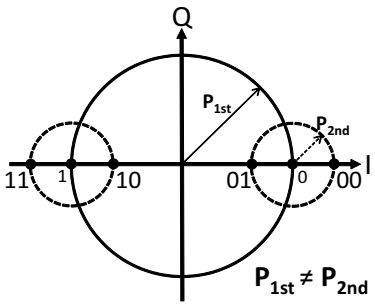

図 20 BPSK を用いた重畳符号化の信号ダイヤグラム

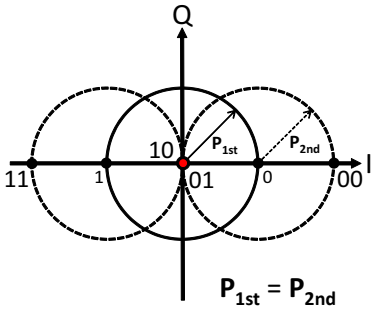

図 21 BPSK を用いた重畳符号化の信号ダイヤグラム

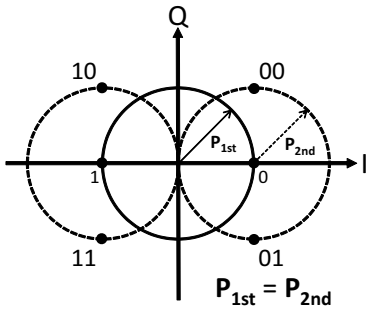

図 22 $(0, \pi/2)$-BPSK を用いた重畳符号化の信号ダイヤグラム

$$C_{AC} = log\Big(1 + \frac{P|h_{AC}|^2}{N}\Big) \tag{8}$$

式 (8) から分かるように，$P_{1st} = P_{2nd}$ であったとしても，干渉除去が成功してファーストレイヤとセカンドレイヤの信号が分離できれば通信できるはずである．

しかしながら，ファーストレイヤとセカンドレイヤが共に BPSK の場合にはほとんど通信することができなくなる．図 20 に，$P_{1st} \neq P_{2nd}$ の際に BPSK を用いて重畳符号化した場合の信号ダイヤグラムを示す．各信号点の左側がファーストレイヤ，右側がセカンドレイヤを意味する．図 20 から分かるように，$P_{1st} \neq P_{2nd}$ の時には信号は 4 つとなり，ファーストレイヤとセカンドレイヤを分離できる．

図 21 に，$P_{1st} = P_{2nd}$ の際に BPSK を用いて重畳符号化した場合の信号ダイヤグラムを示す．ファーストレイヤとセカンドレイヤが共に 1 か共に 0 (11 か 00) の場合は信号を判定できるものの，ファーストレイヤとセカンドレイヤの信号が異なる時には信号ダイヤグラムの中心に信号点が来るため，判定できなくなる．

この問題点に対して，重畳するセカンドレイヤの位相を $\pi/2$ ずらした $(0, \pi/2)$-BPSK を用いた重畳符号化を考案し，実装した．図 22 に通常の BPSK と位相を $\pi/2$ ずらした BPSK を重畳した場合の信号ダイヤグラムを示す．図 22 から分かるように，BPSK と $(0, \pi/2)$-BPSK を重畳させることで，$P_{1st} = P_{2nd}$ であっても信号点は 4 つとなり，ファーストレイヤとセカンドレイヤを分離できる．

5.1 送信機

Algorithm 4 に $(0, \pi/2)$-BPSK を用いた重畳符号化の送信機の動作を示す．BPSK を用いた重畳符号化と $(0, \pi/2)$-BPSK を用いた重畳符号化での送信機の違いは，17 行目から 23 行目のセカンドレイヤ信号の作成部分である．位相を $\pi/2$ ずらした BPSK の信号を用いてセカンドレイヤの信号 ($2nd\_layer\_signal$) を作成する．$2nd\_layer\_bit$ が 0 ならば i の信号を，1 ならば $-i$ の信号に P_{2nd} を乗算することで，P_{2nd} の振幅の $2nd\_layer\_signal$ を作成する．

