Perancangan Dan Simulasi *Punctured Convolutional Encoder* Dan *Viterbi Decoder* Dengan *Code Rate* 2/3 Menggunakan Raspberry Pi

Marjan Maulataufik[1], Hertog Nugroho[2]
[1,2]Politeknik Negeri Bandung
Jalan Gegerkalong Hilir, Ciwaruga, Parongpong, Kab Bandung Barat, Jawa Barat 40559
marjanmaulataufik@gmail.com[1], hertog@polban.ac.id[2]

Abstrak *– Convolutional encoder merupakan salah satu metoda teknik error control coding yang memiliki kekurangan, yaitu code rate yang rendah. Pada penelitian ini telah diimplementasikan pengembangan Convolutional encoder yaitu Punctured Convolutional encoder untuk menghasilkan code rate yang lebih tinggi, yaitu code rate 2/3. Punctured convolutional encoder dan viterbi decoder ini telah disimulasikan menggunakan serial komunikasi antara PC/laptop – Raspberry PI. Hasil dari penelitian menunjukkan bahwa punctured convolution codes dengan decoder code rate 2/3 mampu mengoreksi kesalahan bit error untuk 35 bit. Dari evaluasi karakteristik puncture matrix dengan menggunakan generator polynomial 7 & 5, hanya puncture matrix* $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ *yg menghasilkan solusi yang tunggal, sedangkan pada evaluasi karakteristik puncture matrix dengan menggunakan generator polynomial 6 & 7, semua puncture matrix menghasilkan solusi yang tunggal. Kesimpulan dari penelitian ini yaitu Punctured codes berhasil dibuat dengan code rate 2/3 yang dibangun dari code rate ½ menggunakan puncture matrix* $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.

Kata kunci: *Punctured Convolutional encoder, Viterbi Decoder, Code Rate, Puncture Matrix.*

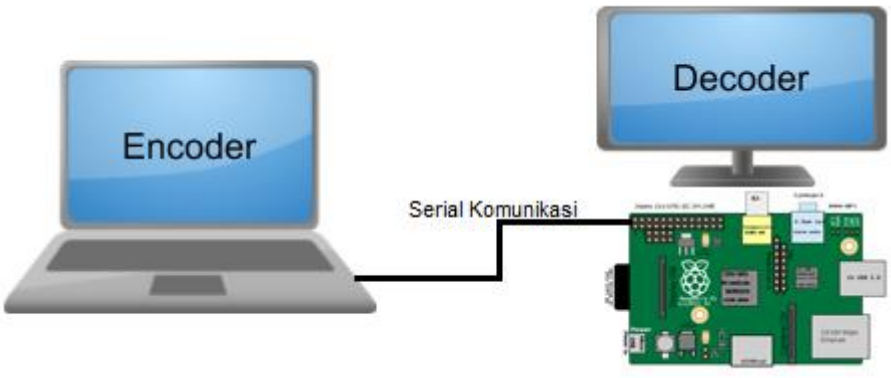
1. Pendahuluan

Teknik *Error control coding* merupakan salah satu solusi untuk mengatasi/meminimalisir kesalahan yang terjadi pada kanal transmisi yang disebabkan oleh derau. Teknik *error control coding* dikenal juga sebagai *error detection and correction* yang memiliki kemampuan untuk mendeteksi dan mengoreksi suatu *error* dengan merekonstruksinya kembali menjadi data semula (*original data*). Salah satu jenis *Error control coding* yaitu *Forward Error Correction* (FEC) *codes* atau *error-correcting codes* yang menggunakan bit – bit tambahan (*redundant*) pada saat pengkodean untuk mengembalikan data semula. Salah satu metoda *error-correcting codes* yaitu *convolutional codes* dengan *viterbi decoder* untuk mendapatkan hasil yang optimal [2].

Convolution code diketahui sebagai salah satu kelas *error-correcting* yang cukup bagus. Ketika digunakan bersamaan dengan *interleaving*, mereka dapat memberikan proteksi yang sangat bagus untuk *fading channels*. *Convolution codes* biasa digunakan untuk transmisi yang kontinyu. Akan tetapi kode – kode tersebut terbatas untuk *code rate* yang rendah, dikarenakan konstruksi *decoder* cukup kompleks jika untuk mencapai *code rate* yang tinggi [3].

