DESAIN ARSITEKTUR DAN IMPLEMENTASI PENGKODE-PENDEKODE *HARD DECISION* LDPC MENGGUNAKAN ALGORITMA *MESSAGE PASSING* PADA FPGA

Architectural Design and Implementation of Hard decision LDPC Encoding-Decoding with Message -Passing Algorithm on FPGA

Aditia Tarigan[1], Rita Purnamasari, ST.,MT.[2], Efa Maydhona Saputra, ST.,MT.[3]

[1,2,3] Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1] aditia.tarigan,id@ieee.org , [2]ritapurnamasari@telkomuniversity.ac.id , [3] maydhona@telkomuniversity.ac.id

Abstrak

Low Density *Parity* Check atau LDPC merupakan teknik *channel coding* yang dikembangkan oleh Robert G. Gallager pada tahun 1962. LDPC dikenal akan kemampuannya sebagai teknik pengkode yang terdekat untuk mencapai maksimum dari *Shannon capacity* dan kompleksitas rendah pada implementasi. Penelitian ini bertujuan untuk merancang arsitektur dan mengimplementasikan teknik pengkode dan pendekode LDPC pada FPGA. Metode koreksi *error* yang digunakan pada implementasi adalah algoritma *message passing*. Proses implementasi menggunakan dua jenis *code rate* yaitu *code rate* ½ dan *code rate* ¾. Pada *code rate* ½, matriks yang digunakan adalah matriks 4x8, matriks 8x16, dan matriks 24x48. Pada *code rate* ¾, matriks yang digunakan adalah matriks 4x16. Dari hasil percobaan menunjukkan bahwa algoritma *message passing* dapat melakukan koreksi *error* lebih dari satu *bit error*. Hasil implementasi sistem menunjukkan bahwa frekuensi kerja matriks 4x8 adalah 1,35 MHz, frekuensi kerja matriks 8x16 adalah 0,909 MHz, frekuensi kerja matriks 24x48 adalah 0,156 MHz, dan frekuensi kerja matriks 4x16 adalah 1,35 MHz.

Kata kunci : ***code rate, FPGA, LDPC, message passing.***

Abstract

Low Density Parity Check or LDPC is one of channel coding technique which was developed by Robert G. Gallager in 1962. LDPC is known for its capability as the nearest channel coding technique for reaching maximum of Shannon Capacity and low complexity for implementation. This research purposes were designing architectural of LDPC code system and implementing LDPC code system on FPGA board. The error correction which was used for this project was message passing algorithm. Two types of code rates were used for implementation. The code rates were ½ and ¾. Code rate ½ used matrix 4x8, matrix 8x16, and matrix 24x48. Code rate ¾ used matrix 4x16.The results showed that message passing algorithm could do error correction process for more than one bit error. The results of implementation were the frequency of matrix 4x8 systems was 1,35 MHz, the frequency of matrix 8x16 systems was 0,909 MHz, the frequency of matrix 24x48 systems was 0,156 MHz, and the frequency of matrix 4x16 systems was 1,35 MHz.

Keywords ***: code rate, FPGA , LDPC, message passing.***

1. Pendahuluan

Pada Tahun 1962, R.G. Gallager menemukan teknik pengkode yang disebut *Low Density Parity Check Codes* (LDPC)[2]. Menurut Gallager, teknik pengkode kanal pada umumnya (probabilitas *error decoding* dapat dibuat eksponensial mendekati nol dengan menambahkan panjang kode) tidak memperhitungkan kompleksitas waktu komputasi dan perangkat yang dibutuhkan[3]. LDPC merupakan teknik pengkode kanal yang dapat memfasilitas blok kode yang panjang, khususnya untuk mencapai low *error* probability, dan tidak membutuhkan waktu komputasi dan alat yang berlebihan [3].

Penelitian [8] mengusulkan untuk LDPC pada DVB-S2 teknik *encoding* menggunakan Lower Triangular dan teknik *decoding* menggunakan *bit flipping*. Menggunakan constraint (12,6), ukuran matriks cek paritas 6 x 12, dan *code rate* ½ pada FPGA (Field Programmable Gate Array), penelitian [8] membutuhkan *resource* slices 0%, slices flip-flop 0%, *resource* LUT 4 *input* 0%, *resource* bonded IOB 3%, *resource* GCLKs 3% . *Clock* yang dibutuhkan dari penelitian ini adalah 28 *clock* (280 ns). *Bit* rate yang dihasilkan adalah 21,4 Mbps. Pada proses decode penelitian [8], algoritma *bit flipping* menggunakan proses iterasi dalam proses *forward error correction*.

Algoritma *bit flipping* melakukan pengecekan per satu *bit* atau iterasi dilakukan sebanyak n *bit* yang ada. Dari penelitian [8] diperoleh juga bahwa *bit flipping* forward *error* correction hanya bisa dilakukan untuk maksimal *error* 1 *bit*.

Dalam penelitian ini penulis mencoba untuk merancang dan mengimplementasikan aristektur LDPC menggunakan algoritma *message passing*. Aristektur LDPC yang dirancang mencakup blok *encoder* dan blok *decoder*. Implementasi sistem akan dilakukan pada FPGA Cyclon II.

Pada penelitian ini penulis melakukan studi literatur dengan mencari, mengumpulkan, dan memahami baik berupa jurnal, artikel, buku referensi, internet, dan sumber-sumber lain yang berhubungan dengan masalah penelitian. Adapun tujuan dari penelitian ini melakukan pengujian dan pengimplementasian LDPC pada dua tipe *code rate*. *Code rate* yang pertama adalah ½ . *Code rate* ini menggunakan matriks LDPC 4x8, matriks LDPC 8x16, matriks LDPC 24x48. *Code rate* yang kedua adalah ¾. *Code rate* ini menggunakan matriks LDPC 4x16. Hasil penelitian ini diharapkan dapat terancang dan terimplementasinya LDPC dengan algortima *message passing*.

2. LDPC dan FPGA

LPDC merupakan suatu teknik *channel coding* yang dikembangkan oleh R.G.Gallager [2]. Teknik ini sangat cocok untuk digunakan pada *high data rate* dan *high channel noise*.

Field Programmable Gate Arrays adalah sebuah integrated circuits yang membuat seorang perancang dapat merancang suatu program logika digital ke dalam suatu wadah. Inti dari sebuah FPGA adalah selayaknya rangkaian integrated circuits pada umumnya.

2.1 Low Density *Parity* Check

Low Density *Parity* Check (LDPC) adalah sebuah kelas dari blok kode *linear* [4]. LDPC ditemukan pada tahun 1962 oleh R.G.Gallager[1,4]. LDPC dinyatakan oleh Gallager dapat meningkatkan kemampuan komputasi untuk *codeword* yang besar untuk memenuhi nilai *Shannon capacity* [2].