Algorithm 4 $(0, \pi/2)$-BPSK を用いた重畳符号化の送信機のアルゴリズム

```
 1: for i = 0 to header_bit size do
 2:     if header_bit[i] is 0 then
 3:         header_signal[i] ← 1
 4:     else if header_bit[i] is 1 then
 5:         header_signal[i] ← -1
 6:     else if header_bit[i] is −1 then
 7:         header_signal[i] ← 0
 8:     end if
 9: end for
10: for i = 0 to 1st_layer_bit size do
11:     if 1st_layer_bit[i] is 0 then
12:         1st_layer_signal[i] ← P_1st · 1
13:     else
14:         1st_layer_signal[i] ← P_1st · -1
15:     end if
16: end for
17: for i = 0 to 2nd_layer_bit size do
18:     if 2nd_layer_bit[i] is 0 then
19:         2nd_layer_signal[i] ← P_2nd · i
20:     else
21:         2nd_layer_signal[i] ← P_2nd · −i
22:     end if
23: end for
24: for i = 0 to 1st_layer_bit size do
25:     payload_signal[i]
           ← 1st_layer_signal[i] + 2nd_layer_signal[i]
26: end for
27: send(header_signal)
28: send(payload)
```

5.2 受信機

Algorithm 5 に $(0, \pi/2)$-BPSK を用いた重畳符号化のパケットデコードの動作を示す．BPSK を用いた重畳符号化と $(0, \pi/2)$-BPSK を用いた重畳符号化での送信機の違いは，21 行目のセカンドレイヤ信号からセカンドレイヤの位相変換の部分である．セカンドレイヤ信号 ($2nd\_layer\_signal$) の偏角から $\pi/2$ を減算し，セカンドレイヤの位相 ($2nd\_layer\_phase$) に代入している．

Algorithm 5 $(0, \pi/2)$-BPSK を用いた重畳符号化の
パケットデコードのアルゴリズム

for $i = 0$ to spc_signal size **do**
  $spc\_phase[i] \leftarrow \arctan(spc\_signal[i])$
  if $spc\_phase[i] > \pi/2$ **then**
    $1st\_layer\_bit[i] \leftarrow 0$
  else
    $1st\_layer\_bit[i] \leftarrow 1$
  end if
end for
for $i = 0$ to $1st\_layer\_bit$ size **do**
  if $1st\_layer\_bit[i]$ is 0 **then**
    $1st\_layer\_signal[i] \leftarrow P_{1st} \cdot 1$
  else if $1st\_layer\_bit[i]$ is 1 **then**
    $1st\_layer\_signal[i] \leftarrow P_{1st} \cdot$ -1
  end if
end for
normalize spc_signal scale
for $i = 0$ to spc_signal **do**
  $2nd\_layer\_signal[i] \leftarrow spc\_signal[i] - 1st\_layer\_signal[i]$
end for
for $i = 0$ to $2nd\_layer\_signal$ size **do**
  $2nd\_layer\_phase[i] \leftarrow \arctan(2nd\_layer\_signal[i]) - \pi/2$
  if $2nd\_layer\_phase[i] > \pi/2$ **then**
    $2nd\_layer\_bit[i] \leftarrow 0$
  else
    $2nd\_layer\_bit[i] \leftarrow 1$
  end if
end for

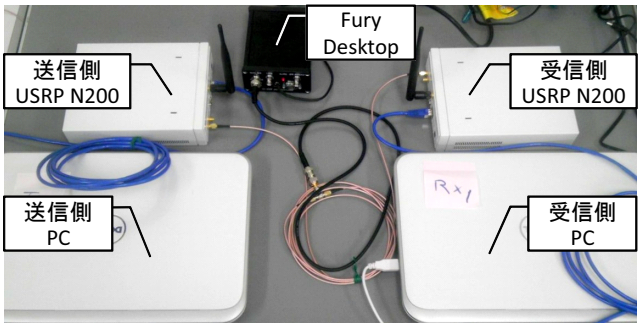

図 23 実験環境

6. 動作検証

6.1 実験構成

アンテナとして VERT2450 を用いた．実験では，4 節に示した BPSK を用いた重畳符号化と，5 節に示した $(0, \pi/2)$-BPSK を用いた重畳符号化の 2 つの変調方式を検証に用いた．

