Penggunaan Aljabar Boolean dalam Perancangan Rangkaian Full Adder

Fransiskus Febryan Suryawan - 13519124[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13519124@std.stei.itb.ac.id

***Abstrak*—Komputer adalah perangkat elektronik yang saat ini dapat ditemui dimana saja. Salah satu komponen penting dalam komputer adalah Central Processing Unit atau CPU, yang mampu untuk melakukan operasi aritmatika secara cepat. Bagian dalam CPU yang bertugas untuk melakukan operasi-operasi tersebut adalah ALU. Operasi aritmatika dasar yang diimplementasikan dalam ALU salah satunya adalah penjumlahan, yang menjadi basis untuk rangkaian pengurang serta pengali. Penjumlahan dalam ALU dilakukan oleh rangkaian penjumlah atau *adder*, yang mampu menjumlahkan dua bilangan dan mengeluarkan satu bilangan hasil. Agar CPU dapat bekerja dengan optimal, tentu komponen penjumlah harus dapat menjumlahkan dua buah bilangan dengan kecepatan optimal juga. Karena komponen penjumlah pada dasarnya adalah sekumpulan gerbang logika, maka komponen penjumlah dapat disederhanakan dengan hukum-hukum Aljabar Boolean dan peta Karnaugh, sehingga gerbang logika yang digunakan dapat dibuat sesedikit mungkin, yang kemudian menghasilkan komponen penjumlah yang lebih optimal.**

***Kata Kunci*—Aljabar Boolean, *full adder*, rangkaian, Peta Karnaugh.**

I. Pendahuluan

Saat ini, perangkat elektronik telah muncul dalam berbagai aspek kehidupan. Salah satu perangkat yang dapat terlihat dimana-mana adalah *smartphone*. Komponen utama yang digunakan agar perangkat tersebut bekerja adalah *Central Processing Unit* atau CPU, yang melakukan operasi-operasi dalam kecepatan sangat tinggi sehingga dapat menampilkan beragam hal kepada penggunanya. Dalam CPU, terdapat satu komponen yang bernama *Arithmetic Logic Unit* (ALU), yang bertugas melakukan operasi aritmatika serta logika pada data yang diberikan.

Komponen ALU memiliki tugas yang penting, yaitu untuk melakukan operasi dengan cepat agar data dapat ditampilkan secepat mungkin kepada pengguna. Operasi yang dapat dilakukan dalam ALU, lebih spesifiknya dalam *Arithmetic Unit*, mencakup penjumlahan dua buah bilangan. Operasi penjumlahan dua buah bilangan dalam ALU dilakukan dengan menggunakan rangkaian khusus, yang disebut dengan rangkaian penjumlah atau *adder*. Rangkaian ini menerima masukan berupa bilangan yang akan dijumlahkan, dan akan mengeluarkan hasil penjumlahannya. Komponen adder ini merupakan basis dari komponen-komponen aritmatika lainnya, termasuk pengurang dan pengali, serta Floating-Point Unit(FPU) yang mengoperasikan bilangan dalam representasi *floating point*.

Karena ALU merupakan komponen pembangun dalam banyak komponen komputer lainnya, maka penting untuk memahami bagaimana ALU bekerja. Kinerja ALU dapat ditingkatkan dengan cara mengurangi penggunaan daya, yang salah satunya dapat dilakukan dengan mengurangi jumlah gerbang logika yang digunakan didalamnya. Selain itu, karena gerbang logika membutuhkan waktu untuk melakukan operasinya, maka pengurangan gerbang logika diinginkan untuk mencapai suatu rangkaian yang efisien.

Selain itu, pemahaman akan komponen penjumlah merupakan tambahan menarik untuk pemain permainan komputer yang memberikan mekanik untuk membangun rangkaian, seperti dalam *Minecraft*, *Terraria*, *No Man's Sky*, dan lainnya. Karena kebanyakan rangkaian dalam komputer membutuhkan komponen penjumlah, maka pengetahuan untuk membangun rangkaian penjumlah yang optimal akan sangat berguna.