Nama LDPC berasal dari karakteristik check *parity matrix*-nya. Pada LDPC matriks *parity* mengandung edikit *bit* 1 dibandingkan dengan jumlah *bit* 0 pada matriks tersebut [2,4,8]. Keuntungan utama dari metode ini adalah penyajian performansi yang mendekati kapasitas dari banyak kanal yang berbeda dan linear time complex algorithms untuk *decoding*. [2,4]

Pada LDPC suatu matriks dianggap regular jika bobot pada kolom (wc) dan pada baris seragam(wr)[4]. Sehingga, pada regular matriks berlaku mwr=nwc.

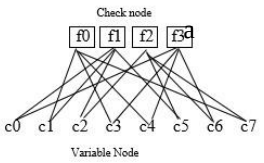
Code rate (R) pada LDPC didefinisikan sebagai berikut :

$$R = \frac{k}{n} = \frac{n-m}{n} = \frac{n-(n\frac{w_c}{w_r})}{n} = 1 - \frac{m}{n} \qquad (1)$$

2.2 LDPC Representation

Pada LDPC wr digunakan untuk mendeskripsikan jumlah 1 pada setiap baris dan wc pada kolom[3,4]. Suatu matriks yang disebut low density jika memenuhi kondisi wc << n (banyak baris) dan wr << m (banyak kolom)[4]. Contoh representasi matriks diilustrasikan pada gambar dibawah ini.

$$H = \begin{bmatrix} 0\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1\,0\,1\,0 \end{bmatrix}$$



b.

Gambar 1. a. Matriks Representasi LDPC
b. Representasi *Tanner Graph* LDPC [4]

Tanner Graph merupakan bipartite graph. *Bipartite graph* artinya dua *node* pada *graph* dipisahkan menjadi dua set khas dan ujung – ujungnya hanya menghubungkan dua jenis *graph* yang berbeda.

Pada LDPC terdapat dua tipe *node* pada *Tanner Graph* yaitu *variable node* (v-*nodes*) dan *check node* (c-*nodes*). *Variable nodes* (c0,c1,c2,c3..,cn) merupakan representasi dari *codeword*. Pada gambar 1 *check node* (fi) terhubung dengan *variable node* cj jika element hij pada matriks H = 1.

2.3 *Hard decision Decoding* LDPC

Metode *Hard-Decision decoding* menggunakan algoritma *message passing* [4] terdiri dari beberapa tahap. Pada tahap pertama v-*nodes* ci mengirim pesan ke c-*nodes* fj yang mana *bit* yang dikirim dianggap *bit* yang benar. Pada tahap ini informasi hanya ada a di v-*nodes* ci, yang mana bersesuain dengan *bit* ke-i dari c, yi. Sebagai contoh,pada gambar.1, c0 mengirim pesan mengandung 1 ke f1 dan f3, *node* c1 mengirim pesan mengandung y1 (1) ke f0 dan f1, dan selanjutnya. Pada tahap kedua, setiap cek *nodes* fj menghitung respon fj -> ci setiap terhubung dengan *variable node*. Pesan respon mengandung *bit* yang mana fj beranggapan bahwa *bit* ini yang benar untuk v-*nodes* ci (diasumsikan v-*nodes* yang lain terhubung ke fj benar). Pada tahap ini v-*nodes* menerima pesan dari cek ci -> fj *nodes* dan menggunakan informasi ini untuk menentukan apakah *bit* asalnya OK. Langkah paling mudah adalah dengan menggunakan majority vote. Kemudian langkah selanjutnya validasi atau kembali ke langkah 2.

2.4 *Message passing Encoding*

Prosedur pengkode *message passing* dapat dijelaskan dengan perulangan yang mana semua *variable nodes* dan semua *check nodes* secara paralel dilakukan proses iterasi melewatkan pesan sepanjang sisi yang saling berdekatan.

Nilai dari setiap *bit* kode terus diperbaharui. Algoritma ini terus berulang hingga seluruh pengikisan terpenuhi, atau sampai jumlah iterasi atau perulangan tertentu.

Proses *encoding* sudah dijelaskan oleh B.M. Leiner [4].

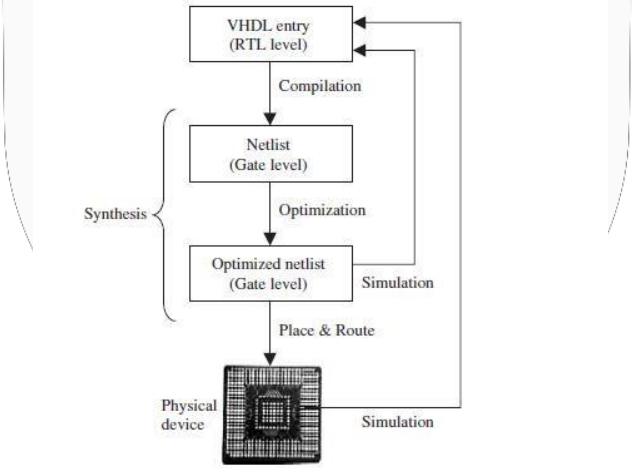
2.5 VHDL

VHDL merupakan bahasa pendeskripsian hardware. VHDL mendeskripsikan suatu perilaku atau behavior suatu perangkat elektronik agar nantinya sistem dari perangkat tersebut dapat diimplementasikan[7].

VHDL merupakan singkatan dari *VHSIC Hardware Description Language*. VHSIC merupakan singkatan dari *Very High Speed Integrated Circuit*, yang merupakan suatu proyek Kementrian Pertahanan Amerika Serikat pada tahun 1980-an [7].

Penggunaan VHDL sebagai bahasa pemrograman cukup umum dilakukan. VHDL dapat diimplementasikan pada perangkat-perangkat yang bersifat programmable (FPGA, PLD,dsb) dari Xilinx, Altera, Atmel,dsb.

Dalam pengimplementasian VHDL ke dalam ada beberapa skema yang dilakukan ditunjukkan oleh ilustrasi berikut ini :



Gambar 2. Proses Implementasi pada FPGA [5]

Ada tiga struktur pada suatu bahasa VHDL, yaitu :