表 2 実験パラメータ

Sampling Rate	195.3125ksps
Modulation of 1st layer	BPSK
Modulation of 2nd layer	BPSK/$\frac{\pi}{2}$shift BPSK
Center Frequency	5.11GHz
Payload size	1472Byte
Packet size	10

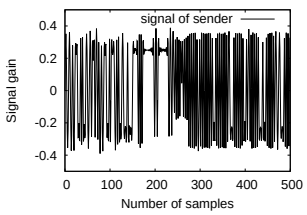

図 24 送信信号

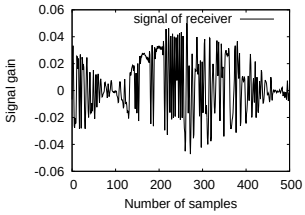

図 25 受信信号

表 2 に実験パラメータを示す．サンプリングレートは 195.3125 [ksps] (デシメーションが 512)，中心周波数は 5.11 [GHz]，ペイロードサイズは 1472 [Bytes] で検証した．図 23 に実験環境を示す．送信側，受信側の PC をギガビットイーサネットケーブルでそれぞれ送信側，受信側の USRP と接続し，外部クロックとして Jackson Labs 社の Fury Desktop [38] を送信側，受信側の USRP と接続した．

6.2 周波数オフセットの検証

USRP N200 では，クロックの精度が低いため，異なる USRP 間で通信するときに中心周波数が異なるという問題が発生する．図 24 に中心周波数が揃っていない場合の送信信号を，図 25 に受信信号を示す．図 25 の受信信号では，中心周波数の違いによってうなりが発生している．

USRP において周波数オフセットを緩和する方法として，周波数をデジタル信号処理で補正する手法と，外部クロックを用いる手法の 2 つが考えられる．本稿の目的は重畳符号化の実装の基礎的の詳細を明らかにすることであるため，外部クロックを用いることとした．

図 26 に，使用するハードウェア構成の違いによる周波数オフセットを示す．図 26 より，外部クロック無しでは約 1200 [Hz]，GPSDO Kit では約 600 [Hz]，Fury Desktop では約 5 [Hz] の周波数オフセットであることが分かる．図 26 の Default は外部クロック無しを意味している．USRP N200 に XCVR2450 を接続しただけの構成となっている．図 26 の GPSDO Kit は Ettus Research 社の GPSDO Kit を USRP N200 に組み込み，USRP N200 同士のクロックは共有していない構成を意味している．GPSDO Kit は，USRP N200 の内部クロックよりも高い精度を備えている．図 26 の Fury Desktop は Jackson 社の Fury Desktop [38] から出力される 10MHz の信号を送信側と受信側の USRP の両方に入力した構成を意味している．図 27 に Fury Desktop の外観を，図 28 に Fury Desktop と USRP N200 の接続方法を示す．図 28 に示しているように，Fury Desktop の 10MHz SINE ポートと USRP N200 の REF IN ポートを接続した．