II. Landasan Teori

A. Aljabar Boolean

Aljabar Boolean adalah cabang dari aljabar yang pertama kali dikenalkan oleh matematikawan bernama George Boole dalam bukunya yang berjudul *The Laws of Thought*. Aljabar Boolean yang banyak digunakan adalah Aljabar Boolean dua nilai, yaitu Aljabar Boolean yang memiliki dua nilai yang mungkin untuk nilai variabelnya, yang biasa dilambangkan dengan 0 dan 1. Nilai 0 melambangkan nilai terkecil, sedangkan nilai 1 melambangkan nilai terbesar. Karena sistem digital menggunakan biner untuk melakukan beragam operasi, Aljabar Boolean dapat digunakan untuk membantu merancang rangkaian yang digunakan.

Ada tiga buah operasi yang harus didefinisikan dalam Aljabar Boolean, dua operator biner dan satu operator uner. Operator biner yang didefinisikan adalah operator AND dan

operator OR. Operator AND diwakilkan oleh simbol "·" dan operator OR yang diwakilkan oleh simbol "+". Operator AND akan mengeluarkan 1 ketika kedua masukan yang diterima bernilai 1, seperti dapat dilihat pada tabel kebenaran pada Tabel I.

Tabel I. Tabel Kebenaran Operator AND

AND	0	1
0	0	0
1	0	1

Operator AND biasa disebut juga sebagai operator perkalian dalam Aljabar Boolean. Operator OR akan mengeluarkan 1 ketika setidaknya satu dari masukan yang diterima bernilai 1. Tabel kebenaran untuk Operator OR dapat dilihat pada Tabel II.

Tabel II. Tabel Kebenaran Operator OR

OR	0	1
0	0	1
1	1	1

Operator OR biasa disebut juga sebagai operator penjumlahan dalam Aljabar Boolean. Operator uner yang didefinisikan dalam Aljabar Boolean adalah operator NOT yang diwakilkan oleh simbol "". Operator NOT akan mengeluarkan nilai yang berkebalikan dengan masukannya. Jika masukannya adalah 1, maka keluarannya 0. Begitu pula untuk masukan 0, keluarannya haruslah 1. Tabel kebenaran untuk operator NOT dapat dilihat pada Tabel III.

Tabel III. Tabel Kebenaran Operator NOT

NOT	
0	1
1	0

Di dalam aljabar Boolean juga berlaku hukum-hukum berikut:

1. Identitas
$$a + 0 = a$$
$$a \cdot 1 = a$$
2. Idempoten
$$a + a = a$$
$$a \cdot a = a$$
3. Komplemen
$$a + a' = 1$$
$$a \cdot a' = 0$$
4. Dominansi
$$a \cdot 0 = 0$$
$$a + 1 = 1$$
5. Involusi
$$(a')' = a$$
6. Penyerapan
$$a + a \cdot b = a$$
$$a \cdot (a + b) = a$$
7. Komutatif
$$a + b = b + a$$
$$a \cdot b = b \cdot a$$
8. Asosiatif
$$a + (b + c) = (a + b) + c$$
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$
9. Distributif
$$a + (b \cdot c) = (a + b) \cdot (a + c)$$
$$a \cdot (b + c) = a \cdot b + a \cdot c$$
10. De Morgan
$$(a + b)' = a' \cdot b'$$
$$(a \cdot b)' = a' + b'$$
11. 0/1
$$0' = 1$$
$$1' = 0$$

Fungsi ekspresi Boolean dalam Aljabar Boolean dapat diekspresikan dalam dua bentuk yang berbeda, yaitu *sum-of-product(SOP)* dan *product-of-sum*(POS). Dalam bentuk SOP, ekspresi dinyaatakan sebagai hasil penjumlahan dari *minterm*, yaitu perkalian beberapa nilai. Sedangkan bentuk POS menyatakan ekspresi sebagai hasil kali dari beberapa *maxterm*, yaitu penjumlahan beberapa nilai. Bentuk kanonik dari ekspresi Boolean adalah ekspresi yang mengandung semua variabel, baik dalam bentuk awal maupun bentuk komplemen.