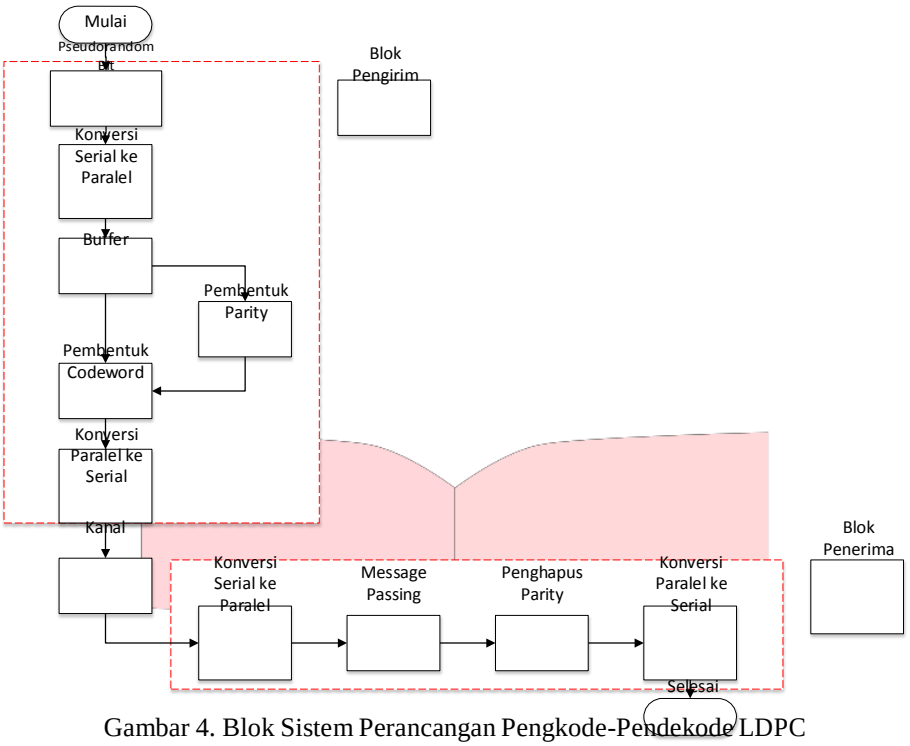
1. *Library* : berisikan seluruh *library* yang digunakan untuk men-*design* suatu sistem.
2. *Entity* : suatu kumpulan spesifikasi dari pin *input* dan ataupun pin *output*.
3. *Architecture* : bagian yang mendeskripsikan bagaimana suatu sirkuit berfungsi.

2.6 FPGA

FPGA atau *Field Programmable Gate Array* merupakan suatu rangkaian *integrated circuit.* FPGA memampukan seorang perancang perangkat *digital* untuk merancang desainnya dan mengimplementasikannya dengan menggunakan bahasa VHDL di FPGA.

3. Arsitektur dan Implementasi Blok Sistem LDPC

Berikut adalah blok diagram sistem LDPC yang akan dirancang.



Gambar 4. Blok Sistem Perancangan Pengkode-Pendekode LDPC

Blok sistem terbagi menjadi tiga blok besar yaitu blok pengirim, blok kanal, dan blok penerima. Blok pengerim terdiri dari dua fungsi yaitu menghasilkan sinyal informasi, dan melakukan pengkodean atau pembentukan *codeword*. Blok penerima terdiri dari dua fungsi utama yaitu melakukan *forward error correction* dan mengembalikan informasi ke bentuk semula (sinyal tanpa *codeword*). Blok kanal berfungsi untuk memberikan nilai salah pada *codeword* keluaran pengkode.

3.1 *Design* Sistem Pada FPGA

Clock FPGA yang akan digunakan pada sistem ini adalah *clock* 50 Mhz. *Clok* ini akan didistribusikan ke seluruh blok sistem dengan pembagian dengan cara serial. Untuk memudahkan dalam pengamatan, *clock* akan direpresentasikan secara bilangan *integer* oleh sinyal *counter*. Nilai *counter* akan bertambah setiap kali *clock* mengalami *rising edge*.

Reset pada sistem ini akan menyebabkan *counter* ke nilai 0 dan keluaran dari seluruh sistema adalah "00...". Kondisi *reset* sistem dapat disebabkan oleh dua hal. Yang pertama adalah dengan mengaktifkan *switch button* 1 pada FPGA. Yang kedua adalah ketika *counter* mencapai batasan maksimum yang ditentukan.

Blok kontrol utama merupakan blok yang mengatur seluruh kerja sistem. Blok ini memiliki *counter* yang berfungsi menjadi kontrol dari kinerja seluruh sistem. Blok kontrol utama memiliki dua buah *input* yang menjadi bagian utama dari seluruhan sistem juga yaitu *clock* dan *reset*.

Blok *pseudorandom bit* merupakan blok yang berfungsi menjadi blok pembangkit sumber informasi pada seluruh sistem. Blok ini membangkitkan sinyal acak yang berfungsi sebagai informasi bersifat *random*. Blok ini akan mulai bekerja ketika *enable* 1 aktif.

Ada dua jenis blok serial to parallel pada penelitian ini, yaitu blok serial to parallel 1 dan blok serial to parallel 2.

Blok serial to parallel 1 adalah blok yang berfungsi mengubah data serial keluaran dari blok pseudorandom bit menjadi bentuk paralel. Tujuan dari pengubahan sinyal dari bentuk serial ke bentuk paralel adalah untuk memudahkan pemudahan pemrosesan pada blok pembentuk parity. Blok serial to parallel 2 adalah blok yang mengubah nilai keluaran kanal yang berbentuk serial menjadi bentuk paralel. Tujuan dari blok ini adalah mempermudah pemrosesan pada blok message passing. Berikut adalah pemrosesan counter blok serial to parallel 2 saat blok aktif.

Pengkode terdiri dari tiga buah blok yaitu blok buffer, blok parity generator, dan blok codeword generator. Blok buffer akan menerima masukan dari blok serial to parallel 1 dan menahannya hingga enable 3 tidak aktif. Blok parity generator bertugas untuk menghasilkan nilai *parity* dari *codeword*. Setelah *parity* terbentuk, blok codeword generator akan aktif dan menggabungkan *parity* dan informasi untuk membentuk *codeword*.

Terdapat dua buah blok parallel to serial pada sistem yang dirancang di penelitian ini, yaitu blok parallel to serial 1 dan blok parallel to serial 2. Blok ini secara keseluruhan berfungsi untuk mengubah sinyal dari bentuk paralel menjadi bentuk serial. Blok parallel to serial 1 bekerja merubah sinyal dari bentuk paralel keluaran keluaran

pengkode. Tujuan mengubah bentuk sinyal menjadi serial adalah untuk melakukan penghematan terhadap *resource pin* pada FPGA. Blok parallel to serial 2 bekerja untuk merubah sinyal keluaran pendekode yang paralel menjadi bentuk informasi semula yaitu serial.

Kanal berfungsi untuk membuat kesalahan pada codeword keluaran pengkode. Kanal pada sistem ini terdiri dari dua blok yaitu blok error generator dan blok kanal. Blok kanal berisikan bit-bit yang akan membuat nilai codeword menjadi salah. Nilai dari bit-bit ini sudah ditentukan dari awal sistem dibentuk. Blok kanal menerima dua masukan yaitu dari blok parallel to serial 1 dan blok error generator. Kedua masukan akan mengalami operasi modulo-2 pada blok ini.

Blok pendekode terdiri dari dua blok yaitu blok message passing dan blok penghilang parity. Blok message passing memiliki fungsi yaitu menentukan informasi yang diterima oleh blok ini merupakan informasi yang benar atau tidak. Selain menentukan informasi, blok ini juga diharapkan dapat melakukan koreksi error atau forward error correction (FEC) Blok penghapus parity bertugas untuk menghapus bit parity dan memisahkannya dari nilai informasi yang sesungguhnya. Blok ini berada satu kesatuan dengan blok serial to parallel 2.

