Praktikum Sistem Digital
Responsi 2 – Rangkuman Seluruh Materi

Muhammad Alwiza Ansyar_M0520051

Materi 1: LOGIKA DASAR

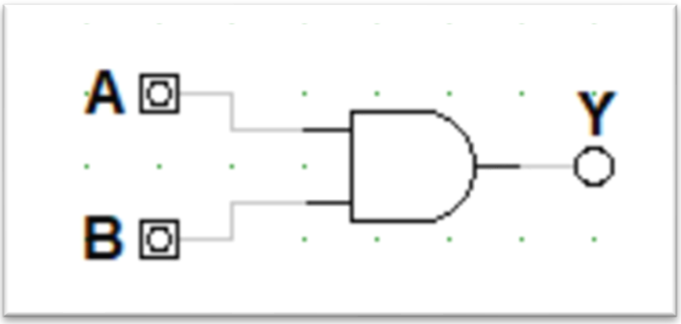
Gerbang logika merupakan rangkaian yang mengolah *input-input* yang berupa bilangan *biner* dengan menggunakan Teori Matematika Boolean sehingga dihasilkan sebuah sinyal *output*. Gerbang logika memiliki satu atau lebih sinyal masukan (*input*) dan menghasilkan satu sinyal keluaran (*output*). Masukan dan keluaran merupakan sinyal digital yang memiliki dua kondisi, yaitu tegangan tinggi/logika tinggi/*high logic*/logika 1 atau tegangan rendah/logika rendah/*low logic*/logika 0.

Gerbang logika dasar memiliki 7 macam yaitu gerbang AND, gerbang OR, gerbang NOT, gerbang NAND, gerbang NOR, gerbang XOR, dan gerbang XNOR

Jenis-jenis gerbang logika

1. Gerbang AND

   Ialah gerbang yang hanya akan menghasilkan *output* logika 1 jika semua *input* merupakan logika 1. Selain itu, gerbang akan menghasilkan *output* logika 0.



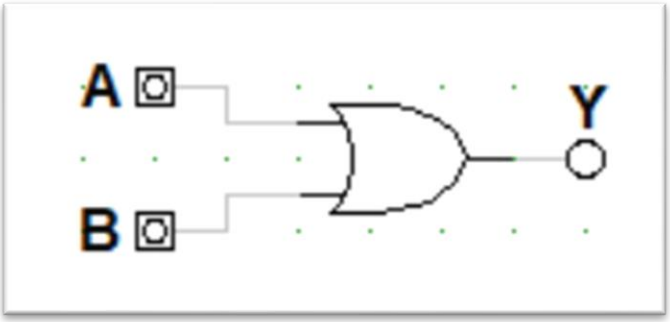
Tabel kebenaran:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Ekspresi: Y = A.B

2. Gerbang OR

   Ialah gerbang yang hanya akan menghasilkan *output* logika 0 jika semua *input* merupakan logika 0. Selain itu, gerbang akan menghasilkan *output* logika 1.



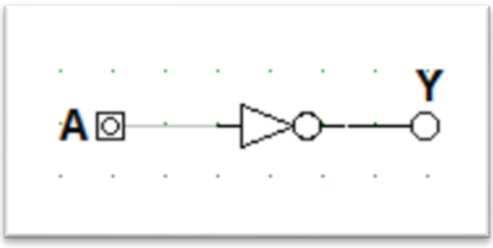
Tabel kebenaran:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Ekspresi: Y = A+B

3. Gerbang NOT

   Ialah gerbang yang meng-*invert* atau membalikkan logika *input*. Gerbang ini hanya memiliki satu *terminal input*.



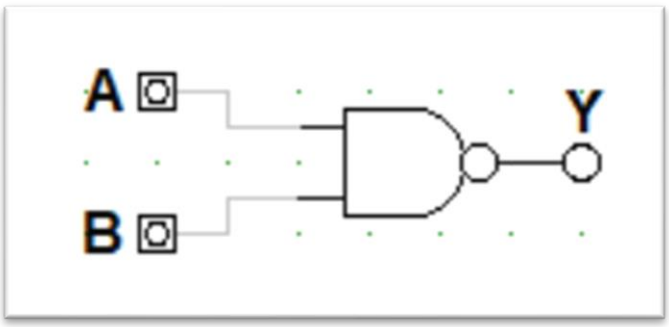
Tabel kebenaran:

A	Y
0	1
1	0

Ekspresi: Y = A’

4. Gerbang NAND

   Ialah perpaduan antara gerbang AND dan NOT. Gerbang ini hanya akan menghasilkan *output* logika 0 jika semua *input* merupakan logika 1. Selain itu, gerbang akan menghasilkan *output* 1.



