BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era teknologi digital yang kini sangat berkembang pesat, kebutuhan akan perangkat komputasi yang memiliki kinerja tinggi namun tetap efisien, cepat, dan hemat energi semakin dibutuhkan. Salah satu elemen penting dalam sistem digital adalah operasi aritmatika, terutama penjumlahan biner yang menjadi dasar dari banyak proses komputasi. Komponen penjumlah biner ini biasanya terdapat dalam *Arithmetic Logic Unit* (ALU), yang berperan penting dalam proses pengolahan data numerik di dalam prosesor. ALU sendiri tidak bisa lepas dari operasi dasar seperti penjumlahan biner, yang menjadi inti dari berbagai perhitungan matematis di dalam sistem digital.

Penjumlahan biner merupakan operasi *fundamental* yang sering dilakukan secara berulang dalam berbagai jenis pemrosesan data digital. Oleh karena itu, desain arsitektur penjumlah yang optimal sangat penting untuk menunjang performa keseluruhan suatu sistem digital, terutama dalam konteks *Very Large Scale Integration* (VLSI). Dalam lingkungan VLSI, efisiensi ruang dan konsumsi daya menjadi faktor kunci yang harus dipertimbangkan dalam setiap rancangan rangkaian logika. Oleh karena itu, penelitian untuk mengembangkan dan membandingkan arsitektur penjumlah biner menjadi hal yang sangat relevan.

Dalam implementasinya, pemilihan jenis arsitektur penjumlah harus disesuaikan dengan kebutuhan sistem. Misalnya, untuk sistem yang mengutamakan kecepatan tinggi, CLA mungkin menjadi pilihan yang lebih baik. Namun, untuk sistem yang lebih sederhana dan hemat area, RCA bisa menjadi alternatif yang lebih efisien. Sementara itu, CSA menawarkan efisiensi tertentu dalam operasi yang membutuhkan penyimpanan sementara dari hasil *carry*.

Perkembangan teknologi VLSI turut mendorong kebutuhan akan desain penjumlah yang optimal. VLSI memungkinkan integrasi jutaan transistor dalam satu *chip*, sehingga efisiensi desain menjadi krusial untuk mengurangi konsumsi daya dan memperkecil ukuran *chip*. Penggunaan *software* seperti DSCH2 dan Microwind2 sangat membantu dalam merancang dan mensimulasikan rangkaian digital sebelum diproduksi secara fisik.

Simulasi yang dilakukan dengan menggunakan *software* DSCH2 memungkinkan perancangan skematik digital yang kemudian dapat dikonversi ke dalam *layout* menggunakan *software* Microwind2. Dengan demikian, dapat dilakukan analisis terhadap berbagai parameter seperti waktu tunda *(delay)*, luas area, serta konsumsi daya dari setiap jenis arsitektur penjumlah. Hasil simulasi ini sangat penting untuk memberikan gambaran nyata mengenai performa masing-masing arsitektur.

Selain aspek teknis, penelitian ini juga berkontribusi dalam memberikan referensi bagi pengembang sistem digital dalam memilih arsitektur penjumlah yang tepat sesuai dengan kebutuhan spesifik aplikasinya. Dalam dunia industri, pemilihan

arsitektur yang efisien tidak hanya berdampak pada performa tetapi juga pada biaya produksi dan konsumsi energi dari sebuah perangkat.

Penelitian ini bertujuan untuk melakukan perbandingan komprehensif antara RCA, CLA, dan CSA dengan menggunakan pendekatan simulasi berbasis *software*. Analisis dilakukan berdasarkan parameter waktu tunda *(delay)*, luas area, dan konsumsi daya. Sehingga, dapat diketahui kelebihan dan kekurangan dari masing-masing arsitektur.

Studi komparatif ini diharapkan memberikan kontribusi nyata dalam pengambilan keputusan desain sistem digital, baik di bidang akademik maupun industri. Hasilnya dapat menjadi acuan bagi pengembang VLSI dalam memilih arsitektur penjumlah yang sesuai dengan tuntutan aplikasi yang dihadapi, baik dari segi kecepatan, efisiensi ruang, maupun konsumsi energi.

Dengan adanya penelitian ini, diharapkan dapat memberikan kontribusi dalam pengembangan desain sirkuit digital yang lebih optimal. Selain itu, hasil penelitian ini juga dapat menjadi referensi bagi akademisi dan praktisi dalam bidang teknik elektro dan komputer dalam mengembangkan perancangan sistem digital yang lebih baik di masa depan.

1.2 Rumusan Masalah

Adapun rumusan masalah dari penelitian yang dilakukan:

1. Bagaimana perbandingan kinerja antara *Ripple Carry Adder* (RCA), *Carry Look Ahead Adder* (CLA), dan *Carry Save Adder* (CSA) dalam hal waktu tunda *(delay)*, konsumsi daya, dan luas area?
2. Penjumlah biner manakah yang paling optimal digunakan untuk desain sirkuit digital berdasarkan hasil simulasi pada perangkat lunak DSCH2 dan Microwind2?
3. Apa keunggulan dan kelemahan masing-masing arsitektur penjumlah dalam konteks implementasi pada sistem digital berbasis VLSI?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian yang dilakukan:

1. Menganalisis dan membandingkan kinerja 3 jenis penjumlah biner yaitu RCA, CLA, dan CSA melalui simulasi berbasis perangkat lunak.
2. Mengidentifikasi arsitektur penjumlah yang paling efisien berdasarkan parameter waktu tunda *(delay)*, konsumsi daya, dan luas area.
3. Memberikan rekomendasi arsitektur penjumlah yang optimal untuk digunakan dalam desain sistem digital berteknologi VLSI.