B. Peta Karnaugh (K-Map)

Peta Karnaugh adalah metode untuk menyederhanakan ekspresi dalam Aljabar Boolean, pertama kali diperkenalkan oleh Maurice Karnaugh pada tahun 1953. Metode ini memetakan ekspresi Aljabar Boolean dalam bentuk standar SOP menjadi bentuk tabular, dimana setiap sel dalam tabel merepresentasikan kombinasi masukan yang mungkin, dan isi dari sel tersebut merupakan keluarannya. Sel-sel dalam Peta Karnaugh diurutkan berdasarkan *Gray Code*, yaitu dua posisi yang bersebelahan hanya berbeda dalam 1 bit. Gray code berbeda dengan cara bilangan ditulis dalam biner pada umumnya, karena *Gray Code* memastikan hanya satu bit yang berubah antar dua bilangan yang bersebelahan, sedangkan dalam penulisan biner biasa, seringkali terdapat lompatan jumlah bit yang berubah. Tabel IV berisi urutan *Gray Code* dari urutan 0 hingga 15.

Tabel IV. Gray Code

Indeks	**4**	**3**	**2**	**1**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0

5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Penyederhanaan ekspresi Boolean menggunakan Peta Karnaugh dapat dilakukan dengan mengelompokkan sel-sel yang ada dalam kelompok *pair*(dua elemen bersebelahan), *quad*(empat elemen bertetangga), maupun *octet*(delapan elemen bertetangga). Kelompok juga dapat berisi satu elemen, tetapi kelompok yang ideal adalah yang memiliki anggota paling banyak. Elemen yang dikelompokkan dapat berupa nilai keluaran 1 atau kondisi *don't care*, dimana keluaran tidak dipedulikan. Pengelompokkan elemen-elemen ini dapat dilakukan dengan melihat tabel sebagai sebuah toroida yang tergulung atas-bawah dan kiri-kanannya.

Hasil penyederhanaan menggunakan Peta Karnaugh dapat menghasilkan ekspresi dalam bentuk POS, yang mengelompokkan semua 1, atau dalam bentuk SOP, yang mengelompokkan semua 0. Dalam Peta Karnaugh, *minterm* ataupun *maxterm* dapat dicari dengan cara melihat variabel mana yang tidak berubah dalam satu kelompok. Jika variabel mempertahankan nilai 0, maka operasi komplemen harus diaplikasikan terlebih dahulu pada variabel sebelum dimasukkan sebagai bagian dari *minterm* ataupun *maxterm.*

C. Gerbang Logika

Gerbang logika adalah metode untuk merepresentasikan ekspresi Boolean. Setiap gerbang logika yang berbeda merepresentasikan operasi Boolean yang berbeda. Terdapat tiga macam gerbang logika dasar, yaitu gerbang AND, OR dan NOT, yang masing-masing merepresentasikan operator AND, OR dan NOT pada aljabar Boolean. Gerbang AND direpresentasikan oleh simbol pada gambar 1, untuk gerbang OR direpresentasikan pada gambar 2, dan gerbang NOT direpresentasikan pada gambar 3.

Gambar 1. Gerbang AND

Gambar 2. Gerbang OR

Gambar 3. Gerbang NOT

Selain tiga gerbang dasar yang telah disebutkan, terdapat juga gerbang turunan yang didapat dengan menggabungkan beberapa gerbang dasar. Gerbang-gerbang logika turunan yaitu gerbang NAND, gerbang NOR, gerbang XOR, dan gerbang XNOR. Gerbang NAND merepresentasikan operator $(xy)'$ dan direpresentasikan oleh simbol pada gambar 4.

Gambar 4. Gerbang NAND

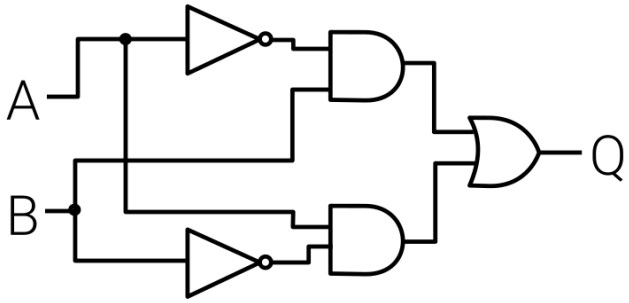
Gerbang NAND dapat dibangun dengan menghubungkan gerbang NOT pada keluaran gerbang AND. Gerbang NOR merepresentasikan operator $(x + y)'$ dan direpresentasikan oleh simbol pada gambar 5.