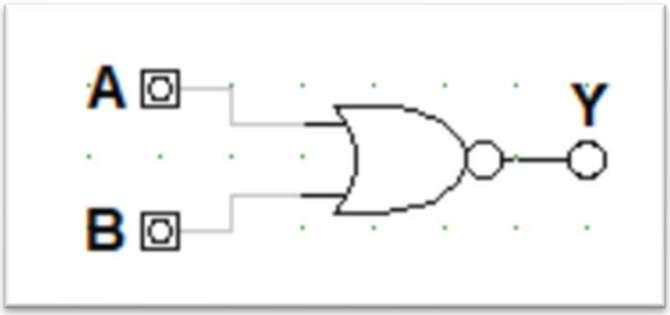
Tabel kebenaran:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Ekspresi: Y = (A.B)’

5. Gerbang NOR

   Ialah perpaduan antara gerbang OR dan NOT. Gerbang ini hanya akan menghasilkan *output* logika 1 jika semua *input* merupakan logika 0. Selain itu, gerbang ini menghasilkan *output* 0



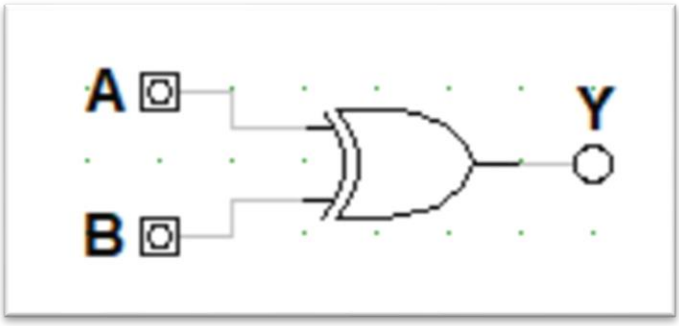
Tabel kebenaran:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Ekspresi: Y = (A+B)’

6. Gerbang XOR

   Ialah sebuah gerbang eksklusif atau khusus dari gerbang OR. Gerbang ini hanya akan menghasilkan *output* logika 0 jika semua *input* merupakan logika yang sama. Selain itu, gerbang akan menghasilkan *output* logika 1



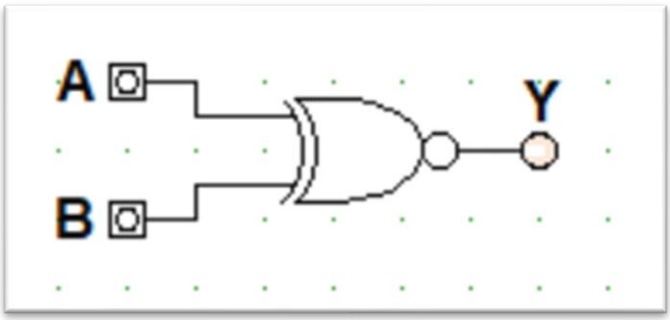
Tabel kebenaran:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Eksrpresi: Y = (A⊕B)

7. Gerbang XNOR

   Ialah perpaduan antara gerbang XOR dan NOT. Gerbang ini hanya akan menghasilkan *output* logika 1 jika semua *input* merupakan logika yang sama. Selain itu, gerbang akan menghasilkan *output* logika 0.



Tabel kebenaran:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Ekspresi: Y = (A⊕B)'

Materi 2: DALIL-DALIL ALJABAR BOOLEAN

Aljabar Boolean

Aljabar Boolean adalah matematika yang digunakan untuk menganalisis dan menyederhanakan gerbang logika pada rangkaian-rangkaian digital elektronika. Boolean pada dasarnya merupakan tipe data yang hanya terdiri dari dua nilai yaitu “True” dan “False atau “Tinggi” dan “Rendah” yang biasanya dilambangkan dengan angka “1” dan “0” pada gerbang logika ataupun bahasa pemrograman komputer.

Hukum-hukum Boolean:

1	Identitas	X + 0 = X	X . 1 = X
2	Komplemen	X + X' = 1	X . X' = 0
3		X + X = X	X . X = X
4		X + 1 = 1	X . 0 = 0
5	Involution	(X')' = X	
6	Commutative	X + Y = Y + X	X . Y = Y . X
7	Associative	X + (Y + Z) = (X + Y) + Z	X.(Y.Z) = (X.Y).Z
8	Distributive	X.(Y+ Z) = (X.Y)+(X.Z)	X+(Y.Z) = (X+Y).(X+Z)
9	De Morgan	(X+Y)' = X' . Y'	(XY)' = X' + Y'
10	Absorption	X + X.Y = X	X.(X+Y) = X

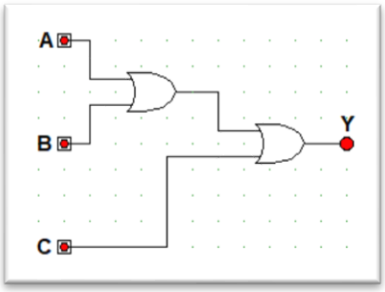
Pada Materi ini akan dibahas pembuktian dalil asosiatif dan distributif.