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian yang dilakukan:

1. Menjadi referensi bagi akademisi dan praktisi dalam menentukan jenis penjumlah biner yang sesuai dengan kebutuhan desain sistem digital.
2. Memberikan gambaran teknis yang komprehensif terkait kelebihan dan kekurangan masing-masing arsitektur penjumlah.

3. Mendukung pengembangan sistem digital yang lebih efisien dalam hal waktu tunda *(delay)*, konsumsi daya, dan luas area khususnya pada aplikasi teknologi VLSI.

1.5 Batasan Masalah

Adapun batasan masalah dari penelitian yang dilakukan:

1. Penelitian ini hanya membandingkan 3 jenis arsitektur penjumlah biner yaitu *Ripple Carry Adder* (RCA), *Carry Look Ahead Adder* (CLA), dan *Carry Save Adder* (CSA).
2. Perancangan dan simulasi dilakukan pada ukuran data *4-bit* untuk masing-masing jenis penjumlah.
3. Simulasi dan analisis dilakukan dengan menggunakan *software* DSCH2 untuk melakukan perancangan skematik dan *software* Microwind2 untuk melakukan analisis *layout* dalam mencapai 3 parameter kinerja yang ingin diteliti.
4. Teknologi yang digunakan dalam simulasi dibatasi pada CMOS 0.12 µm.
5. Parameter kinerja yang dianalisis terbatas pada waktu tunda *(delay)*, konsumsi daya, dan luas area.

1.6 Teori

1.6.1 Penjumlah Biner

Penjumlah biner *(binary adder)* adalah rangkaian logika digital yang digunakan untuk menjumlahkan dua bilangan biner. Dalam sistem digital, penjumlah merupakan bagian penting dari unit aritmetika dan logika (ALU) yang menangani berbagai operasi dasar seperti penjumlahan, pengurangan, dan manipulasi data biner. Setiap hasil penjumlahan menghasilkan dua keluaran, yaitu hasil *(sum)* dan bawaan *(carry)*. Proses ini melibatkan penggunaan gerbang logika XOR untuk menghasilkan nilai *sum*, dan gerbang AND untuk menghasilkan *carry* (Wakerly, 2006).

Terdapat dua jenis dasar dari penjumlah biner yaitu *half adder* dan *full adder*. *Half adder* menjumlahkan dua *bit input*, yaitu A dan B, tanpa mempertimbangkan *carry* dari *bit* sebelumnya. Sementara itu, *full adder* memperhitungkan tiga *input* yaitu dua *bit* yang dijumlahkan (A dan B) serta satu *bit carry-in* dari posisi sebelumnya. Kombinasi dari beberapa *full adder* digunakan untuk membentuk penjumlah *multi-bit* seperti *4-bit adder* (Mano & Ciletti, 2017).

Dalam implementasi lebih lanjut, arsitektur penjumlah berkembang menjadi beberapa bentuk untuk meningkatkan efisiensi dan kecepatan. Salah satu jenis paling sederhana adalah *Ripple Carry Adder* (RCA), yang menjumlahkan *bit* satu per satu dari LSB *(Least Significant Bit)* ke MSB *(Most Significant Bit)*. Namun, kelemahannya adalah propagasi *carry* yang lambat karena rangkaiannya bersifat seri (Weste & Harris, 2010).

Sebagai solusi terhadap masalah *delay* pada RCA, dikembangkanlah *Carry Look Ahead Adder* (CLA), yang menghitung nilai *carry* secara paralel dengan menggunakan logika *generate* dan *propagate*. Teknik ini secara signifikan mempercepat waktu penjumlahan, namun membutuhkan logika lebih kompleks dan area *layout* yang lebih besar (Rabaey, Chandrakasan, & Nikolić, 2003).

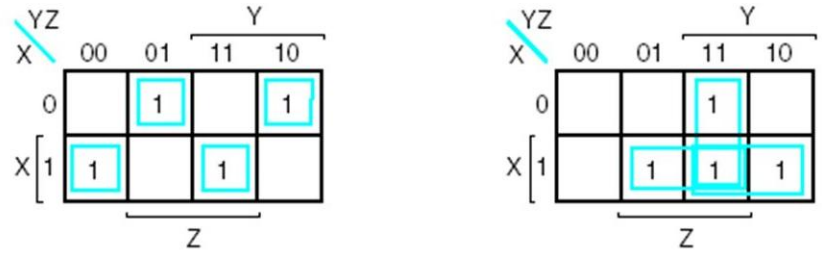
Pemilihan jenis penjumlah sangat bergantung pada kebutuhan aplikasi. Jika sistem memprioritaskan kecepatan, CLA merupakan pilihan yang baik. Namun, jika ruang dan daya adalah prioritas utama, RCA lebih disukai karena desainnya yang sederhana. Dalam beberapa kasus, arsitektur *hybrid* atau pendekatan seperti *Carry Save Adder* (CSA) juga digunakan untuk menyeimbangkan efisiensi dan performa (Weste & Harris, 2010).