3.2 Hasil Pengujian

Pada bagian ini akan dipaparkan hasil pengujian dari blok sistem LDPC yang telah dirancang.

a. Pengujian Clock Blok Pengkode Terhadap Perubahan Ukuran Matriks

Pengujian pengaruh terkadap perubahan ukuran matriks dilakukan pada matriks yang memiliki code rate yang sama. Dalam hal ini adalah matriks 4x8, matriks 8x16, matriks 24x48. Dari Hasil implementasi di dapatkan sebagai berikut :

Tabel. 1 Perbandingan Proses Pengkodean Pada Ukuran Matriks Berbeda

Matriks	*Clock* Yang Terpakai	Jumlah *Clock* Maksimum yang dapat digunakan
4x8	13	13
8x16	18	23
24x48	20	266

b. Pengujian Clock Blok Pengkode Terhadap Perubahan Code Rate

Dari hasil pengujian blok pengkode, semakin tinggi *code rate* yang digunakan akan semakin cepat proses pengkodean yang digunakan. Matriks 4x16 dan matriks 8x16 menghasilkan 16 *bit* informasi pada keluaran *codewordnya*. Tetapi, matriks 4x16 membutuhkan waktu proses yang lebih sedikit.

Tabel. 2 Perbandingan Proses Pengkodean Pada *Code Rate* Berbeda

Matriks	Code Rate	Jumlah *Clock* Yang Digunakan
4x16	3/4	13
8x16	1/2	23

c. Hasil Pengujian Kemampuan Koreksi *Error*

Proses koreksi *error* pada matriks 4x8 menunjukkan kemampuan *error* koreksi maksimal satu buah *bit error*. Jika ukuran matriks diperbesar menjadi 8x16, kemampuan maksimum untuk melakukan koreksi *error* pun menjadi meningkat. Pada matriks 8x16, kemampuan koreksi *error* ini menjadi dua buah *bit*. Jumlah maksimum kemampuan koreksi *error* pada matriks 24x48 adalah tiga *bit error*. Peningkatan *code rate* memberikan hasil yang tidak baik. Kemampuan koreksi *error* matriks 4x16 dengan *code rate* ¾ hanya mampu melakukan maksimal koreksi *error* satu buah *bit*. *Bit-bit* yang mampu dikoreksi oleh *decoder* matriks ini hanyalah *bit* ke-1 dan *bit* ke-16. Kemampuan koreksi *error* ini jauh sangat berbeda yang dapat dengan matriks 8x16 yang mampu mengkoreksi *error* hingga dua *bit*.

d. Hasil Pengujian Penggunaan *Resource* FPGA

Hasil implementasi pengkode dan pendekode LDPC pada FPGA mempengaruhi pemakaian *resources* dari FPGA sendiri. *Resources* yang terpakai antara lain adalah *total logic elements, total combinational functions, dedicated logic register, total pins,* dan *total memory bits*. Berikut adalah hasil implementasi dari pengkode-pendekode LDPC di FPGA Cyclone II.

Perbandingan penggunaan *resources* pada tabel 3 menunjukkan bahwa pada *code rate* yang sama semakin besar ukuran matriks maka semakin besar *resources* FPGA yang digunakan.

Pada matriks 8x16 dan matriks 4x16 memiliki jumlah yang sama pada penggunaan *resources* FPGA. Hal ini menunjukkan bahwa *resource* tidak terlalu berpengaruh pada perbedaan *code rate*, tetapi jumlah bit yang diproses. Kedua matriks melakukan pemrosesan 16 *bit*.

Tabel. 3 Perbandingan Pemakaian *Resources* FPGA

Matriks	*Code Rate*	*Resources*				
		total logic elements	total combinational functions	dedicated logic register	total pins	total memory bits
4x8	1/2	19%	15%	11%	14%	4%
8x16	1/2	28%	22%	17%	11%	7%
24x48	1/2	72%	57%	31%	11%	16%
4x16	3/4	28%	22%	17%	11%	7%

e. Hasil Pengujian Frekuensi Kerja Sistem

Clock yang digunakan pada implementasi ini adalah *clock* internal FPGA dengan frekuensi kerja 50 MHz. Frekuensi 50 MHz sama dengan $\frac{1}{50 \times 10^6} = 2 \times 10^{-8} = 20$ ns. Dengan kata lain, satu *clock* pada FPGA setara dengan 20 ns.

Tabel. 4 Perbandingan *Clock*, Perioda, dan Frekuensi Kerja Sistem

Matriks	*Code Rate*	Kebutuhan *Clock* Sistem	Perioda Kerja Sistem	Frekuensi Kerja Sistem $F = \frac{1}{T}$
4x8	1/2	37	740 ns	1,35 MHz
8x16	1/2	55	1100 ns	0,909 MHz
24x48	1/2	319	6380 ns	0,156 MHz
4x16	3/4	37	740 ns	1,35 MHz

Hasil implementasi kerja sistem menunjukkan frekuensi kerja sistem pengkode-pendekode LDPC lebih kecil dari internal *clock* FPGA yang digunakan. Sehingga, sistem dapat diimplementasikan.

4. Kesimpulan

Kesimpulan yang didapatkan dari penelitian ini adalah sebagai berikut. Jika ukuran matriks semakin besar pada *code rate* yang sama, maka proses *encoder* akan membutuhkan *clock* yang semakin banyak. Pada jumlah *codeword* yang sama dan *code rate* yang lebih tinggi membutuhkan *clock* yang lebih sedikit. Blok pendekode dapat melakukan lebih dari satu *bit* koreksi *error*. Pada jumlah *codeword* yang sama dengan *code rate* yang lebih tinggi, performansi koreksi *error* blok *decoder* mengalami penurunan. Frekuensi kerja matriks 4x8 adalah 1,35 MHz. Frekuensi kerja matriks 8x16 adalah 0,909 MHz. Frekuensi kerja matriks 24x48 adalah 0,156 MHz. Frekuensi kerja matriks 4x16 adalah 1,35 MHz. Pemanfaatan *resource*s FPGA tidak bergantung dari *code rate* yang digunakan, tetapi bergantung dari jumlah *bit* yang diproses.

Daftar Pustaka:

[1] *Cyclone II Device Handbook, Volume 1* . 2008. San Jose: www.Altera.com.

[2] GALLAGER, R. G. 1962. *Low-Density Parity-Check Codes. IRE TRANSACTIONS ON INFORMATION THEORY, 21-28*.

[3] Haykin, S. 2000. *Communication Systems 4th Edition*. New York: John Wiley & Sons, Inc

[4] Leiner, B. M. 2005. *LDPC Codes – a brief Tutorial*.

[5] MEYLANI, L. 2010. *DIKTAT KULIAH SISTEM KOMUNIKASI II*. Bandung.

