Pengembangan Sistem Square Root Digital dengan Algoritma Newton-Raphson untuk Pengolahan Data dari ADC 16-bit

Kelompok 01

Anggota:

1. **13224001 – Yozia Gedalya Marcho Ginting**
2. **13224002 – Jeva Steve Sinaga**
3. **13224003 – Benedictus Kenneth Setiadi**

EL 2002 Sistem Digital

2025

Daftar Isi

Pendahuluan

Berbagai operasi matematis selalu digunakan untuk menyelesaikan permasalahan sehari-hari. Operasi-operasi dasar yang sering ditemui adalah penjumlahan, pengurangan, perkalian, dan pembagian. Namun, ada operasi lain yang memiliki cakupan penggunaan yang luas, salah satunya adalah akar kuadrat. Akar kuadrat dapat kita temui dalam perhitungan teorema Phytagoras untuk menghitung jarak, perhitungan RMS (*Root Mean Square*), penentuan magnitudo bilangan kompleks dalam aplikasi kelistrikan, serta rumus-rumus lain yang berkaitan dalam bidang Matematika dan Fisika.

Dalam dunia digital, operasi akar kuadrat cenderung sulit untuk diimplementasikan dengan rangkaian-rangkaian logika dasar seperti *adder, subtractor, left shifter*, dan *right shifter*. Penggunaan LUT (*look up table*) juga tidak ideal untuk sistem yang skalabel karena membutuhkan memori dalam jumlah yang signifikan. Oleh karena itu, dibutuhkan sebuah metode untuk menghitung akar kuadrat yang cepat, akurat, dan efisien dalam penggunaan sumber daya.

Salah satu metode untuk mengaproksimasi akar kuadrat adalah algoritma Newton-Raphson. Algoritma tersebut memanfaatkan operasi-operasi dasar seperti penjumlahan, perkalian, dan pembagian yang diulang berkali-kali (lihat persamaan 1). Algoritma NR cocok diimplementasikan di FPGA karena memiliki tingkat konvergensi yang tinggi (*quadratic convergence*) dan hanya memerlukan operasi dasar yang mudah direalisasikan.

Oleh karena itu, proyek ini bertujuan untuk mengimplementasikan operasi perhitungan akar kuadrat dengan algoritma Newton-Raphson yang cepat, akurat, hemat sumber daya, dan cocok untuk sistem digital berbasis FPGA.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Persamaan 1. Bentuk umum rumus iteratif Newton-Raphson

Deskripsi Sistem

Proyek ini akan diimplementasikan untuk mendukung perhitungan akar kuadrat dengan masukan data dari sebuah komponen ADC (*Analog to Digital Converter*) dengan lebar data sebesar 16-bit bertipe *integer*. Proses aproksimasi hasil akar kuadrat akan menggunakan algoritma Newton-Raphson (NR) sebagai kerangka utamanya. Algoritma NR membutuhkan tebakan awal yang baik untuk memperkecil jumlah iterasi sehingga akan dibentuk sebuah LUT (*Lookup Table*) dengan 512 entri untuk menunjang hal tersebut. Setiap entri akan diisi dengan rerata geometrik dari batas bawah dan atas entri. Setelah proses komputasi menggunakan algoritma Newton-Raphson, data keluaran akan memiliki format Q8.8 atau 8-bit integer untuk memperluas pilihan data keluaran yang dikehendaki pengguna.

Arsitektur dari proses komputasi akan mengedepankan efisiensi sumber daya dan kecepatan komputasi tanpa mengurangi tingkat akurasi dari hasil akhir. Hal ini dicapai dengan manipulasi matematis dan pemilihan algoritma yang efisien. Berikut adalah penjabaran rinci mengenai deskripsi sistem.

Input dan *Output*

Berdasarkan konteks penggunaan sistem akar kuadrat di proposal proyek ini, sistem akan memiliki beberapa input seperti yang dijabarkan pada tabel di bawah ini.

Tabel 1. Tabel perincian masukan sistem beserta bentuk dan fungsinya

Input	Bentuk	Fungsi
clk	std_logic	Clock sistem untuk desain sinkron.
reset	std_logic	Active-low pengaturan ulang register secara sinkron.
start	std_logic	Pulsa siklus tunggal untuk memulai komputasi akar kuadrat.
x_in	std_logic_vector (15 downto 0)	Data masukan sebesar 16-bit bertipe *unsigned integer* dari ADC sensor.