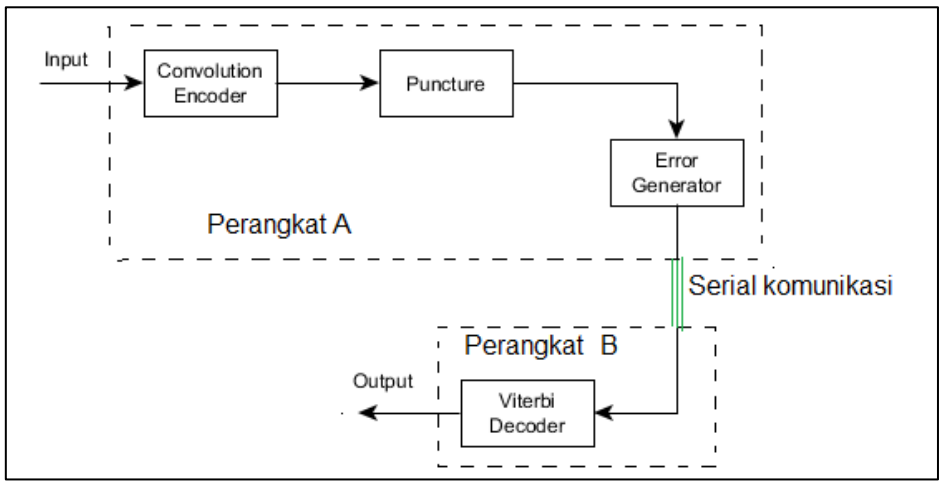
Penelitian ini bertujuan untuk mendapatkan *code rate* 2/3 dan pengaruh *punctured codes* terhadap koreksi *error bit* serta karakteristik *puncture matrix* pada *viterbi decoder* untuk *code rate* 2/3 dengan menggunakan *generator polynomial* [1 1 1] & [1 0 1] atau dalam bilangan oktal [7 & 5] dan *generator polynomial* [1 1 0] & [1 1 1] atau dalam bilangan oktal [6 & 7].

2. Metode Penelitian



Gambar 1. Ilustrasi sistem *Punctured Convolutinal Encoder* dan *Viterbi Decoder*.

Sistem yang telah dibangun dibagi menjadi 2 bagian, seperti yg terlihat pada gambar 1, yang menggambarkan sebuah sistem menghubungkan 2 perangkat dengan menggunakan mode komunikasi serial. Gambar 2 menjelaskan rincian masing-masing perangkat. Perangkat A digunakan untuk proses enkoding dan mengirimkan data ke perangkat B, disaat yang sama perangkat B melakukan *looping* secara kontinyu untuk menerima data dari perangkat A. *Punctured convolutional encoder* dirancang pada perangkat A dengan merancang terlebih dahulu sebuah *convolutional encoder* kemudian dilakukan *puncturing* sesuai *code rate* yang diinginkan, selanjutnya *error* diberikan secara acak yang kemudian ditransmisikan. *Viterbi decoder* dirancang pada perangkat B, kemudian dilakukan proses *decoding* dengan menampilkan proses algoritma *Viterbi* secara animasi hingga didapat hasil dan selanjutnya dibandingkan dengan *input* pada perangkat A.



Gambar 2. Blok diagram sistem *Punctured Convolutional Encoder* dan *Viterbi Decoder*.

Berikut uraian dari masing – masing blok:

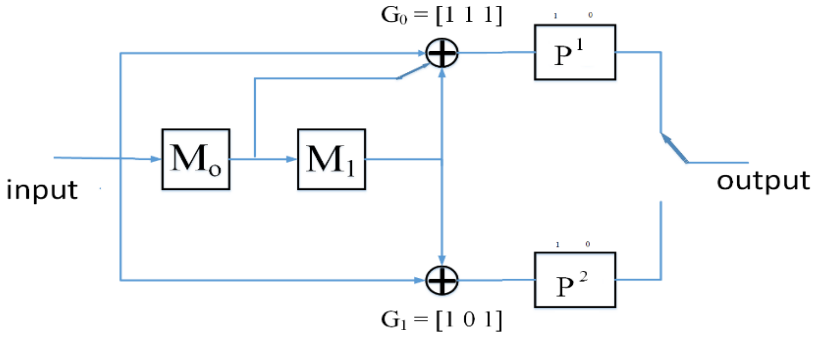
- *Convolutional encoder*, pengkodean input dengan *rate* ½ dan *generator polynomial* [7 5] dan [6 7].
- *Puncture*, input yang telah terkodekan dilakukan proses *puncturing* atau membuang bit berdasarkan *puncture matrix* untuk menghasilkan *code rate* 2/3.
- *Error Generator*, pembangkit bit *error* yang dimasukkan dalam bit input yang telah terkodekan secara acak.