6.3 電力割り当てに対するビットエラーレート

重畳符号化の基本性能を見ることを目的として，ファー

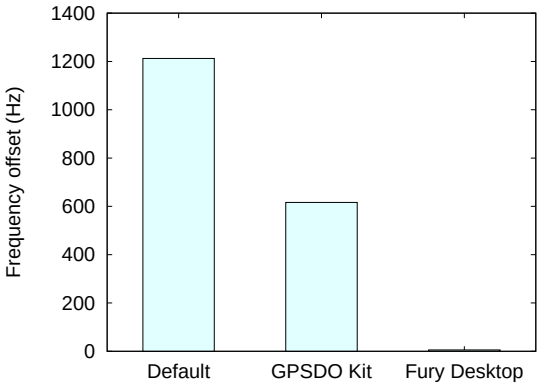

図 26 周波数オフセット計測結果

図 27 Fury Desktop

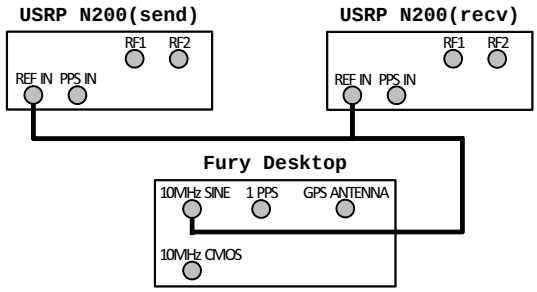

図 28 Fury Desktop と USRP N200 の接続方法

ストレイヤとセカンドレイヤに割り当てる電力を変えた場合のビットエラーレートの評価を行った．送信側の USRP N200 に設定した `tx_gain` は 0 [dB] である．送信側と受信側の USRP N200 を向かい合わせに 1 メートル離して計測した．ファーストレイヤに対する電力の割り当てを 5%ずつ変化させ，各パラメータに対してそれぞれ 100,000 ビット計測した．

図 29 に BPSK を用いた重畳符号化の電力割り当てに対するビットエラーレート，図 30 に $(0, \pi/2)$-BPSK を用いた重畳符号化の電力割り当てに対するビットエラーレートを示す．横軸はファーストレイヤに割り当てた電力の割合を表している．0 がセカンドレイヤに全ての電力を割り当てた場合，1 がファーストレイヤに全ての電力を割り当てた場合を意味している．ビットエラーが計測できなかった場合には便宜的に 10^{-6} の位置に結果をプロットしている．

図 29，図 30 より，次の 2 つのことが分かる．1 つ目は，BPSK を用いた重畳符号化では，ファーストレイヤとセカンドレイヤに対する電力割り当てが近い場合にはファーストレイヤとセカンドレイヤが干渉してビットエラーレートが高くなることである．図 29 の x 軸の 0.5 付近で BPSK のファーストレイヤとセカンドレイヤのビットエラーレートが 10^{-1} 以上になっている．2 つ目は，$(0, \pi/2)$-BPSK を用いた重畳符号化では，ファーストレイヤとセカンドレイヤに対する電力割り当てが近い場合でもビットエラーレートが低いことである．5 節で述べたように，$(0, \pi/2)$-BPSK を用いることでファーストレイヤとセカンドレイヤを分離できているからだと考えられる．