1.6.2 *Full Adder*

Full adder adalah salah satu blok dasar dalam sistem logika digital yang dirancang untuk melakukan penjumlahan tiga *bit* biner sekaligus, yaitu dua *bit input* data (A dan B) serta satu *bit carry-in* dari posisi sebelumnya. *Output* dari *full adder* terdiri dari *bit* hasil penjumlahan *(sum)* dan *bit* bawaan *(carry-out)*, yang akan diteruskan ke posisi *bit* berikutnya dalam rangkaian penjumlah *multi-bit* (Mano & Ciletti, 2017).

Tabel 1 Tabel Kebenaran *Full Adder*

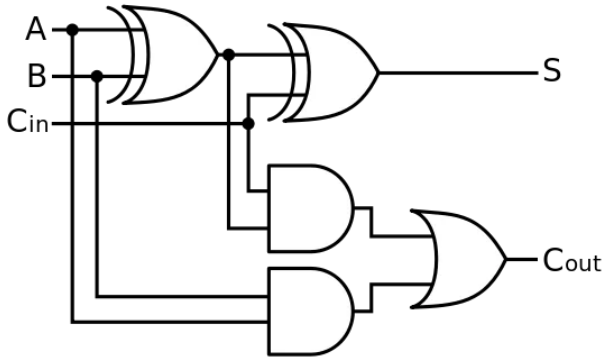
Masukan			Keluaran	
A	B	C_{IN}	SUM	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Gambar 1 *Karnaugh Map Full Adder*

Struktur dasar dari *full adder* biasanya dibangun dengan menggunakan dua buah *half adder* dan satu gerbang OR. *Half adder* pertama menjumlahkan A dan B untuk menghasilkan hasil parsial dan *carry* parsial. *Half adder* kedua kemudian menjumlahkan hasil parsial dengan *carry-in*. Akhirnya, dua nilai *carry* dari masing-masing *half adder* digabungkan menggunakan gerbang OR untuk menghasilkan *carry-out* (Wakerly, 2006).

Dalam representasi aljabar Boolean, *output sum* pada *full adder* dapat dituliskan sebagai $A \oplus B \oplus CIn$, dan *output carry-out* dituliskan sebagai $(A \cdot B) + (CIn \cdot (A \oplus B))$. Rangkaian ini dapat direalisasikan menggunakan gerbang logika dasar seperti AND, OR, dan XOR, yang menjadikan *full adder* sebagai salah satu komponen kunci dalam desain arsitektur digital kompleks seperti ALU dan berbagai jenis penjumlah biner (Rabaey, Chandrakasan, & Nikolić, 2003).

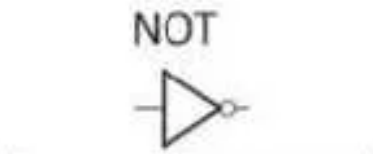


Gambar 2 Rangkaian *Full Adder*

Full adder dapat dihubungkan secara berantai untuk *membentuk Ripple Carry Adder* (RCA) yang mampu menjumlahkan bilangan biner lebih dari satu *bit*. Dalam konfigurasi ini, *carry-out* dari satu *full adder* akan menjadi *carry-in* untuk *full adder* berikutnya. Meskipun desain ini sederhana dan mudah diimplementasikan, kelemahannya adalah waktu tunda yang besar karena propagasi *carry* yang dilakukan secara seri. Dalam implementasi teknologi *Very Large Scale Integration* (VLSI), efisiensi *full adder* sangat penting karena pengaruhnya terhadap performa keseluruhan *chip* (Weste & Harris, 2010).

1.6.2.1 Gerbang NOT

Gerbang NOT, juga dikenal sebagai *inverter* adalah gerbang logika dasar yang hanya memiliki satu *input* dan satu *output*. Fungsinya adalah membalik nilai logika input. Misalnya, jika *input* adalah 1, maka *output* menjadi 0, dan begitu juga sebaliknya. Simbol dari gerbang NOT biasanya dilambangkan dengan segitiga diikuti dengan lingkaran kecil pada ujungnya yang menandakan operasi inversi (Mano & Ciletti, 2017).



Gambar 3 Gerbang NOT

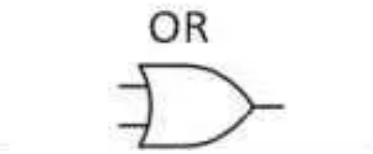
Tabel 2 Tabel Kebenaran Gerbang NOT

INPUT	OUTPUT
A	
0	1
1	0

Dalam sistem digital, gerbang NOT digunakan secara luas dalam berbagai rangkaian logika untuk melakukan negasi logika terhadap sinyal biner. Gerbang ini sangat penting karena memungkinkan pembuatan ekspresi logika yang lebih kompleks seperti NAND, NOR, dan XNOR dari kombinasi gerbang dasar (Wakerly, 2006).

1.6.2.2 Gerbang OR

Gerbang OR adalah gerbang logika yang memberikan *output* logika 1 jika salah satu atau kedua *input*-nya bernilai 1. Jika semua *input* bernilai 0, maka *output* akan menjadi 0. Gerbang ini mewakili operasi logika penjumlahan *(logical addition)* dalam aljabar Boolean, dengan simbol '+' (Rabaey, Chandrakasan, & Nikolić, 2003).