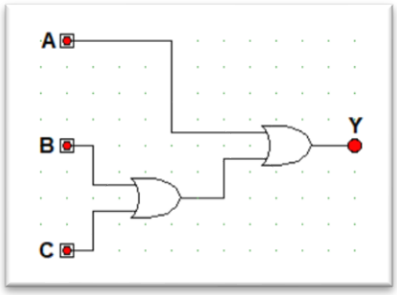
Dalil Asosiatif

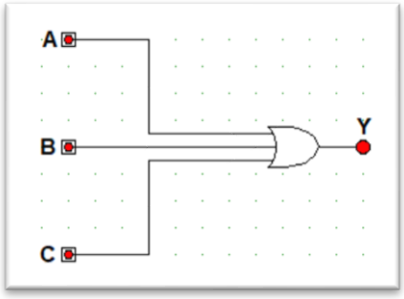
Dalil ini menyatakan bahwa urutan rangkaian logika pada operasi Aljabar Boolean tidak akan mempengaruhi *output* rangkaian tersebut.

Contoh:

$Y = (A + B) + C$ *sama dengan* $Y = A + (B + C)$ *sama dengan* $Y = A + B + C$



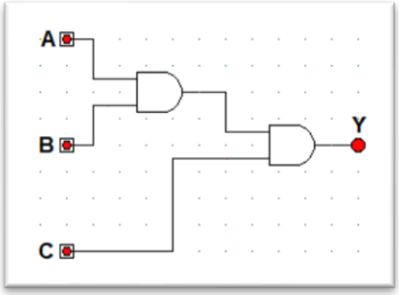


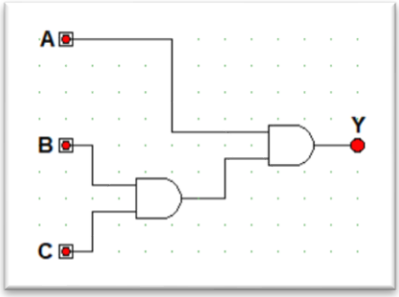


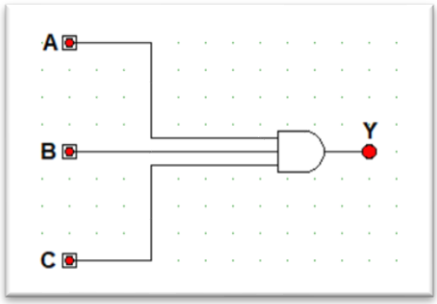
Tabel kebenaran:

A	B	C	(A + B)	(B + C)	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

$Y = (A \,.\, B) \,.\, C$ *sama dengan* $Y = A \,.\, (B \,.\, C)$ *sama dengan* $Y = A \,.\, B \,.\, C$







Tabel kebenaran:

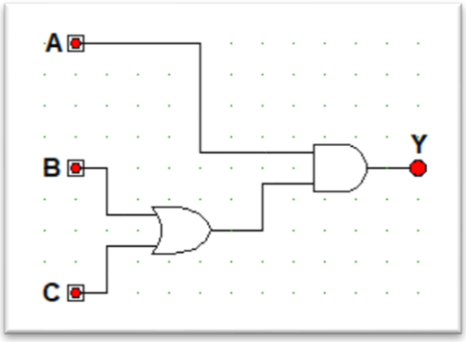
A	B	C	(A . B)	(B . C)	Y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	0
1	1	1	1	1	1

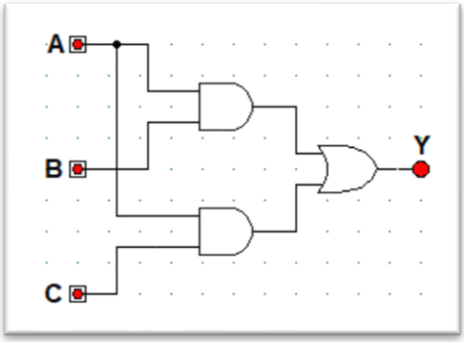
Dalil Distributif

Dalil ini menyatakan bahwa variabel-variabel pada operasi Aljabar Boolean dapat disebarkan tempatnya maupun diubah urutan sinyalnya tanpa mempengaruhi *output* rangkaian tersebut.