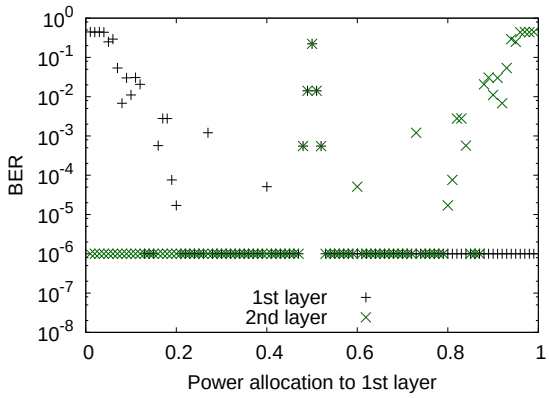

図 29 BPSK を用いた重畳符号化の電力割り当てに対するビットエラーレート

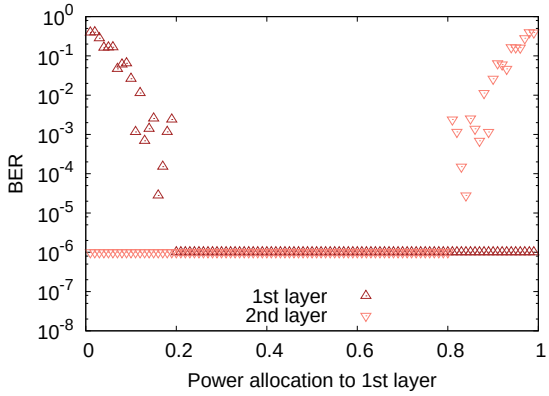

図 30 $(0, \pi/2)$-BPSK を用いた重畳符号化の電力割り当てに対するビットエラーレート

6.4 SN 比に対するビットエラーレート

6.3 節の評価は，通信距離と送信電力を固定しての評価であった．しかしながら，無線通信では通信距離と送信電力が固定であったとしても，チャネルの状態によって SN 比が変動する．このような観点から，ファーストレイヤとセカンドレイヤに割り当てた電力を固定した場合の SN 比に対するビットエラーレートを評価した．送信側と受信側の USRP N200 は向かい合わせに 1 メートル離した．送信側の USRP N200 に設定した `tx_gain` を 0~35 [dB] に 1 [dB] 刻みで変化させ，各評価パラメータに対してそれぞれ

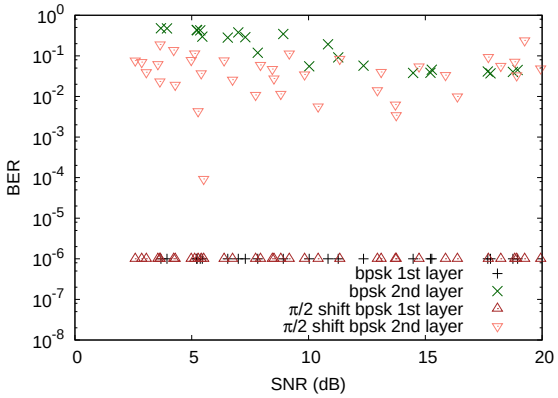

図 31 SN 比に対するビットエラーレート ($P_{1st} : P_{2nd} = 9 : 1$)

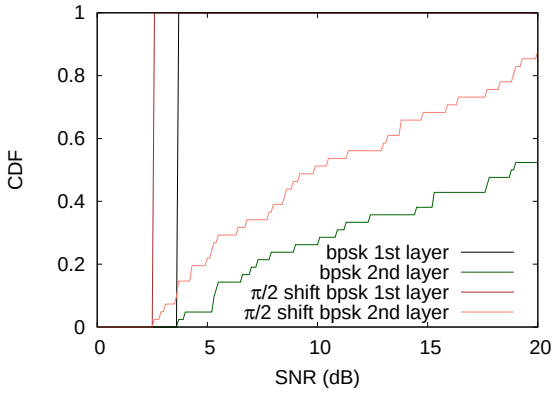

図 32 SN 比に対するビットエラーレート 10^{-3} を満たす計測の累積密度関数 (CDF) ($P_{1st} : P_{2nd} = 9 : 1$)

100,000 ビット計測した．

図 31，図 32 にファーストレイヤとセカンドレイヤに割り当てた電力の比が 9 : 1 である場合の SN 比に対するビットエラーレートの評価結果を示す．図 31 では，横軸が SN 比，縦軸がビットエラーレートである．ビットエラーレートが 0 であった計測は便宜的に 10^{-6} にプロットしている．図 32 では，横軸が SN 比，縦軸がビットエラーレートが 10^{-3} 以上であった計測の累積分布関数を意味している．例えば，SN 比が 1 [dB] で CDF が 0.3 であった場合，SN 比が 1 [dB] 以下の計測点においてビットエラーレートが 10^{-3} 以上であった計測は全体の計測の 30%であることを意味している．

図 31，図 32 より BPSK を用いた重畳符号化と $(0, \pi/2)$-BPSK を用いた重畳符号化どちらもファーストレイヤよりもセカンドレイヤのビットエラーレートのほうが高く，SN 比が高くなるにつれてビットエラーレートが低下していることが分かる．割り当てられている電力が大きい方がビットエラーレートが低くなるのではないかと考えられる．