[6] Moreira, J. C., & Farrell, P. G. 2006. *ESSENTIALS OF ERROR-CONTROL CODING*. West Sussex, England: John Wiley & Sons Ltd.

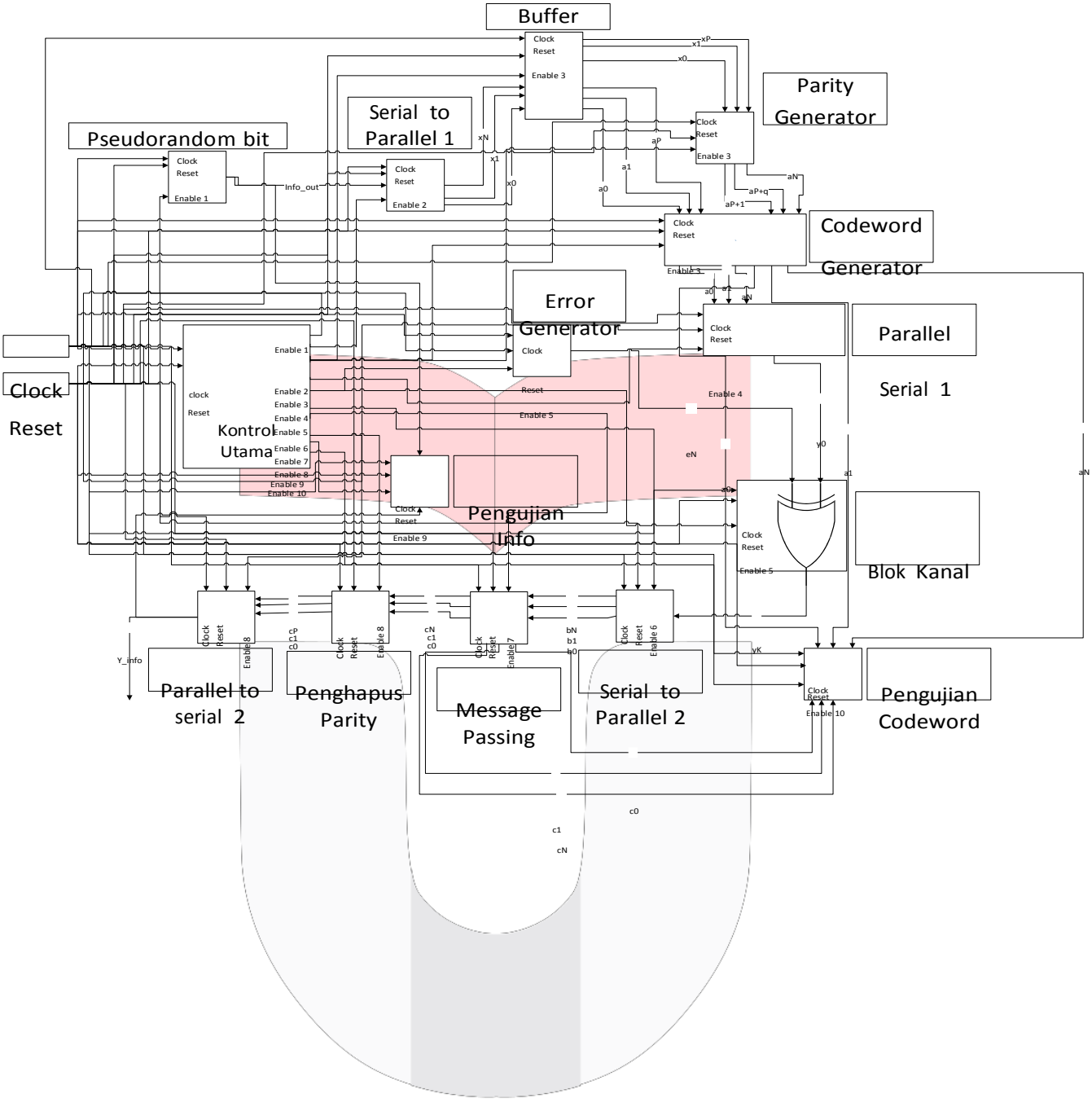
[7] Pedroni, V. A. 2004. *Circuit Design With VHDL*. Cambridge, Massachusetts: MIT Press.

[8] Purnamasari, R., Wijanto, H., & Hidayat, I. 2014. *Design and implementation of LDPC (Low Density Parity Check) coding technique on FPGA (Field Programmable Gate Array) for DVB-S2 (Digital Video Broadcasting-Satellite). Aerospace Electronics and Remote Sensing Technology (ICARES), 2014 IEEE International Conference on (hal. 83-88).* Yogyakarta: IEEE.

[9] Quartus II Introduction. 2010. ww.Altera.com.

[10] Siegel, Paul H. *An Introduction to Low-Density Parity-Check Codes*. University of California, San Diego

Lampiran A
Arsitektur Blok Sistem

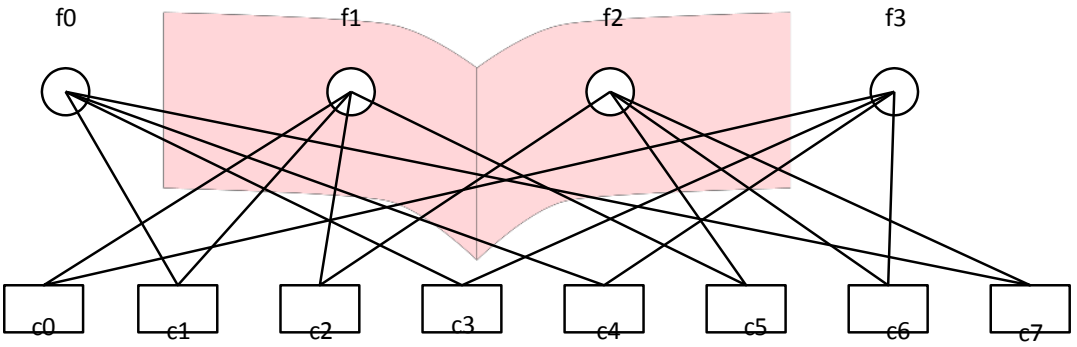


Lampiran B
Matriks LDPC dan Representasi *Tanner Graph*

Matriks 4x8 *code rate* ½ [4]

$$H = \begin{bmatrix} 0\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1\,0\,1\,0 \end{bmatrix}$$