Gambar 5. Gerbang NOR

Gerbang NOR dapat dibangun dengan cara yang serupa dengan gerbang NAND, yaitu dengan memasangkan gerbang NOT pada keluaran gerbang OR. Kemudian, ada gerbang XOR, operasinya dinotasikan dengan $\oplus$, yang menghasilkan nilai 1 hanya ketika salah satu dari kedua masukannya bernilai 1 dan yang lainnya bernilai 0. Gerbang XOR dilambangkan dengan simbol pada gambar 6.

Gambar 6. Gerbang XOR

Gerbang XOR dapat dibentuk dari gerbang logika dasar dengan menggabungkan gerbang AND, OR, dan NOT seperti pada gambar 7.



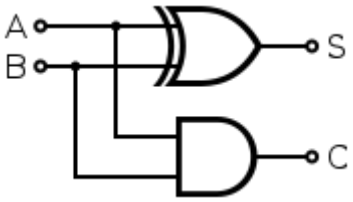
Gambar 7. Gerbang XOR dengan gerbang dasar

Sedangkan gerbang XNOR merupakan turunan dari XOR, merepresentasikan operator $(x \oplus y)'$. Gerbang XNOR dilambangkan dengan simbol pada gambar 8.

Gambar 8. Gerbang XNOR

D. Half Adder

Half adder adalah sebutan untuk rangkaian logika yang dapat menjumlahkan dua buah bilangan tanpa melihat bit bawaan dari penjumlahan digit lain. *Half adder* menjumlahkan dua bit dan mengeluarkan hasilnya, serta satu bit *carry* untuk merepresentasikan bit lebih dari hasil penjumlahan. Rangkaian sederhana untuk *half adder* dapat dilihat pada gambar 9.



Gambar 9. Rangkaian Half Adder

Kelemahan dari *half adder* adalah rangkaian ini hanya dapat menjumlahkan dua bit, karena tidak ada masukan untuk bit hasil penjumlahan berlebih. Namun, kedua bit keluaran dari *half adder* dapat merepresentasikan nilai 3, walaupun nilai terbesar dari masukan adalah 1_2 dan 1_2 untuk A dan B, yang akan menghasilkan nilai 2 bila dijumlahkan. Rangkaian *full adder* dibuat untuk mengatasi kelemahan dari rangkaian *half adder* yang telah disebutkan.

III. PERANCANGAN RANGKAIAN FULL ADDER

Rangkaian *full adder* adalah rangkaian yang dapat menjumlahkan tiga bit, yaitu dua bit bilangan dan satu bit untuk *carry* dari penjumlahan lain. Rangkaian *full adder* memiliki kemiripan dengan rangkaian *half adder*. Kedua rangkaian mengeluarkan dua bit, satu untuk jumlah dan satu untuk *carry*, yaitu hasil penjumlahan yang berlebih. Rangkaian *full adder* memiliki kelebihan yaitu ketiga input bit dapat menghasilkan keluaran jumlah 1_2 dan *carry* 1_2.

Untuk rangkaian *full adder* 1 bit, Ketiga bit ditambahkan untuk menghasilkan bit jumlah dan *carry*. Dengan demikian, rangkaian *full adder* 1 bit digabungkan sedemikian sehingga rangkaian mampu untuk menambahkan lebih banyak bit sekaligus.

Perancangan rangkaian *full adder* harus dimulai dengan membuat tabel kebenarannya terlebih dahulu. Tabel kebenaran untuk *full adder* 1 bit dapat dilihat pada Tabel V.