高い電力が割り当てられているレイヤの方が低いビットエラーレートとなるかどうかを確認することを目的として，ファーストレイヤとセカンドレイヤに割り当てた電力を変えた場合の SN 比に対するビットエラーレートの評価

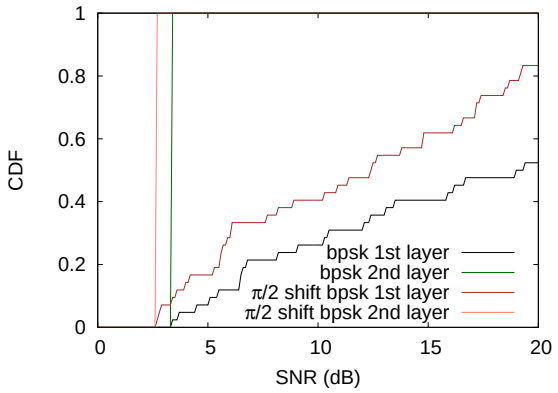

図 33 SN 比に対するビットエラーレート 10^{-3} を満たす計測の累積密度関数 (CDF) ($P_{1st} : P_{2nd} = 1 : 9$)

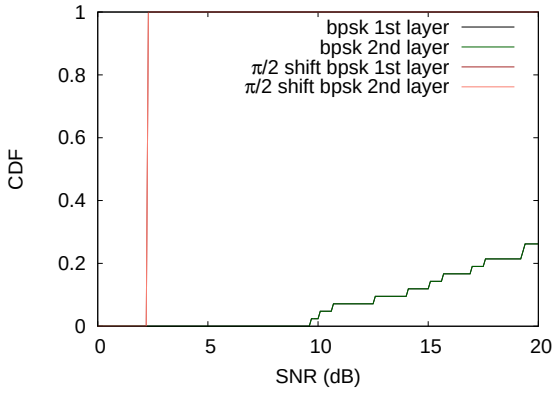

図 34 SN 比に対するビットエラーレート 10^{-3} を満たす計測の累積密度関数 (CDF) ($P_{1st} : P_{2nd} = 1 : 1$)

を行った．図 33，図 34 にファーストレイヤとセカンドレイヤに割り当てている電力の比がそれぞれ 1 : 9，1 : 1 の場合の SN 比に対するビットエラーレートを示す．横軸が SN 比，縦軸がビットエラーレートが 10^{-3} 以上であった計測の累積分布関数を意味している．

図 32, 図 33，図 34 より，BPSK を用いた重畳符号化でも $(0, \pi/2)$-BPSK を用いた重畳符号化でも，電力割り当てが大きいレイヤの方がビットエラーレートが低いことが分かる．ファーストレイヤに割り当てた電力とセカンドレイヤに割り当てた電力の比が 9:1 の場合にはファーストレイヤのビットエラーレートが低く，1:1 の場合にはビットエラーレートが等しく，1:9 の場合にはセカンドレイヤのビットエラーレートが低い．

7. まとめ

本稿では，ソフトウェア無線機である USRP N210 と USRP のドライバである UHD を用いて C++で実装した重畳符号化の基礎的な部分の詳細を示した．重畳符号化におけるファーストレイヤとセカンドレイヤに割り当てた電力が近い場合にファーストレイヤとセカンドレイヤの信号の分離ができない例として BPSK を用いた実装を示した．

また，ファーストレイヤとセカンドレイヤの信号の分離ができる例として，$(0, \pi/2)$-BPSK を用いた実装を示した．動作検証の結果，$(0, \pi/2)$-BPSK を用いた場合ではファーストレイヤとセカンドレイヤに割り当てられた電力が等しい場合でもファーストレイヤとセカンドレイヤを分離できることが分かった．

謝辞

本研究は科研費基盤研究 A (24240009) の助成を受けて行った．

参考文献

[1] 総務省．平成 22 年度情報通信審議会報告．ぎょうせい，2010.