Gambar 4 Gerbang OR

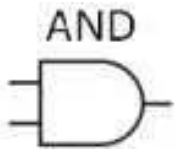
Tabel 3 Tabel Kebenaran Gerbang OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

Dalam aplikasi digital, gerbang OR digunakan untuk mendeteksi kondisi “salah satu benar” dalam rangkaian kontrol, *multiplexer*, dan deteksi sinyal. Fungsi ini sangat berguna dalam pengambilan keputusan logis dalam desain sistem digital kompleks (Weste & Harris, 2010).

1.6.2.3 Gerbang AND

Gerbang AND memberikan *output* logika 1 hanya jika semua *input*-nya bernilai 1. Jika salah satu *input* bernilai 0, maka *output* akan menjadi 0. Operasi AND direpresentasikan oleh simbol ‘.’ dalam aljabar Boolean (Mano & Ciletti, 2017).



Gambar 5 Gerbang AND

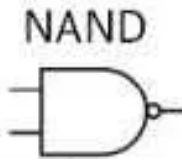
Tabel 4 Tabel Kebenaran Gerbang AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

Gerbang ini digunakan untuk menyatakan kondisi “semua harus benar” dalam logika sistem. Dalam desain rangkaian, gerbang AND digunakan untuk mengatur validasi sinyal, enkripsi logika, dan dalam fungsi kombinatorial seperti penjumlahan biner (Wakerly, 2006).

1.6.2.4 Gerbang NAND

Gerbang NAND (NOT-AND) adalah kebalikan dari gerbang AND. *Output*-nya akan bernilai 0 hanya jika semua *input* bernilai 1; selain itu, *output* bernilai 1. Gerbang ini dikenal sebagai gerbang universal karena bisa digunakan untuk membentuk semua gerbang logika lainnya (Rabaey, Chandrakasan, & Nikolić, 2003).



Gambar 6 Gerbang NAND

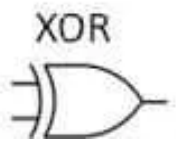
Tabel 5 Tabel Kebenaran Gerbang NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

Secara praktis, gerbang NAND banyak digunakan dalam teknologi VLSI karena lebih mudah diimplementasikan dalam teknologi CMOS. Rangkaian logika kompleks seringkali dibangun hanya dengan gerbang NAND untuk menghemat area dan konsumsi daya (Weste & Harris, 2010).

1.6.2.5 Gerbang XOR

Gerbang XOR (*Exclusive* OR) memberikan *output* logika 1 hanya jika jumlah *input* bernilai 1 adalah ganjil. Untuk dua *input*, *output* adalah 1 jika satu *input* bernilai 1 dan lainnya 0. XOR adalah gerbang penting dalam operasi penjumlahan biner dan fungsi pembanding logika (Mano & Ciletti, 2017).



Gambar 7 Gerbang XOR

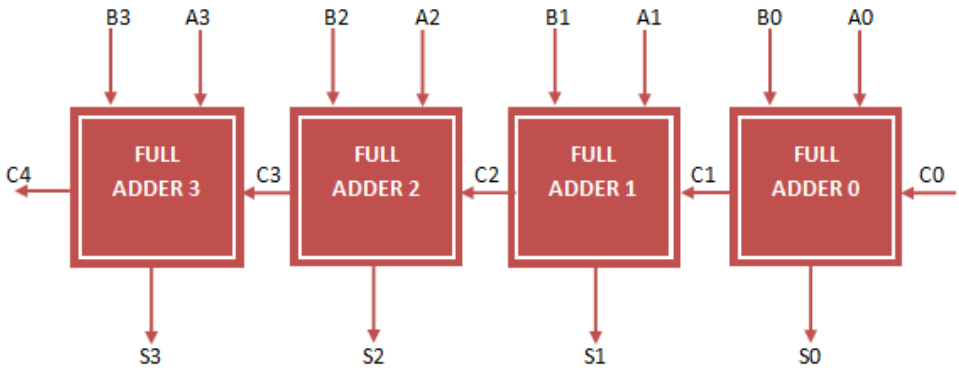
Tabel 6 Tabel Kebenaran Gerbang XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

Gerbang ini digunakan dalam desain rangkaian seperti *half adder* dan *full adder*. Karena sifatnya yang “eksklusif”, XOR memainkan peran penting dalam aplikasi komunikasi digital dan pengolahan sinyal (Wakerly, 2006).

1.6.3 *Ripple Carry Adder* (RCA)

Ripple Carry Adder (RCA) adalah arsitektur penjumlah biner yang paling sederhana dan umum digunakan dalam sistem digital. RCA dibentuk dari rangkaian berantai *full adder*, di mana setiap *carry-out* dari satu tahap akan diteruskan sebagai *carry-in* ke tahap berikutnya. Proses ini dilakukan secara berurutan dari *bit* paling rendah (LSB) ke *bit* paling tinggi (MSB), sehingga menghasilkan struktur yang mudah untuk diimplementasikan (Mano & Ciletti, 2017).



Gambar 8 Rangkaian RCA *4-bit*

Kelebihan utama RCA adalah desainnya yang sederhana dan penggunaan jumlah *gate* logika yang lebih sedikit dibandingkan arsitektur *adder* lainnya. Karena setiap *full adder* hanya bergantung pada hasil dari tahap sebelumnya, maka RCA cocok untuk sistem yang tidak terlalu menuntut kecepatan tinggi (Wakerly, 2006).