Adapun, sistem juga akan memiliki beberapa keluaran yang dapat menunjang kestabilan sistem dan kemudahan pembacaan hasil seperti penjabaran tabel di bawah ini.

Tabel 2. Tabel perincian keluaran sistem beserta bentuk dan fungsinya

Output	Bentuk	Fungsi
sqrt_q88	std_logic_vector (15 downto 0)	Keluaran hasil komputasi dengan format Q8.8.
sqrt_int	std_logic_vector (7 downto 0)	Keluaran hasil komputasi yang hanya memuat 8-bit awal untuk direpresentasikan sebagai integer.
busy	std_logic	Bernilai high ketika sistem sedang dalam proses komputasi dan bernilai low ketika sistem dalam keadaan idle.
done	std_logic	Keluaran yang menandakan jika hasil valid dan aman untuk dibaca.
error	std_logic	Keluaran yang menandakan jika hasil tidak valid atau terdapat error internal.
uart_tx	std_logic	Keluaran data valid untuk ditampilkan di

		komputer dengan protokol komunikasi UART.

Dasar Perhitungan Matematis

Rumus yang dicantumkan pada pendahuluan (persamaan 1) merupakan bentuk umum dari algoritma aproksimasi Newton-Raphson. Untuk menunjang perhitungan akar kuadrat, diperlukan rumus dengan bentuk khusus.

Asumsikan kita ingin menghitung $\sqrt{A}$, dengan $A \in \mathbb{R}$ dan $A \geq 0$. Ambil

$$f(x) = x^2 - A = 0$$

Persamaan 2. Fungsi untuk akar kuadrat

Substitusikan $f(x)$ ke bentuk umum sehingga diperoleh

$$x_{n+1} = x_n - \frac{x_n^2 - A}{2x_n}$$

Persamaan 3. Bentuk klasik rumus iteratif Newton-Raphson untuk akar kuadrat

Secara matematis, kita dapat menggunakan persamaan 3 untuk menjadi kerangka komputasi utama. Akan tetapi, kita dapat menyederhanakan bentuk tersebut seperti di persamaan 4.

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{A}{x_n}\right)$$

Persamaan 4. Bentuk sederhana rumus iteratif Newton-Raphson untuk akar kuadrat

Persamaan 4 lebih efisien dalam hal sumber daya komputasi perangkat keras karena hanya membutuhkan satu pembagi, satu penjumlah, dan satu *right-shifter*. Melalui ini kita dapat menghilangkan ketergantungan dalam membuat modul penghitung kuadrat dan operasi pengurangan yang berulang.

Pemilihan bentuk rumus yang lebih sederhana belum cukup untuk mengoptimasi algoritma NR karena ada permasalahan lain yang mempunyai dampak signifikan terhadap kecepatan komputasi yaitu pemilihan nilai tebakan awal. Hal ini dapat diselesaikan dengan mengimplementasikan sebuah LUT. Pada Cyclone IV, terdapat BRAM berupa M9K dengan total 4Mbit. Mengetahui jumlah memori yang melimpah, akan dipakai satu blok M9K untuk menampung LUT tersebut. Oleh karena itu, LUT akan memuat 512 entri dan setiap entri akan mempunyai cakupan indeks selebar 128 angka.

Implementasi LUT untuk tebakan nilai awal dapat membantu mengurangi jumlah iterasi NR dapat dibuktikan secara matematis. Pertama, perlu didefinisikan persamaan galat untuk algoritma NR akar kuadrat, yakni:

$$e_k = x_k - \sqrt{A}$$

Persamaan 5. Galat absolut dari algoritma NR akar kuadrat

$$\varepsilon_k = \frac{e_k}{\sqrt{A}}$$

Persamaan 6. Galat relatif dari algoritma NR akar kuadrat

Setelah proses substitusi persamaan 5 ke persamaan 4 dan sedikit operasi aljabar, maka

$$e_{k+1} = \frac{e_k^2}{2\sqrt{N}}$$

Persamaan 7. Karakteristik galat absolut

$$\varepsilon_{k+1} = \frac{e_k^2}{2}$$

Persamaan 8. Karakteristik galat relatif

Karakteristik galat relatif pada persamaan 8 memiliki pola berulang seperti pada persamaan 9, yaitu