ISBN: 978-602-512-810-3

- *Viterbi decoder*, teknik untuk melakukan *decoding* dari bit yang telah terkodekan menggunakan *convolutional encoder* salah satunya menggunakan algoritma Viterbi. Algoritma viterbi diterapkan pada *decoder* dengan memanfaatkan *trellis diagram* dari *convolution codes*. [4]

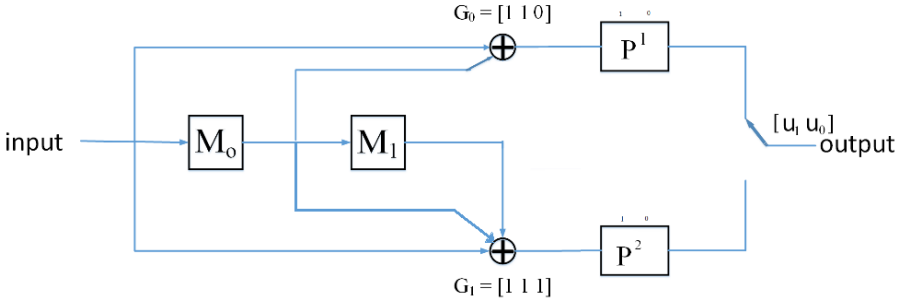
2.1 Perencanaan konfigurasi *punctured convolutional encoder*

Pemilihan parameter untuk konfigurasi enkoder didasarkan kepada beberapa pertimbangan mengacu kepada kebutuhan yang akan digunakan. Parameter yang digunakan untuk perancangan *punctured convolutional encoder* adalah membangun *code rate* 2/3 dari *code rate* ½ dengan menggunakan *constraint length* 3, *Generator Polynomial* 7 & 5 seperti diperlihatkan pada gambar 3.



Gambar 3. Rangkaian *Punctured convolutional encoder* dengan G[7 5].

dan *Generator Polynomial* 6 & 7, seperti diperlihatkan pada gambar 4.



Gambar 4. Rangkaian *Punctured Convolutional encoder* dengan G[6 7].

dengan *puncture matrix* berikut:

$$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}, \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}, \text{dan } \begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}$$

atau dalam bentuk implementasinya:

P1 = [1 1], [0 1], [1 0], dan [1 1].

P2 = [0 1], [1 1], [1 1], dan [1 0].

Dalam artian, bit ‘1’ merepresentasikan bahwa bit input yang terkonvolusi (*codewords*) diteruskan menjadi sebuah output, sedangkan bit ‘0’ berarti *codewords* dibuang (*punctured*).

Berdasarkan *puncture matrix* diatas, untuk setiap *matrix* dilakukan perhitungan secara manual proses *puncturing* dengan bit – bit input yang memiliki jumlah yang bervariasi. Jumlah bit input yang diuji disajikan pada tabel 1.

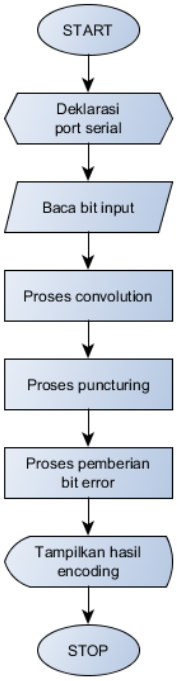
ISBN: 978-602-512-810-3

Tabel 1. Bit – bit input acak yang akan diuji.

Jumlah Bit	Input
2	10
5	10011
8	01100001
11	10010111110
25	1110 0010 1010 0111 0101 00001
35	0011 0101 1010 0100 1110 0001 1110 0100 010

2.2 Perancangan *punctured convolutional encoder*

Subsistem perangkat A merupakan *punctured convolutional encoder* yang disimulasikan pada perangkat PC/Laptop dengan sebuah aplikasi yang dibangun menggunakan bahasa pemrograman Python yang bertujuan untuk melakukan pengkodean bit input serta menyisipkan bit *error* lalu dikirimkan melalui port serial. Perancangan aplikasi yang mensimulasikan *punctured convolutional encoder* direpresentasikan dalam *flowchart* (Gambar 5).