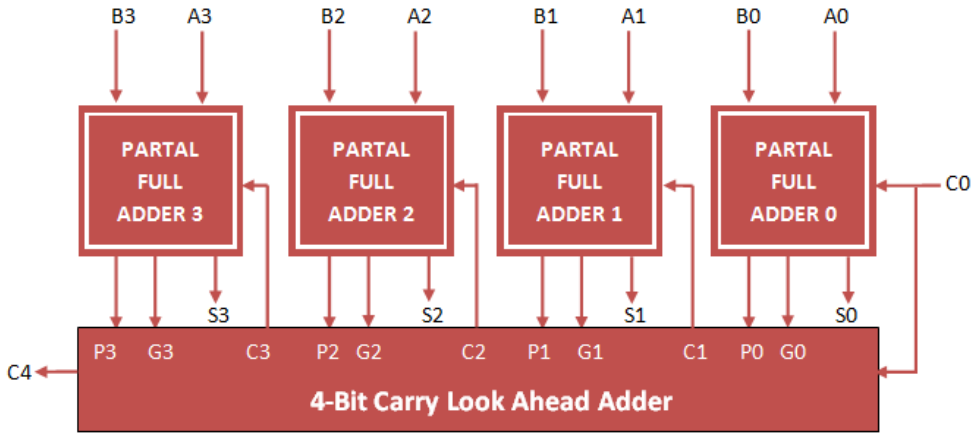
Namun, kelemahan utama RCA terletak pada waktu tunda atau *propagation delay*. Karena *carry* harus merambat dari *bit* ke *bit* secara berurutan, total *delay* akan meningkat secara *linear* terhadap jumlah *bit*. Hal ini membuat RCA kurang efisien dalam aplikasi yang memerlukan kecepatan tinggi (Weste & Harris, 2010).

Meskipun begitu, RCA tetap digunakan dalam berbagai aplikasi *low-power* dan sistem tertanam *(embedded systems)* yang memprioritaskan efisiensi area dan konsumsi daya rendah. Sifatnya yang modular juga memudahkan desain dan analisis (Rabaey, Chandrakasan, & Nikolić, 2003).

RCA sangat berguna sebagai dasar pembelajaran dan sebagai bagian dari desain *hybrid adder* yang menggabungkan kelebihan beberapa arsitektur penjumlah. Dengan pemahaman yang baik tentang RCA, perancang sistem dapat lebih mudah membandingkan dengan arsitektur yang lebih kompleks (Weste & Harris, 2010).

1.6.4 *Carry Look Ahead Adder* (CLA)

Carry Look Ahead Adder (CLA) dikembangkan untuk mengatasi kelemahan RCA dalam hal propagasi *carry*. CLA menggunakan prinsip *generate* dan *propagate* untuk menghitung nilai *carry* pada setiap *bit* secara paralel, bukan serial, sehingga menghasilkan waktu tunda yang jauh lebih cepat (Mano & Ciletti, 2017).



Gambar 9 Rangkaian CLA *4-bit*

Konsep utama CLA adalah menghitung dua sinyal tambahan, yaitu *generate* (G = A·B) dan *propagate* (P = A⊕B). Dengan kedua sinyal ini, nilai *carry* pada masing-masing *bit* dapat dihitung langsung tanpa harus menunggu *bit* sebelumnya. Hal ini membuat CLA sangat efisien untuk operasi aritmatika berkecepatan tinggi (Wakerly, 2006).

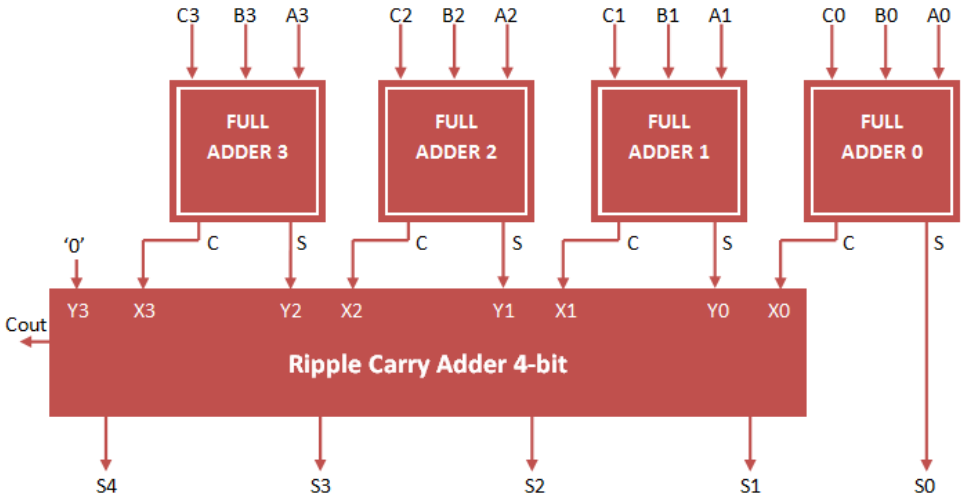
Kelemahan CLA terletak pada kompleksitas rangkaiannya yang lebih tinggi. Untuk setiap *bit* tambahan, dibutuhkan lebih banyak gerbang logika dan interkoneksi, sehingga penggunaan area dan konsumsi daya meningkat dibandingkan RCA (Rabaey, Chandrakasan, & Nikolić, 2003).

Namun, karena kecepatannya yang tinggi, CLA sangat cocok diterapkan dalam prosesor *modern* dan sistem yang memerlukan performa tinggi seperti aplikasi *multimedia* dan pemrosesan sinyal digital. CLA juga dapat dikombinasikan dengan teknik *pipelining* untuk efisiensi lebih lanjut (Weste & Harris, 2010).