$$\varepsilon_k = \frac{\varepsilon_0^{2^k}}{2^{(2^k-1)}}$$

Persamaan 9. Pola besaran galat

Berdasarkan penentuan *output* dari sistem yakni Q8.8, target akurasi adalah kurang dari 2^{-8}. Dengan terdefinisnya tujuan akurasi dan karakteristik galat, dapat ditentukan jumlah iterasi minimal dari algoritma NR

$$\frac{\varepsilon_0^{2^k}}{2^{(2^k-1)}} < 2^{-8}$$

Persamaan 10. Target galat sistem

Atau dalam jumlah iterasi,

$$k > log_2\left(\frac{9}{(1-log_2(\varepsilon_0))}\right)$$

Persamaan 11. Target jumlah iterasi sistem

Untuk melihat perbandingan besaran LUT dan dampaknya terhadap jumlah iterasi algoritma NR, akan dijabarkan melalui tabel di bawah ini.

Tabel 3. Tabel perbandingan performa besaran LUT

Jumlah bit LUT	ε_0 maksimal	Jumlah iterasi	Jumlah Siklus	Harga Memori
0 (Tak ada)	42.7%	3-4	30-40	0 bit
4	1.56%	2	20	256 bit
6	0.39%	1-2	12-20	1024 bit
8	0.098%	1	12	4096 bit
9	0.048%	1	12	8192 bit
10	0.024%	1	12	16384 bit

Berdasarkan tabel tersebut, ukuran LUT 9-bit dipilih karena meminimalisasi jumlah iterasi dan galat awal. LUT berukuran lebih dari 9-bit tidak dipilih karena menghasilkan *diminishing returns* serta memerlukan BRAM yang lebih banyak. Akan tetapi, perlu diperhatikan bahwa nilai entri harus dalam bentuk rerata geometrik untuk mengurangi galat awal dan disimpan dalam format Q8.8.

Salah satu permasalahan dari algoritma NR adalah operasi pembagian karena membutuhkan sumber daya yang paling besar. Metode pembagian yang paling primitif adalah *restoring division*, tetapi metode ini memakan waktu yang lama. Untuk sistem ini, akan diimplementasikan metode Goldschmidt (lihat gambar 1). Metode ini dipakai karena mencapai konvergensi lebih cepat dengan mengorbankan beberapa DSP (*Digital Signal Processor*). Ini dapat terjadi karena algoritma Goldschmidt menggunakan konsep rasio dan memanfaatkan perkalian berulang. Keefektifan metode ini terjadi karena menjamin nilai konvergen selama telah dinormalisasi. Hal tersebut dapat dibuktikan dengan penjabaran matematis.

Misal kita ingin menghitung nilai dari $\frac{N}{D}$ dengan $N,\ D \in \mathbb{R}$ dan $D \in [0.5,\ 1)$. Maka,