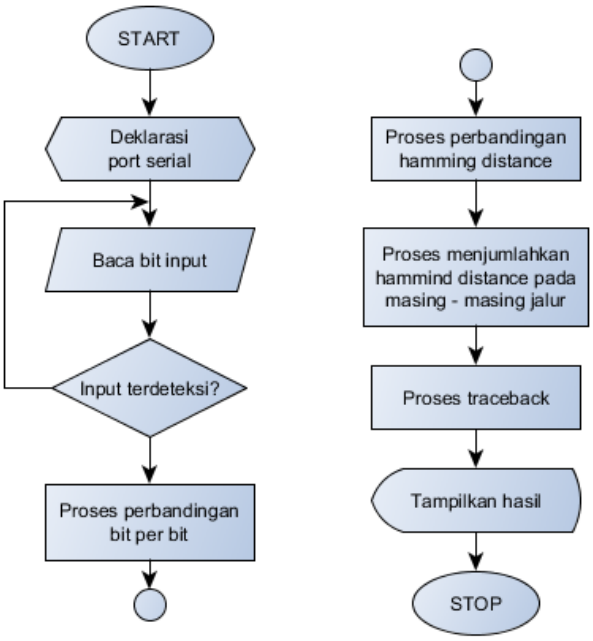


Gambar 5. *Flowchart Punctured Convolutional encoder*.

2.2 Perancangan *viterbi decoder*

Perancangan untuk *viterbi decoder* dilakukan pada perangkat B dengan tujuan menerima data dari perangkat A melalui serial komunikasi dan diimplementasikan menggunakan perangkat Raspberry PI. Pada subsistem ini, *puncture matrix* yang digunakan pada bagian *encoder* digunakan kembali untuk menghasilkan output yang selaras. Proses *decoding* pada program di perangkat B, disimulasikan dalam bentuk sebuah animasi yang menggambarkan *trellis diagram* hingga mendapatkan jalur yang sesuai dan mendapatkan hasil yang sama dengan perhitungan secara empiris. Gambar 6 memperlhatkan *flowchart* dari proses dekodernya.

ISBN: 978-602-512-810-3



Gambar 6. *Flowchart Viterbi Decoder*.

3. Hasil dan analisis

3.1 Hasil pengujian sistem

Pengujian *puncture matrix* dilakukan pada masing – masing *generator polynomial* untuk mendapatkan *puncture matrix* yang paling baik. Pengujian *puncture matrix* dibuktikan secara empiris berdasarkan teori yang telah dikemukakan pada pendahuluan, hasil dari pengujian *puncture matrix* pada masing-masing *generator polynomial* diringkas dan disajikan pada tabel 2.

Tabel 2 Hasil pengujian *puncture matrix* dengan G[7 5].

Puncture Matrix	**Hasil**
$\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}$	H*amming distance* tidak didapatkan pada awal proses *decoding* (Tidak optimal)
$\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$	Memiliki 2 jalur dengan hamming distance terkecil yang sama (ambigu).
$\begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$	Memiliki 2 jalur dengan hamming distance terkecil yang sama (ambigu).
$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}$	Benar

Untuk hasil pengujian *puncture matrix* dengan G[6 7], semua konfigurasi *puncture matrix* pada proses *decoding* mendapatkan **hasil yang sama benar atau sesuai dengan input.**

Pengujian *punctured codes* dilakukan dengan menggunakan *puncture matrix* yang paling baik berdasarkan hasil pengujian sebelumnya. Pengujian ini dilakukan untuk mendapatkan tingkat keberhasilan dalam mengoreksi kesalahan berdasarkan jumlah input, hasil dari pengujian ini disajikan pada tabel 3 dengan indikasi warna merah yaitu bit yang tidak terkoreksi.

ISBN: 978-602-512-810-3

Tabel 3. Hasil pengujian *punctured codes* terhadap *error*.

Input	Bit *error*	Output
2 = 10	0	10
	1	01
	2	11
5 = 10011	0	10011
	1	10011
	2	11011
8 = 01100001	0	01100001
	1	01100001
	2	01100101
11 = 10010111110	0	10010111110
	1	10010111110
	2	11101111110
25 = 1110 0010 1010 0111 0101 00001	0	11100010101001110101000 01
	1	11100010101001110101000 01
	2	00010010101001110101000 01
35 = 0011 0101 1010 0100 1110 0001 1110 0100 010	0	00110101101001001110000111100100010
	1	00110101101001001110000111100100010
	2	00110101101001001110000111100100010

3.2 Analisa

Berdasarkan hasil pengujian *puncture matrix*, dengan menggunakan *generator polynomial* 7 & 5 masing – masing *puncture matrix* memiliki respon yang berbeda terhadap hasil proses *decoding* sehingga untuk pengujian '*punctured codes* terhadap *error'* digunakan *puncture matrix* yang paling baik yaitu $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$. Sedangkan dengan menggunakan *generator matrix* 6 & 7, semua konfigurasi *puncture matrix* memiliki respon yang sama dan membuktikan bahwa kombinasi *generator polynomial* yang paling bagus untuk *constraint length* 3 yaitu 6 & 7 sesuai dengan penelitian J. Busgang [9].