Dalam dunia VLSI, CLA merupakan pilihan populer untuk desain arsitektur aritmatika yang menyeimbangkan antara kecepatan dan efisiensi. Beberapa varian CLA bahkan dikembangkan untuk mengurangi kompleksitas logikanya, seperti *Look Ahead Carry Unit* (LCU) atau *Block* CLA (Mano & Ciletti, 2017).

1.6.5 *Carry Save Adder* (CSA)

Carry Save Adder (CSA) merupakan salah satu pendekatan arsitektur penjumlah biner yang lebih efisien dari segi konsumsi daya dan luas area dibandingkan CLA, namun tetap menawarkan kecepatan lebih baik daripada RCA. CSA bekerja dengan cara menyimpan hasil *carry* untuk digunakan pada tahap akhir perhitungan (Weste & Harris, 2010).



Gambar 10 Rangkaian CSA *4-bit*

Alih-alih langsung meneruskan *carry* seperti pada RCA atau menghitungnya secara paralel seperti CLA, CSA menunda pemrosesan *carry* hingga operasi penjumlahan tahap berikutnya. Strategi ini mengurangi jumlah *switching activity* dalam rangkaian, yang secara langsung berdampak pada konsumsi daya lebih rendah (Rabaey, Chandrakasan, & Nikolić, 2003).

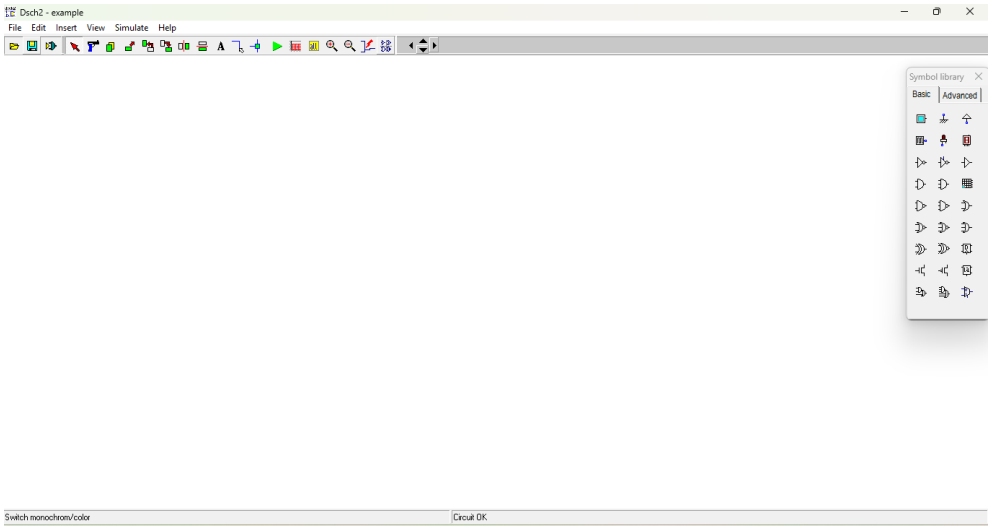
CSA biasanya digunakan dalam aplikasi VLSI yang mengutamakan efisiensi daya dan ukuran *chip*, seperti perangkat *mobile* dan sistem tertanam. Dengan pendekatan penyimpanan *carry*, kompleksitas logika juga menjadi lebih terstruktur, meskipun memerlukan pengelolaan waktu dan urutan pemrosesan yang tepat (Mano & Ciletti, 2017).

Namun, kelemahan utama CSA adalah potensi terjadinya penumpukan *delay* jika tidak dirancang dengan pengendalian waktu yang baik. Karena *carry* disimpan dan baru diproses kemudian, koordinasi antar bagian dalam rangkaian sangat penting untuk menghindari *error* logika (Wakerly, 2006).

Dengan fleksibilitasnya, CSA sering digunakan sebagai bagian dari desain *hybrid adder*, yang menggabungkan elemen-elemen dari RCA dan CLA untuk mencapai keseimbangan antara kecepatan, daya, dan luas area. Hal ini menjadikan CSA relevan dalam pengembangan sirkuit aritmatika *modern* (Weste & Harris, 2010).

1.6.6 *Software* DSCH2

DSCH2 merupakan *software* berbasis Windows yang digunakan untuk merancang dan mensimulasikan rangkaian logika digital dalam bentuk skematik. Aplikasi ini menyediakan antarmuka grafis yang memudahkan pengguna dalam menyusun gerbang logika seperti AND, OR, NOT, serta rangkaian kompleks seperti *adder* dan *multiplexer*. DSCH2 sangat berguna dalam pembelajaran dan eksplorasi desain digital karena memungkinkan analisis waktu dan pemahaman aliran logika secara visual (Smith, 2015).



Gambar 11 Tampilan *Software* DSCH2

Kelebihan utama dari DSCH2 adalah kemampuannya untuk melakukan simulasi waktu nyata *(real-time simulation)*, yang memungkinkan pengamatan *output* dari perubahan *input* secara langsung. Hal ini penting dalam validasi rancangan sebelum diterjemahkan ke *layout* fisik (Kumar & Singh, 2018).