$$F_0 = 2 - D_0 \in (1,\ 1.5]$$

Persamaan 12. Faktor pengali awal

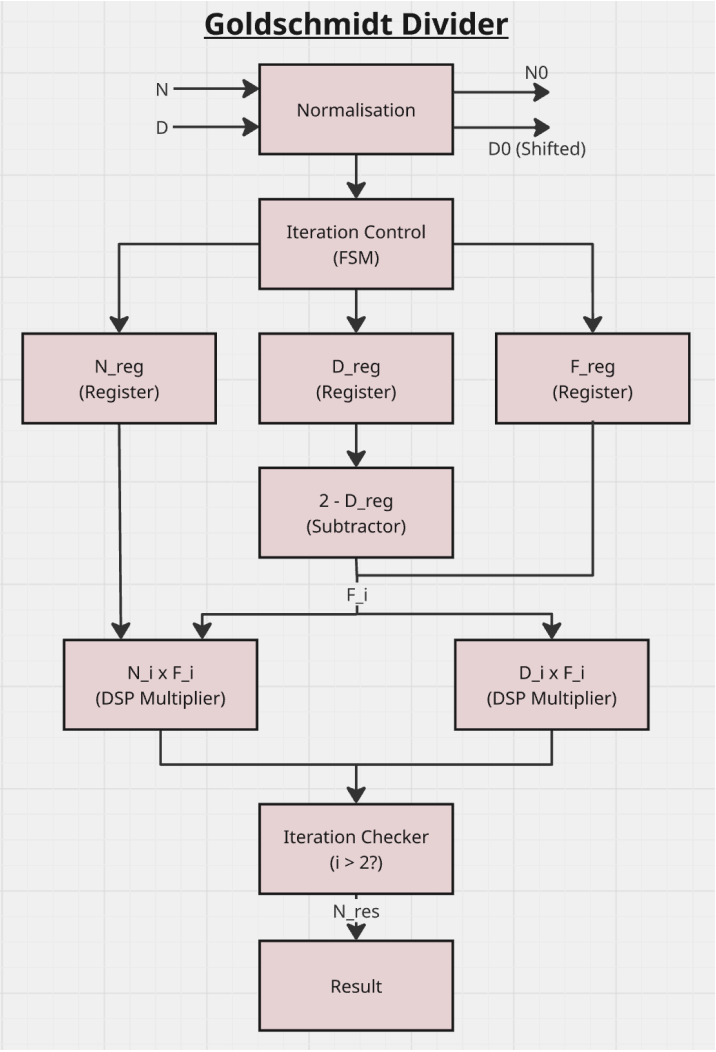
$$D_1 = D_0 \times F_0 = 2D_0 - D_0^2$$

Persamaan 13. Pembilang iterasi pertama dalam hal D_0

Karena galat adalah $\epsilon = 1 - D$, maka

$$D_1 = 1 - \epsilon^2$$

Persamaan 14. Pembilang iterasi pertama dalam hal galat



Gambar 1. Diagram alir algoritma pembagi Goldschmidt

Terakhir, format Q8.8 digunakan karena mempertimbangkan jumlah angka maksimal yang dapat dikeluarkan tanpa mengubah lebar data. Karena masukan data adalah 16-bit *unsigned integer*, maka masukan maksimal adalah 65535 atau $(2^{16} - 1)$. Hasil akar kuadrat dari angka tersebut adalah sekitar 255,99. Berdasarkan kasus ekstrim ini, diperlukan lebar data sebanyak 8-bit untuk menampung hasil bilangan bulat dari hasil akar kuadrat dan 8-bit sisanya untuk representasi angka pecahan. Jika memakai format lain, akan terdapat sumber daya yang terbuang tanpa manfaat yang signifikan.

Blok-Blok Penyusun Sistem

Setelah mendefinisikan dasar teori matematis sistem, dapat dilanjutkan dengan penjabaran blok-blok penyusun sistem. Adapun setiap blok penyusun akan dirincikan pada tabel di bawah ini.

Tabel 4. Tabel blok-blok penyusun sistem beserta fungsi-fungsinya

Blok-Blok	Fungsi
Register	Menampung variabel-variabel iterasi, penghitung iterasi, data masukan konstan. Implementasi akan menggunakan bentuk D flip-flop register.
LUT	Berfungsi untuk menyimpan nilai tebakan awal sehingga jumlah iterasi berkurang.
Divider	Pembagi akan menggunakan metode Goldschmidt untuk menghemat waktu operasi.
Adder	Menjumlahkan pembilang pada rumus NR.
Right-Shifter	Mengimplementasikan pembagi dua dengan logika penggeseran 1-bit ke kanan.
Multiplexer	Kendali alur data seperti pemilihan tebakan awal, mengatur hasil iterasi, format keluaran, dan kerangka dari *barrel shifter*.
Comparator dan Logika Error	Untuk memeriksa operasi-operasi ilegal seperti penyebut bernilai nol dan mengakhiri proses iterasi.
Counter	Menghitung jumlah iterasi dengan bentuk counter 3-bit.
FSM	Mengontrol alur sekuensial, yakni: IDLE - LOAD - DIVIDE - UPDATE - DONE/ERROR.
Output Formatter	Mengonversi hasil komputasi menjadi Q8.8 dan integer.
UART Transmitter	Mengirim hasil keluaran ke komputer dengan protokol UART.