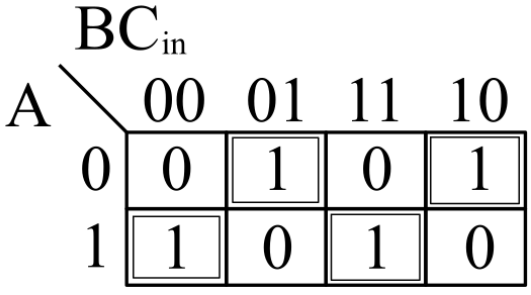
Tabel V. Tabel Kebenaran Full Adder

A	**B**	**C_{in}**	**S**	**C_{out}**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Dari tabel kebenaran, dapat dibuat ekspresi Boolean untuk setiap keluaran, yaitu S(Sum, jumlah) dan C_{out}(*carry*). Ekspresi Boolean untuk S dalam bentuk kanonik SOP yaitu

$$S = \Sigma(m_1, m_2, m_4, m_7)$$
$$C_{out} = \Sigma(m_3, m_5, m_6, m_7)$$

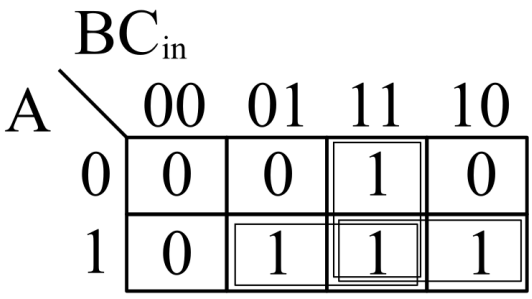
Namun, apabila rangkaian *full adder* langsung dibuat dari fungsi kanonik tersebut, pastilah akan memakan banyak gerbang logika. Maka dari itu, K-Map digunakan untuk menyederhanakan ekspresi Boolean. Gambar 10 menunjukkan peta Karnaugh untuk ekspresi S. Dalam peta Karnaugh yang terbentuk, tidak ada kelompok yang berukuran lebih besar dari satu.



Gambar 10. Peta Karnaugh Ekspresi S

Peta Karnaugh untuk ekspresi C_{out} ditunjukkan pada gambar 11. Dalam peta Karnaugh yang terbentuk, kelompok terbesar

yang terbentuk berukuran 2.



Gambar 11. Peta Karnaugh Ekspresi C_{out}

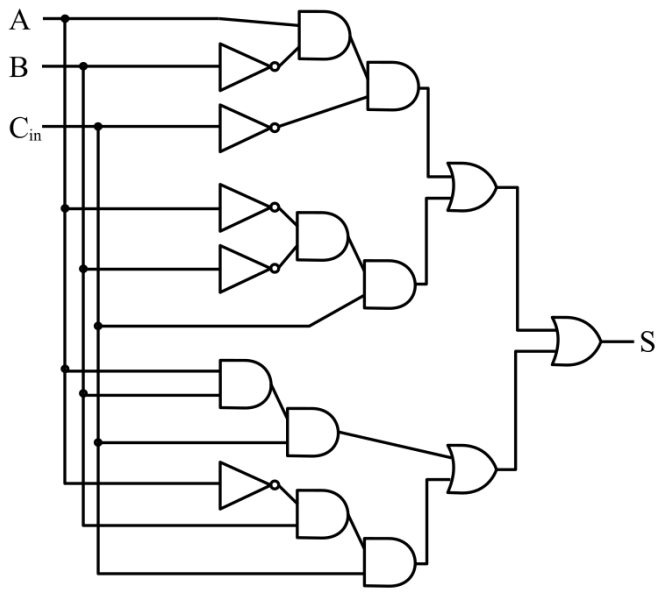
Dari kedua Peta Karnaugh dapat dibentuk ekspresi Boolean yang lebih sederhana untuk S dan C_{out}. Ekspresi yang terbentuk untuk S dapat ditulis sebagai

$$S = A \cdot B' \cdot C'_{in} + A' \cdot B' \cdot C_{in} + A \cdot B \cdot C_{in} + A' \cdot B \cdot C_{in}$$

Sedangkan untuk ekspresi C_{out}, ada beberapa kelompok berukuran dua, yang memungkinkan ekspresinya untuk ditulis ulang sebagai

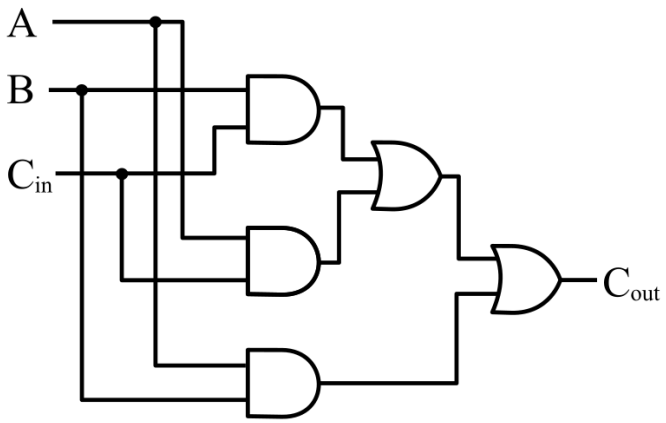
$$C_{out} = B \cdot C_{in} + A \cdot C_{in} + A \cdot B$$