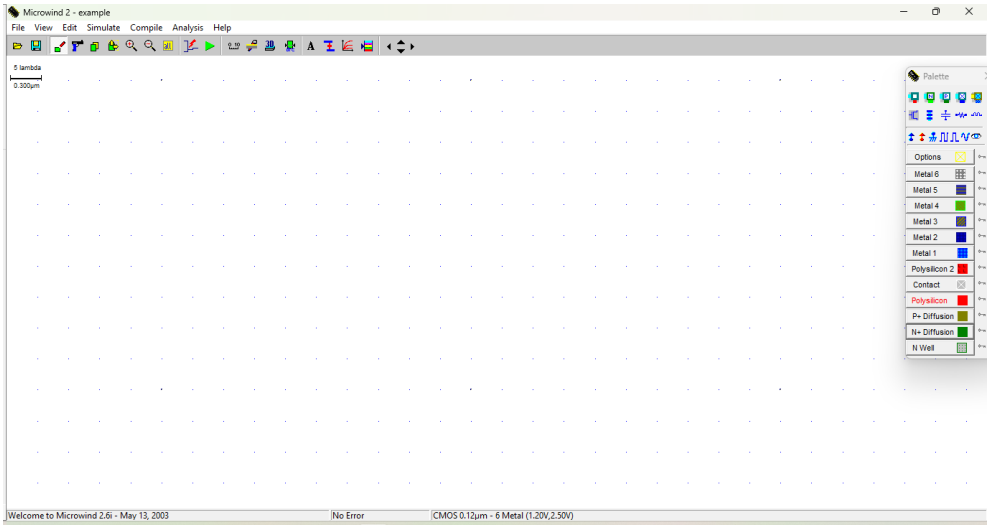
DSCH2 juga memiliki fitur konversi otomatis dari skematik ke kode Verilog, sehingga dapat digunakan sebagai langkah awal dalam pengembangan desain berbasis *Hardware Description Language* (HDL). Integrasi dengan simulasi logika berbasis waktu menjadikan DSCH2 sebagai alat bantu yang lengkap bagi mahasiswa dan perancang sistem digital (Rahman, 2019).

Dalam konteks penelitian atau proyek akhir, DSCH2 sangat berguna dalam merancang dan menguji arsitektur digital seperti *Ripple Carry Adder*, *Carry Look Ahead Adder*, maupun *Carry Save Adder*. Proses perancangan pada DSCH2 memungkinkan analisis fungsional sebelum *layout* difinalisasi di tahap selanjutnya (Weste & Harris, 2010).

Penggunaan DSCH2 sebagai tahap awal dalam alur desain VLSI memberikan efisiensi dalam mengidentifikasi kesalahan logika, serta menyediakan simulasi yang akurat terhadap waktu tunda dan fungsi logika. Dengan demikian, DSCH2 menjadi bagian penting dalam proses desain *top-down* sistem digital (Mano & Ciletti, 2017).

1.6.7 *Software* Microwind2

Microwind2 adalah *software* pelengkap dari DSCH2 yang digunakan untuk merancang *layout* fisik dari rangkaian digital yang telah dirancang secara logika. *Software* ini mensimulasikan desain transistor dan memungkinkan pengguna untuk melihat bagaimana rangkaian akan diimplementasikan pada silikon menggunakan teknologi CMOS. Microwind2 sangat populer dalam bidang desain *Very Large Scale Integration* (VLSI) karena menyediakan lingkungan interaktif untuk simulasi dan analisis *layout* (Smith, 2015).



Gambar 12 Tampilan *Software* Microwind2

Salah satu fitur unggulan dari Microwind2 adalah kemampuannya dalam melakukan simulasi analog digital *(mixed-signal simulation)*, termasuk analisis konsumsi daya, *delay*, dan luas area *chip*. Fitur ini memungkinkan pengguna melakukan validasi performa dari desain digital secara fisik sebelum diproduksi secara nyata (Weste & Harris, 2010).

Microwind2 juga menyediakan dukungan untuk berbagai teknologi CMOS, seperti 0.12 µm, 0.18 µm, hingga 90 nm. Dengan fitur ini, perancang dapat membandingkan hasil *layout* dari berbagai teknologi fabrikasi dan menentukan pendekatan yang paling efisien sesuai kebutuhan sistem (Rahman, 2019).

Selain itu, Microwind2 mendukung konversi desain dari skematik DSCH2 ke *layout* transistor secara otomatis. Hal ini mempercepat alur kerja desain dan membantu mengurangi kesalahan saat transisi dari desain logika ke fisik. Microwind2 juga dapat digunakan untuk menguji efek parasitik dan simulasi *crosstalk* dalam sirkuit padat (Kumar & Singh, 2018).

Dalam proses pendidikan dan penelitian, Microwind2 menjadi alat penting untuk memahami konsep desain tingkat rendah, termasuk *layout*, *routing*, serta *timing analysis*. Kombinasi dengan DSCH2 menjadikannya solusi ideal untuk studi perbandingan arsitektur logika digital secara menyeluruh, baik dari sisi fungsional maupun fisik (Mano & Ciletti, 2017).

1.6.8 *Simulation Testing*

Pengujian simulasi pada perangkat lunak *(simulation testing)* adalah proses yang digunakan untuk mengevaluasi perilaku sistem digital atau analog dalam lingkungan *virtual* sebelum diproduksi secara nyata. Dalam bidang rekayasa elektronik, simulasi memainkan peran penting karena memungkinkan perancang untuk menguji kebenaran logika, kinerja waktu, konsumsi daya, serta efisiensi area

dari suatu sirkuit digital tanpa perlu membangun prototipe fisik terlebih dahulu (Smith, 2015).