Datapath

Adapun alur data yang akan diimplementasikan pada sistem ini dimulai saat sinyal start diberikan. Nilai masukan akan ditampung pada sebuah register dan digunakan sebagai konstanta. Setelah itu, algoritma Newton-Raphson untuk perhitungan akar kuadrat dapat dimulai dengan memilih tebakan awal sehingga register iterasi dapat diinisialisasi. Setiap iterasi data akan melakukan operasi pembagian dan hasilnya akan dijumlahkan. Hasil penjumlahan tersebut akan digeser satu bit ke kanan untuk mendapatkan hasil akhir sesuai dengan rumus pada persamaan 3. Nilai ini akan dikembalikan ke register iterasi sampai pengulangan sebanyak 4 kali telah dilakukan. Setelah proses iterasi selesai, data akan dipindahkan ke register output dan sinyal done akan aktif jika tidak terjadi error.

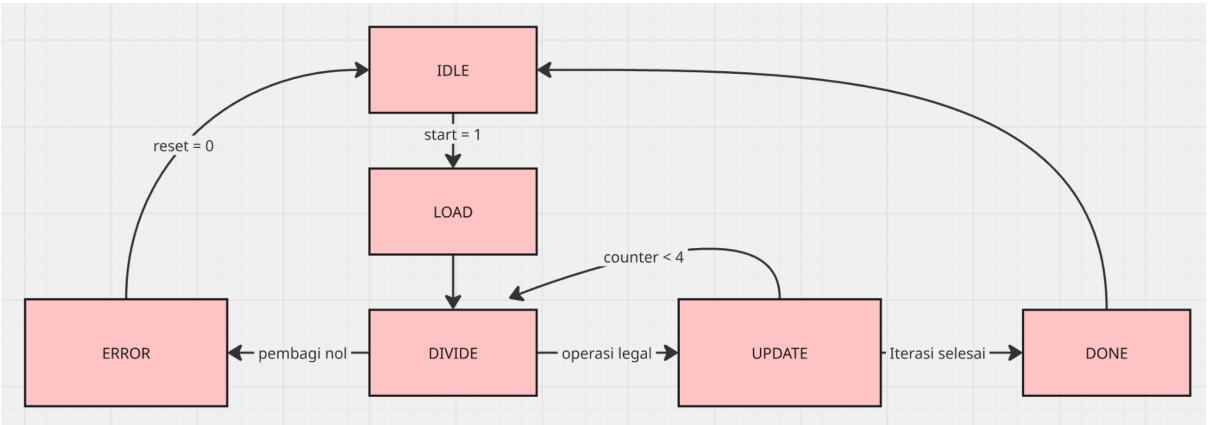
Gambar 2. Diagram alir data sistem

Kendali FSM

FSM berfungsi untuk mengendalikan setiap blok datapath bekerja dalam alur yang tepat. FSM mengatur kapan register menyimpan data, kapan modul divider mulai beroperasi, kapan hasil akhir dapat dibaca, dan mengaktifkan sinyal busy, done, dan error. Penjabaran setiap state FSM ada pada tabel di bawah ini.

Tabel 5. Tabel state sistem beserta fungsi-fungsinya

State	Fungsi
IDLE	Menunggu sinyal start. busy = 0, done = 0, output stabil, semua register tidak menulis. Transisi ke LOAD jika start = 1 dan tetap berada pada IDLE ketika reset = 0 dengan mengatur ulang register.
LOAD	Mengambil data dari ADC dan disimpan dalam register masukan serta menginisiasi iterasi perdana. busy = 1, counter iterasi 0, memilih tebakan awal, divider belum diaktivasi. Setelah melewati satu siklus, transisi ke DIVIDE.
DIVIDE	Mengaktifkan modul pembagi. busy = 1, menunggu sinyal internal ketika proses pembagian selesai, register hasil pembagian belum diperbarui. Jika pembagian selesai, transisi ke UPDATE
UPDATE	Memperbarui nilai iterasi. Aktivasi adder, shifter kanan, dan simpan pada register iterasi serta memperbarui register counter. Jika counter < 4, kembali ke DIVIDE. Jika counter = 4, transisi ke DONE.
DONE	Menghasilkan data akhir untuk dikeluarkan. sqrt_q88 diisi dengan nilai akhir sesuai format, sqrt_int diisi dengan 8-bit pertama dari sqrt_q88, busy = 0, done = 1. Setelah hasil dibaca, kembali ke IDLE.
ERROR	Menangani kasu-kasus operasi ilegal seperti pembagi nol atau output yang tidak valid. error = 1. Jika reset = 0, kembali ke IDLE.