[2] 総務省．平成 24 年度版情報通信白書．ぎょうせい，2012.

[3] David J. Love, Robert W. Heath, and Thomas Strohmer. Grassmannian beamforming for multiple-input multiple-output wireless systems. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (ACM SIGCOMM'09)*, 2009.

[4] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire. Achieving high data rates in distributed MIMO systems. In *Proceedings of the eighteenth annual international conference on Mobile computing and networking (ACM MobiCom'12)*, pp. 41–52, August 2012.

[5] C. Shepard, H. Yu, N. Anand, L. E. Li, T. Marzetta, Y. R. Yang, and L. Zhong. Argos: Practical base stations with large-scale multi-user beamforming. In *Proceedings of the eighteenth annual international conference on Mobile computing and networking (ACM MobiCom'12)*, pp. 53–64, August 2012.

[6] E. Aryafar, M. A. Khojastepour, K. Sundaresan, S. Rangarajan, and M. Chiang. MIDU: Enabling MIMO full duplex. In *Proceedings of the eighteenth annual international conference on Mobile computing and networking (ACM MobiCom'12)*, pp. 257–268, August 2012.

[7] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *Proceedings of the ACM SIGCOMM 2011 Conference on Data Communication (ACM SIGCOMM'11)*, pp. 158–169, August 2011.

[8] A. Gudipati, S. Pereira, and S. Katti. AutoMAC: Rateless wireless concurrent medium access. In *Proceedings of the eighteenth annual international conference on Mobile computing and networking (ACM MobiCom'12)*, pp. 5–16, August 2011.

[9] J. Perry, P. A. Iannucci, K. Fleming, H. Balakrishnan, and D. Shah. Spinal codes. In *Proceedings of the ACM SIGCOMM 2012 Conference on Data Communication (ACM SIGCOMM'12)*, pp. 49–60, october 2012.

[10] Daniel Halperin, Thomas Anderson, and David Wetherall. Taking the sting out of carrier sense: Interference cancellation for wireless LANs. In *Proceedings of the 14th ACM international conference on Mobile computing and networking (ACM MobiCom'08)*, 2008.

[11] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[12] Evan Everett, Melissa Duarte, Chris Dick, and Ashutosh Sabharwal. Empowering full-duplex wireless communication by exploiting directional diversity. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers (IEEE ACSSC'11)*, 2011.

[13] Achaleshwar Sahai, Gaurav Patel, and Ashutosh Sabharwal. Pushing the limits of full-duplex: Design and real-time implementation. Technical report, Rice University Technical Report TREE1104, 2011.

[14] Mayank Jainy, Jung Il Choiy, Tae Min Kim, Dinesh Bharadia, Siddharth Seth, Kannan Srinivasan, Philip Levis, Sachin Katti1, and Prasun Sinha. Practical, real-time, full duplex wireless. In *Proceedings of the 17th annual international conference on Mobile computing and networking (ACM MobiCom'11)*, 2011.

[15] Mellisa Duarte and Ashutosh Sabharwal. Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers 2010 (IEEE ACSSC'10)*, 2010.

[16] Radha Krishna Ganti, Zhenhua Gong, Martin Haenggi, Chia han Lee, Sunil Srinivasa, David Tisza, Sundaram Vanka, and Peter Vizi. Implementation and experimental results of superposition coding on software radio. In *Proceedings of IEEE International Conference on Communications (IEEE ICC'10)*, pp. 1–5, 2010.

[17] S. Vanka, S. Srinivasa, Z. Gong, P. Vizi, K. Stamatiou, and M. Haenggi. Superposition coding strategies: Design and experimental evaluation. In *IEEE transactions on wireless communication*, Vol. 11, pp. 2628–2639, july 2012.