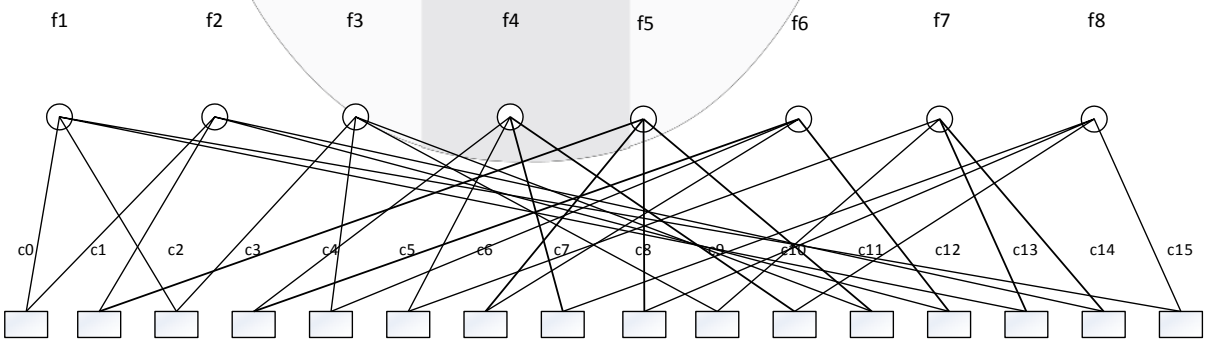
Tanner Graph Matriks 4x8[4]



Matriks 8x16 *code rate* ½ ($w_r = 2$, $w_c = 4$)

$$H = \begin{bmatrix} 1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1 \\ 1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1 \end{bmatrix}$$

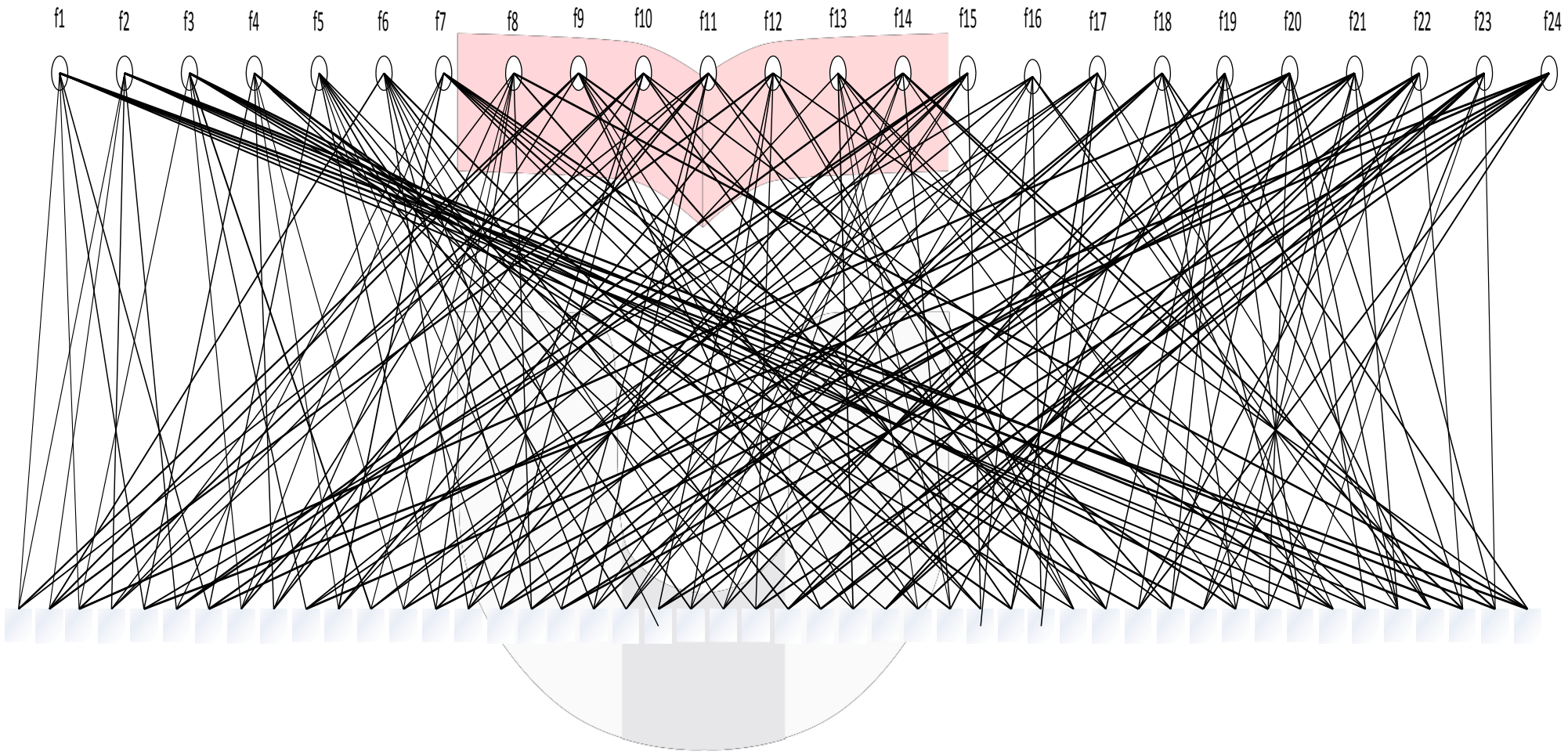
Tanner Graph Matriks 8x16



Matriks 24x48 *code rate* ½

$$H = \begin{bmatrix}
1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\\
1\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\\
0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\\
1\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\\
0\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\\
1\,0\,0\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,0\\
0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\\
0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0
\end{bmatrix}$$