Gambar 2. Diagram FSM

Spesifikasi Sistem

Berikut adalah spesifikasi akhir dari perancangan sistem akar kuadrat digital.

Tabel 6. Tabel state sistem beserta fungsi-fungsinya

Bagian	Spesifikasi
Input	16-bit unsigned integer dengan masukan maksimal 65535
Output	16-bit Q8.8 atau 8-bit truncated integer dengan flag-flag yakni busy, done, dan error.
Komunikasi	Transmisi data dengan protokol UART dan menggunakan ASCII.
Iterasi	Jumlah iterasi yang dilakukan hanya 4 kali.
Siklus	Jumlah siklus adalah 1 siklus untuk LOAD, 16 siklus untuk tiap-tiap operasi DIVIDE, 2 siklus untuk adder, shifter, dan register. Pengulangan sebanyak 4 kali membuat siklus total sebanyak 76 siklus dengan perhitungan konservatif.
Waktu Komputasi	Saat chip berjalan pada frekuensi 50 Mhz, komputasi akan selesai sekitar 1,6 mikrodetik.

Jadwal Kegiatan

Kegiatan	November																			Desember																							
	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Project Kick-off																																																			
Pengambilan FPGA board																																																			
Pemilihan Topik																																																		
Pembuatan Proposal Proyek																																																		
Pendefinisan Spesifikasi dan Perancangan Sistem																																																		
Pengumpulan Proposal Proyek																																																		
Implementasi Modul-Modul Dasar Sistem dalam VHDL dan Simulasi Tahap 1																																																		
Asistensi 1																																																		
Revisi 1																																																		
Implementasi Modul-Modul Dasar Sistem dalam VHDL dan Simulasi Tahap 2																																																		
Implementasi Modul-Modul Dasar Sistem dalam VHDL dan Simulasi Tahap 3																																																		
Pengumpulan Dokumen Rancangan Sistem																																																		
Pemaparan Kemajuan Projek 1																																																		
Asistensi 2																																																		
Revisi 2																																																		
Integrasi Modul-Modul Sistem VHDL dan Simulasi																																																		
Implementasi dan Pengujian Sistem di FPGA+PC																																																		
Pengumpulan Dokumen Implementasi VHDL dan Simulasi																																																		
Pemaparan Kemajuan Projek 2																																																		
Asistensi 3																																																		
Revisi 3																																																		
Pengumpulan Dokumen Implementasi Sistem																																																		
Pemaparan Kemajuan Projek 3																																																		
Asistensi 4																																																		
Revisi 4																																																		
Pemaparan Kemajuan Projek 4																																																		
Asistensi 5																																																		
Revisi 5																																																		
Revisi dan Debugging (Bersifat reaktif dan paralel dengan kegiatan lain)																																																		
Persiapan Ujian Akhir Semester Kuliah																																																		
Libur Natal																																																		
Demonstrasi dan Pemaparan Hasil Kerja																																																		
Pengumpulan Laporan Akhir																																																		

Profil Anggota

	**NIM**	13224001
	Nama	Yozia Gedalya Marcho Ginting
	Asal SMA	SMAK PENABUR Bintaro Jaya
	Asal Daerah	Kota Tangerang Selatan, Banten
	Alamat Bandung	Hegarmanah, Kec. Cidadap, Kota Bandung, Jawa Barat
	HP	+62 856-9453-6989
	Email	13224001@mahasiswa.itb.ac.id

	**NIM**	13224002
	Nama	Jeva Steve Sinaga
	Asal SMA	SMA Unggul Del
	Asal Daerah	Kabupaten Dairi, Sumatera Utara
	Alamat Bandung	Jl. Sekeloa Utara 1, Sekeloa, Kecamatan Coblong, Kota Bandung, Jawa Barat 40134
	HP	+62 813-6126-3091
	Email	13224002@mahasiswa.itb.ac.id

	**NIM**	13224003
	Nama	Benedictus Kenneth Setiadi
	Asal SMA	SMA Santa Angela Bandung
	Asal Daerah	Bandung
	Alamat Bandung	Jl. Merah Delima III No.36c, RW.1, Ciwaruga, Kec. Parongpong, Kabupaten Bandung Barat, Jawa Barat 40559
	HP	+62 811-2275-899
	Email	13224003@mahasiswa.itb.ac.id