Berdasarkan hasil tabel 3 pengujian *punctured codes* terhadap *error*, *decoder* tidak mampu melakukan pengkoreksian *error* terhadap *codewords* yang dihasilkan oleh *punctured encoder* dikarenakan pada dasarnya *punctured codes* tidak digunakan untuk *error-control* akan tetapi digunakan untuk meningkatkan *code rate*, tetapi untuk jumlah bit input yang cukup banyak *decoder* masih mampu untuk mengoreksi kesalahan pada *code rate* 2/3 dikarenakan bit error dapat terjadi pada bit *redundant*. Sedangkan untuk *error-control*, digunakan *convolution codes* maka dari itu *punctured codes* digunakan dalam *Rate-Compatible Punctured Convolution* (RCPC) yang ketika dalam kondisi kanal terburuk maka menggunakan *code rate* ½ yang artinya membentuk *convolution codes*, sedangkan dalam kondisi kanal yang baik, maka akan menggunakan *code rate* yang lebih tinggi dengan menggunakan *punctured codes*.

ISBN: 978-602-512-810-3

4. Kesimpulan

a. *Punctured codes* berhasil dibuat dengan *code rate* 2/3 yang dibangun dari *code rate* ½ menggunakan *puncture matrix* $\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}$. Dengan keberhasilan ini maka kecepatan dalam pengkodean (*code rate*) bit input menjadi lebih cepat dibandingkan dengan *convolutional codes* konvensional.

b. *Puncture matrix* yang digunakan yaitu $\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}$ dengan *generator polynomial* 6 & 7 dikarenakan respon *decoder* terhadap *puncture matrix* lainnya tidak menghasilkan solusi yang tunggal, yaitu solusi yang tidak dapat dijustifikasi sebagai hasil akhir.

c. *Decoder* mampu mengoreksi *codewords* yang memiliki *error bit* 1 dan 2 untuk jumlah bit input 35 pada *code rate* 2/3 karena *bit error* ada kemungkinan terjadi pada *bit redundant*.

Daftar Pustaka

[1] DTC NetConnect, "GANGGUAN PADA SISTEM TRANSMISI SINYAL DATA," DTC NetConnect, [Online]. Available: http://www.dtcnetconnect.com/AMP/index.php/blogs/305-gangguan-pada-sistem-transmisi-sinyal-data. [Accessed 2 August 2017].

[2] Wikipedia, "Error detection and correction," Wikimedia Foundation, Inc., 17 June 2017. [Online]. Available: https://en.wikipedia.org/wiki/Error_detection_and_correction. [Accessed 4 July 2017]

[3] G. Nagarajan, "Error Control Using RCPC Codes," 2007. [Online]. Available: http://shodhganga.inflibnet.ac.in/bitstream/10603/1261/14/14_cahpter%204.pdf. [Accessed 15 07 2017]

[4] M. A. Jabbar, "Rancang Bangun Rangkaian Convolutional Encoder dan Viterbi Decoder," Universitas Indonesia, Depok, 2008

[5] B. A. Kuncoro, "Analisis Kinerja Convolutional Coding dengan Viterbi Decoding pada," Insitut Teknologi Bandung, Bandung, 2008.

[6] N. N. Indaryanto, Implementasi Enkoder dan Dekoder Kode Konvolusi menggunakan TMS 320VC33-150, Bandung: Politeknik Negeri Bandung, 2007.

[7] Wikipedia, "Forward error correction," Wikimedia Foundation, Inc., 27 June 2017. [Online]. Available: https://en.wikipedia.org/wiki/Forward_error_correction. [Accessed 5 July 2017]

[8] Wikipedia, "Convolutional code," Wikimedia Foundation, Inc., 20 February 2017. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_code. [Accessed 5 July 2017].

[9] C. Langton, *Coding and decoding with Convolutional Codes*, www.complextoreal.com, 1999, p. July.

ISBN: 978-602-512-810-3