Setelah mendapat ekspresi Boolean sederhana untuk kedua keluaran. Untuk keluaran S, ekspresi sederhananya sama dengan ekspresi awal, karena tidak ada kelompok dengan ukuran lebih besar dari 1. Rangkaian yang terbentuk dari ekspresi tersebut dapat dilihat pada gambar 12.



Gambar 12. Rangkaian untuk S

Sedangkan untuk ekspresi sederhana C_{out} membentuk rangkaian seperti pada gambar 13. Rangkaian yang terbentuk pada gambar 13 adalah rangkaian paling sederhana menurut peta Karnaugh, yaitu rangkaian yang membutuhkan sesedikit mungkin gerbang logika dasar.



Gambar 13. Rangkaian untuk C_{out}

Rangkaian sederhana yang dibentuk adalah rangkaian yang hanya menggunakan gerbang logika dasar yaitu AND, OR, dan NOT. Untuk rangkaian ekspresi S, tidak didapatkan rangkaian yang lebih sederhana hanya dengan menggunakan gerbang logika dasar. Namun, dengan gerbang logika turunan, rangkaian tersebut dapat dibuat lebih sederhana. Sedangkan untuk ekspresi C_{out}, rangkaian yang dihasilkan sudah cukup sederhana.

V. KESIMPULAN

Aljabar Boolean dapat membantu dalam perancangan rangkaian logika. Dengan menggunakan metode peta Karnaugh, rangkaian sederhana dapat dicari, sehingga mengurangi jumlah gerbang logika yang digunakan dalam rangkaian. Langkah-langkah penyederhanaan dapat diaplikasikan pada perancangan rangkaian *full adder* 1 bit. Rangkaian tersebut dapat disederhanakan lebih jauh dengan gerbang logika turunan, seperti pada rangkaian *half adder.*

Selain pada rangkaian *full adder*, teknik penyederhanaan dengan peta Karnaugh juga dapat digunakan dalam merancang rangkaian-rangkaian logika yang lain. Karena peta Karnaugh dapat diaplikasikan pada aljabar Boolean, teknik ini juga dapat menyederhanakan kondisi logika pada berbagai situasi. Namun penggunaan peta Karnaugh dalam perancangan rangkaian logika sangat besar, karena seringkali gerbang logika yang dapat digunakan terbatas jumlahnya, atau diinginkan rangkaian logika yang efisien.

VI. UCAPAN TERIMAKASIH

Pertama-tama penulis mengucapkan syukur kepada Tuhan Yang Maha Esa, karena tanpa rahmat dari-Nya penulis tidak akan mampu untuk menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga mengucapkan

terima kasih sebesar-besarnya kepada dosen pengampu mata kuliah Matematika Diskrit IF2120, Dr. Nur Ulfa Maulidevi, M.T., yang telah menuntun perjalanan penulis selama mempelajari matematika diskrit, dan kepada Dr. Ir. Rinaldi Munir, M.T. yang telah menyediakan materi pembelajaran secara daring. Penulis pun mengucapkan terima kasih kepada orang tua, saudara, serta sahabat yang senantiasa memberi motivasi kepada penulis.

REFERENCES

[1] S. Givant and P. Halmos, *Introduction to Boolean Algebras*. Undergraduate Texts in Mathematics, Springer.
[2] M. Karnaugh, "The Map Method for Synthesis of Combinational Logic Circuits.
[3] F. M. Brown, *Boolean Reasoning – The Logic of Boolean Equations*. Mineola, New York: Dover Publications, Inc.
[4] https://en.wikibooks.org/wiki/Digital_Circuits/Adders, diakses pada 09 Desember 23.00 WIB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2020

Fransiskus Febryan Suryawan
1351924