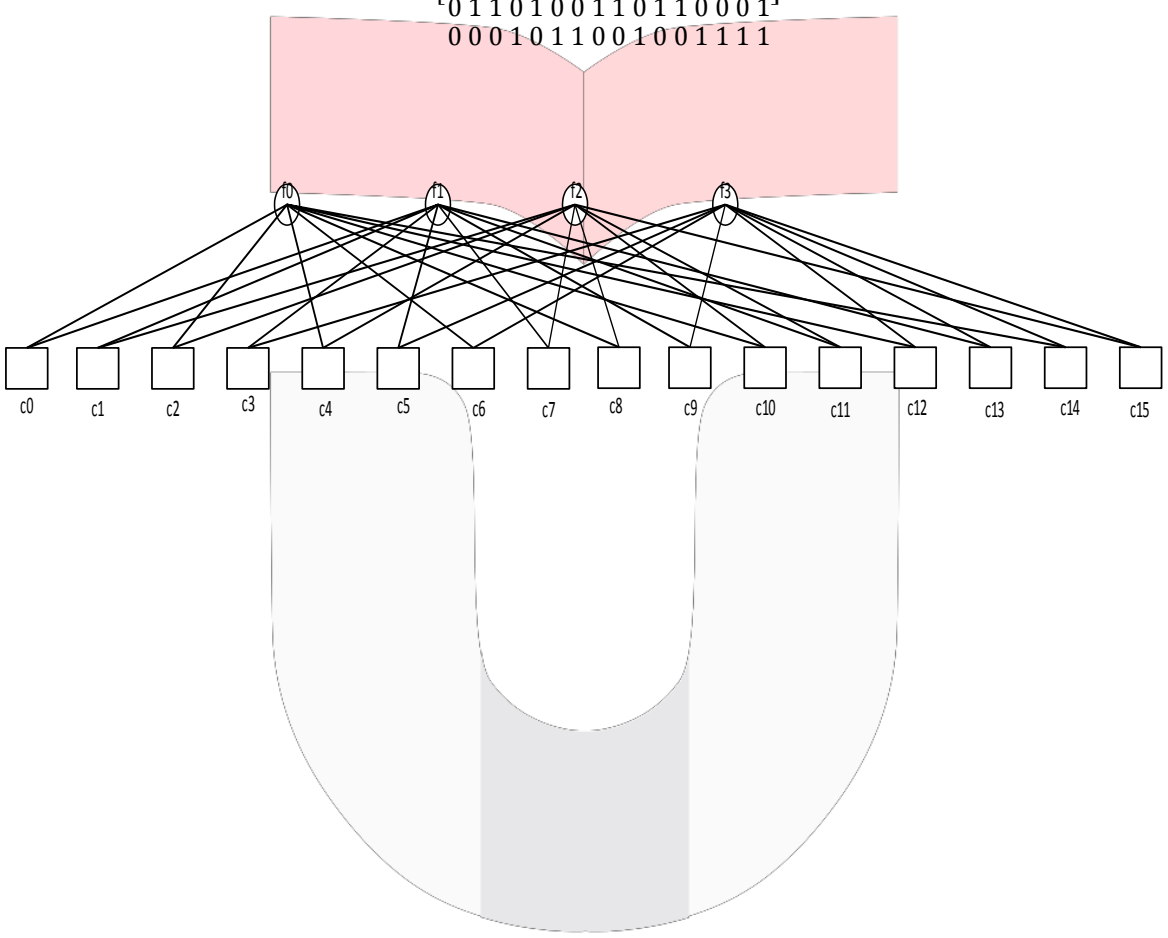
Tanner Graph Matriks 24x48



Matriks 4x16 *code rate* ¾ ($w_r = 2$, $w_c = 8$)

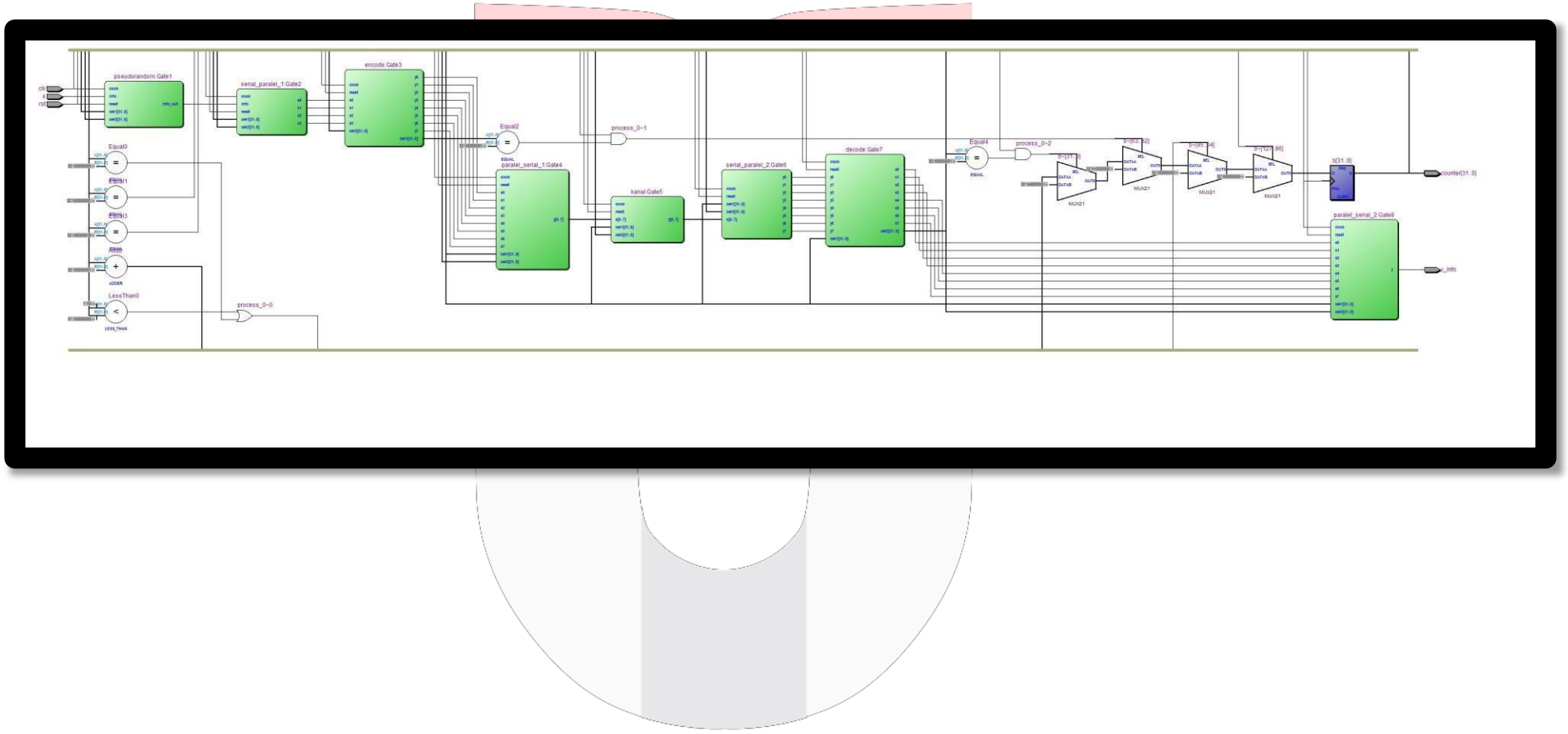
$$H = \begin{bmatrix} 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0 \\ 1\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0 \\ 0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,1\,1 \end{bmatrix}$$