Simulasi memberikan fleksibilitas tinggi dalam proses iterasi desain. Seorang desainer dapat melakukan berbagai modifikasi terhadap parameter dan konfigurasi rangkaian, lalu langsung melihat efeknya dalam bentuk grafik atau tampilan waktu nyata. Ini sangat efisien dari segi biaya dan waktu, terutama dalam pengembangan sirkuit digital berbasis teknologi CMOS (Weste & Harris, 2010).

Dalam konteks *software* seperti DSCH2 dan Microwind2, simulasi dilakukan untuk memverifikasi logika skematik serta menganalisis parameter penting seperti delay, konsumsi daya, dan luas area *chip*. Proses ini tidak hanya membantu dalam validasi fungsi rangkaian, tetapi juga menjadi dasar pengambilan keputusan dalam memilih arsitektur terbaik untuk diterapkan di sistem digital (Kumar & Singh, 2018).

Selain efisiensi teknis, pengujian simulasi juga mendukung dokumentasi hasil yang akurat dan *repeatable*, karena data hasil simulasi bisa disimpan, dibagikan, dan dibandingkan secara objektif. Ini penting dalam dunia akademik maupun industri untuk menjamin kualitas dan keandalan produk akhir (Rahman, 2019).

Dengan semakin berkembangnya *software* simulasi, proses desain dan verifikasi kini menjadi lebih terintegrasi dan cepat. Bahkan, *software* yang *modern* sudah mendukung simulasi berbasis waktu, *mixed-signal*, dan estimasi performa dalam skala nanometer, menjadikan simulasi sebagai komponen tak terpisahkan dalam siklus pengembangan sistem digital *modern* (Mano & Ciletti, 2017).

BAB II
METODE PENELITIAN

2.1 Waktu dan Lokasi Penelitian

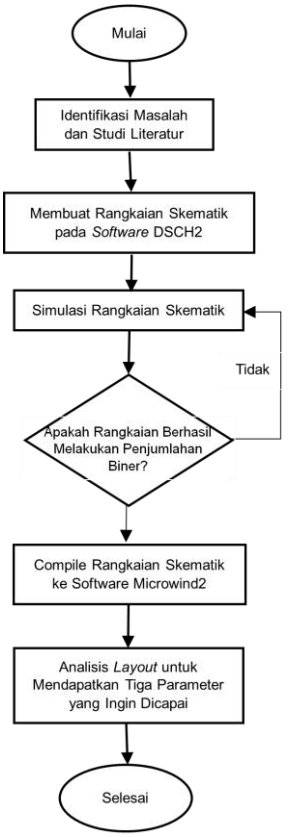
Penelitian ini dilakukan di Departemen Teknik Elektro, Fakultas Teknik, Universitas Hasanuddin yang terletak di Jalan Poros Malino Km. 6, Kelurahan Romang Lompoa, Kecamatan Bontomarannu, Kabupaten Gowa, Sulawesi Selatan, 92171. Penelitian ini dilaksanakan mulai dari bulan Juni 2025 hingga Juli 2025.

2.2 Alat dan Bahan

Adapun alat yang dibutuhkan dalam penelitian ini yaitu sebuah Laptop yang digunakan sebagai alat untuk membuat skematik rangkaian pada *software* DSCH2 dan juga menganalisa hasil simulasi dari *layout* untuk mendapatkan 3 parameter yang ingin diteliti pada *software* Microwind2.

2.3 Alur Penelitian

Alur penelitian dibuat dengan tujuan untuk menyelesaikan masalah yang ada secara terstruktur. Berikut alur penelitian yang akan dilakukan:



Gambar 13 Alur Penelitian

2.4 Teknik Analisis dan Pengumpulan Data

Penelitian ini tergolong sebagai penelitian kuantitatif eksperimental, yang bertujuan untuk menganalisis dan membandingkan performa dari 3 jenis arsitektur penjumlah biner, yaitu *Ripple Carry Adder* (RCA), *Carry Look Ahead Adder* (CLA), dan *Carry Save Adder* (CSA). Pendekatan dilakukan melalui simulasi *software* guna memperoleh hasil yang bersifat objektif berdasarkan parameter teknis yang telah ditentukan.

Teknik pengumpulan data dilakukan melalui kajian literatur terhadap teori-teori yang berkaitan dengan sistem penjumlah biner. Referensi diperoleh dari buku teks, jurnal ilmiah, artikel konferensi, serta laporan tugas akhir mahasiswa terdahulu yang memiliki relevansi terhadap topik penelitian. Selanjutnya, masing-masing arsitektur penjumlah (RCA, CLA, dan CSA) dirancang dalam bentuk skematik digital menggunakan *software* DSCH2, dengan konfigurasi rangkaian *4-bit* serta struktur logika yang sesuai dengan prinsip kerjanya.

Setelah proses perancangan selesai, rangkaian digital yang telah dibuat pada *software* DSCH2 kemudian dikonversi menjadi *layout* fisik menggunakan *software* Microwind2. Melalui *layout* ini, diperoleh data mengenai luas area *chip (chip area)*, konsumsi daya, serta waktu tunda *(delay)*. Data hasil simulasi tersebut selanjutnya dianalisis dan dibandingkan guna menentukan arsitektur yang paling optimal untuk kebutuhan sistem digital tertentu.