[18] S. Vanka, S. Srinivasa, and M. Haenggi. A practical approach to strengthen vulnerable downlinks using superposition coding. In *Proceedings of IEEE International Conference on Communications (IEEE ICC'12)*, 2012.

[19] Xueyuan Su and S. Chan. High-throughput routing with superposition coding and successive interference cancellation. In *IEEE International Conference on Communications (IEEE ICC'11)*, pp. 1–6, 2011.

[20] Li Erran Li, Richard Alimi, Ramachandran Ramjee, Harish Viswanathan, and Yang Richard Yang. muNet: Harnessing multiuser capacity in wireless mesh networks. In *Proceedings of IEEE International Conference on Computer Communications (IEEE INFOCOM'09)*, pp. 2876–2880, 2009.

[21] Li Erran Li, Richard Alimi, Ramachandran Ramjee, Jingpu Shi, Yanjun Sun, Harish Viswanathan, and Yanga Richard Yang. Extended abstract: Superposition coding for wireless networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking (ACM MobiCom'07)*, pp. 330–333, 2007.

[22] Jiao Feng, Rong Zhang, and Lajos Hanzo. Auction-style cooperative medium access control. In *IEEE Vehicular Technology Conference (IEEE VTC'11)*, pp. 1–5, 2011.

[23] 青木勇太, 猿渡俊介, 渡辺尚. 重畳符号化を用いた無線通信における転送量に基づく電力割当方式. 情報処理学会研究報告，モバイルコンピューティングとユビキタス通信研究会, MBL-64-22, pp. 1–8, November 2012.

[24] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, Vol. 27, pp. 379–423, 1948.

[25] Kaveh Pahlavan. A review of wireless in-house data communication systems. In *Proceedings Computer Networking Symposium*, pp. 129–136, December 1984.

[26] Kaveh Pahlavan and Allen H. Levesque. Wireless data communications. *Proceedings of the IEEE*, Vol. 82, No. 9, pp. 1398–1430, September 1994.

[27] Jung Il Choiy, Mayank Jainy, Kannan Srinivasany, Philip

Levis, and Sachin Katti. Achieving single channel, full duplex wireless communication. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking (ACM MobiCom'10)*, 2010.

[28] Siripuram T. Aditya and Sachin Katti. FlexCast: Graceful wireless video streaming. In *Proceedings of the 17th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'11)*, pp. 277–288, Las Vegas, Nevada, September 2011.

[29] Szymon Jakubczak and Dina Katabi. A cross-layer design for scalable mobile video. In *Proceedings of the 17th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'11)*, pp. 289–300, Las Vegas, Nevada, September 2011.

[30] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (ACM SIGCOMM'08)*, 2008.

[31] Tianji Li, Mi Kyung Han, Apurv Bhartia, Lili Qiu, Eric Rozner, Yin Zhang, and Brad Zarikoff. CRMA: Collision-resistant multiple access. In *Proceedings of the 17th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'11)*, pp. 61–72, Las Vegas, Nevada, September 2011.

[32] Ettus Research. Usrp universal software radio peripheral. http://www.ettus.com/.

[33] GNU Radio. http://gnuradio.org/.

[34] S. Gollakota, S.D. Perli, and D. Katabi. Interference alignment and cancellation: Better receivers for a new wireless mac. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (ACM SIGCOMM'09)*, pp. 159–170, Barcelona, August 2009.

[35] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (ACM SIGCOMM'09)*, pp. 3–14, Barcelona and Spain, August 2009.

[36] X. Zhang and K.G. Shin. DAC: Distributed asynchronous cooperation for wireless relay networks. In *Proceedings of the 29th IEEE Conference on Information Communications (IEEE INFOCOM'10)*, pp. 1064–1072, San Diego California, March 2010.

[37] S. Sen, R.R. Choudhury, and S. Nelakuditi. CSMA/CN: Carrier sense multiple access with collision notification. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (ACM MobiCom'10)*, pp. 25–36, Chicago Illinois, September 2010.

[38] Jackson labs. http://www.jackson-labs.com/.