Tanner Graph Matriks 4x16

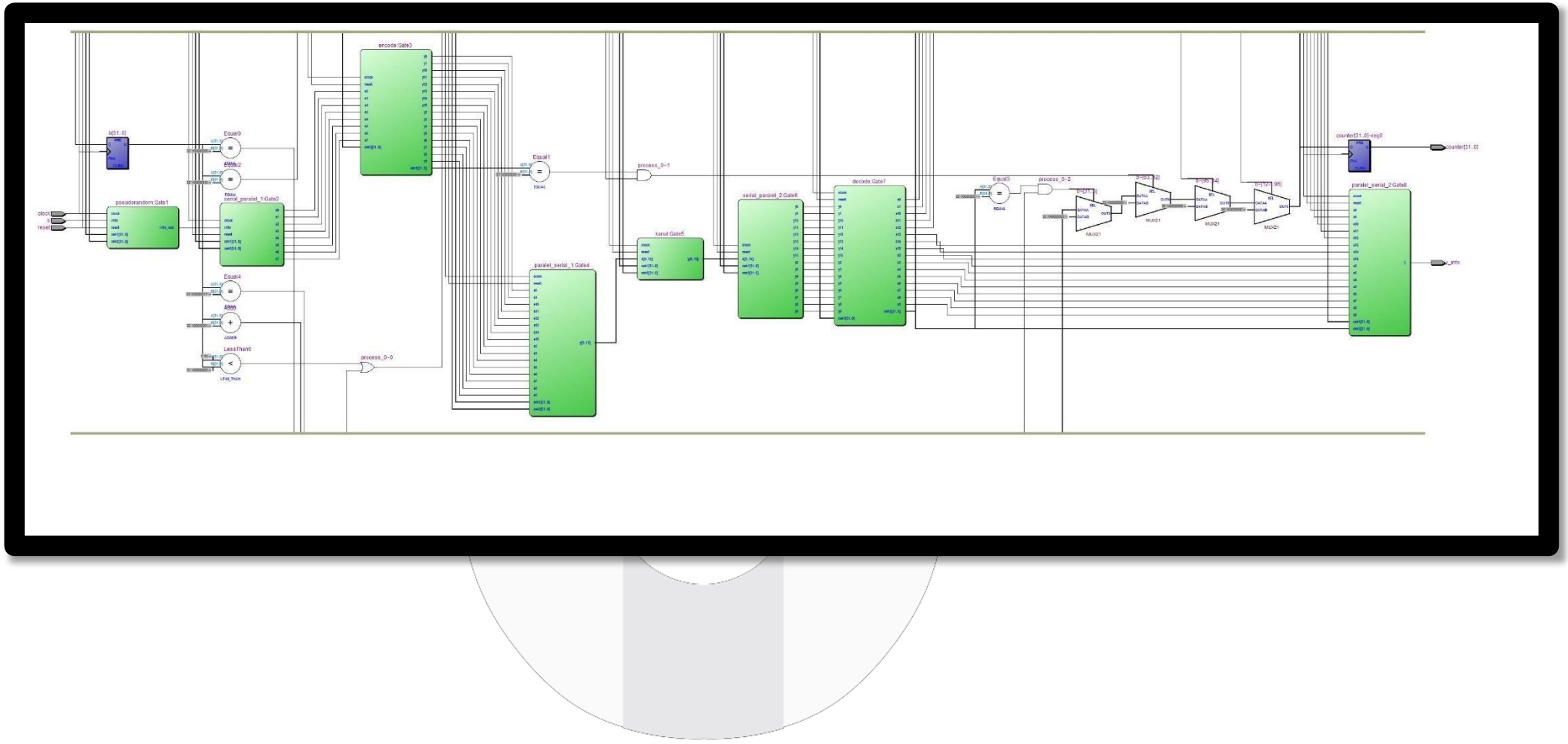


Lampiran C
Hasil Implementasi Arsitektur Pada RTL Viewer Sistem Pengkode-Pendekode LDPC

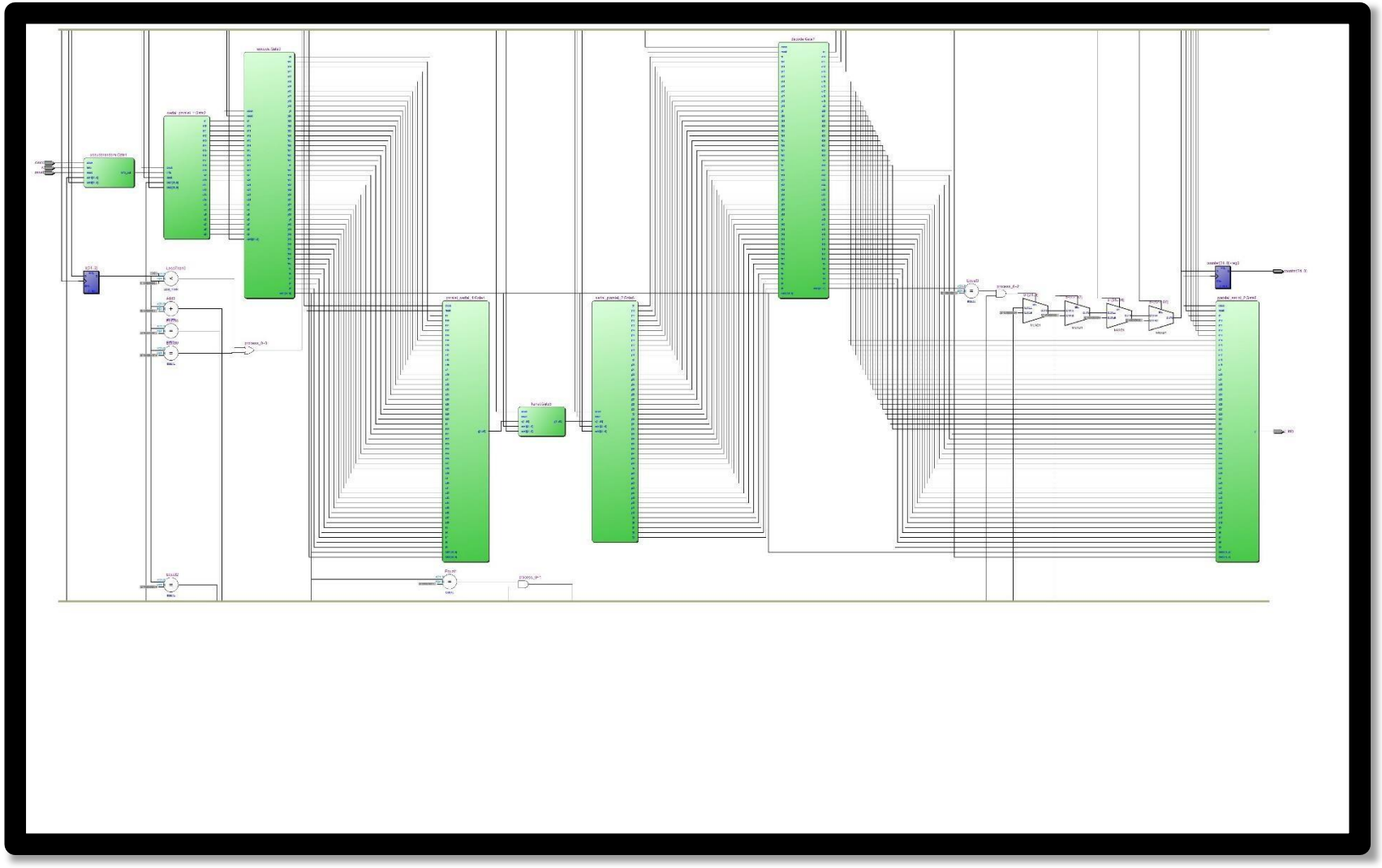
Matriks 4x8



Matriks 8x16



Matriks 24x48



Matriks 4x16