Contoh:

$Y = A\,.\,(B + C)\ sama\ dengan\ Y = (A\,.\,B) + (A\,.\,C)$

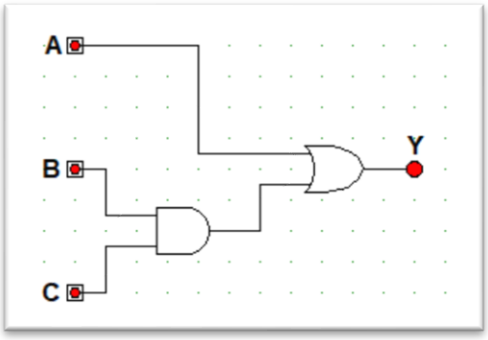


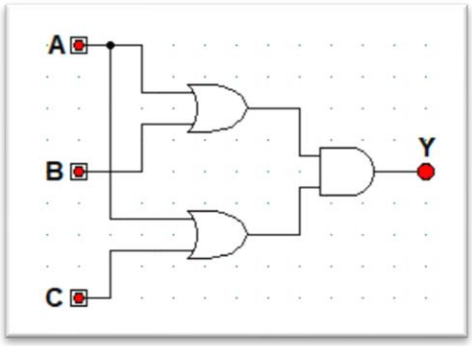


Tabel kebenaran:

A	B	C	(B + C)	(A . B)	(A . C)	Y
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

$Y = A + (B\,.\,C)\ sama\ dengan\ Y = (A + B)\,.\ (A + C)$





Tabel kebenaran:

A	B	C	(B . C)	(A + B)	(A + C)	Y
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

Materi 3: PENYEDERHANAAN ALJABAR BOOLEAN

Fungsi Boolean

Fungsi Boolean adalah fungsi yang dibentuk oleh n variabel Aljabar Boolean. Ekspresi Boolean adalah Fungsi Boolean pula. Variabel pada fungsi Boolean disebut peubah atau literal

Bentuk Kanonik dan Bentuk Baku

Fungsi Boolean dapat dinotasikan dalam bentuk kanonik maupun bentuk baku. Bentuk kanonik adalah bentuk yang memperlihatkan setiap literal yang ada pada fungsi, baik dalam bentuk biasa maupun komplemen. Bentuk baku adalah bentuk yang tidak diharuskan untuk memperlihatkan setiap literal yang ada pada fungsi. Bentuk baku merupakan bentuk yang lebih sederhana daripada bentuk kanonik.

Sum-of-Product (SOP)

Sum-of-Product (SOP) adalah kombinasi literal yang menghasilkan *output* fungsi berupa 1. Setiap suku pada SOP kanonik disebut *minterm*. SOP mengekspresikan fungsi dengan menambahkan (menggunakan OR) setiap *minterm*, yang mana *minterm* mengekspresikan suku SOP dengan mengkalikan (menggunakan AND) setiap literal.

Minterm dapat dinotasikan dengan m_p dengan p adalah urutan *output* fungsi yang dimulai dari 0 sampai dengan $2^n - 1$ dengan n adalah jumlah literal. Pada *minterm*, 0 diekspresikan menjadi literal komplemen sedangkan 1 diekspresikan menjadi literal.

Product-of-Sum (POS)

Product-of-Sum (POS) adalah kombinasi literal yang menghasilkan *output* fungsi berupa 0. Setiap suku pada SOP kanonik disebut *maxterm*. POS mengekspresikan fungsi dengan mengkalikan (menggunakan AND) setiap *maxterm*, yang mana *maxterm* mengekspresikan suku POS dengan menambahkan (menggunakan OR) setiap literal.

Maxterm dapat dinotasikan dengan M_p dengan p adalah urutan *output* fungsi yang dimulai dari 0 sampai dengan $2^n - 1$ dengan n adalah jumlah literal. Pada *maxterm*, 0 diekspresikan menjadi literal sedangkan 1 diekspresikan menjadi literal komplemen.

Karnaugh Map's Method

Karnaugh's *Map Method* atau metode peta Karnaugh adalah salah satu metode untuk menyederhanakan persamaan Aljabar Boolean. Metode ini diperkenalkan pertama kali oleh Maurice Karnaugh pada tahun 1953. Metode ini menggunakan peta Karnaugh yang mana peta tersebut memetakan setiap kombinasi literal pada fungsi dalam bentuk tabel. Kombinasi literal yang akan disederhanakan diekspresikan dengan 1 sedangkan kombinasi literal yang tidak akan disederhanakan diekspresikan dengan 0.

Penyederhanaan dilakukan dengan cara mengelompokkan 1 yang berdekatan. Pengelompokkan tersebut terdiri dari kelipatan 2 yang ditandai dengan membuat kotak, membuat lingkaran, atau menyamakan warna sel tesebut pada tabel. Setiap kelompok lalu disederhanakan dengan menghilangkan literal yang mengandung 0 dan 1.

Contoh:

A	B	C	F(A,B,C)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Penyederhanaan SOP menggunakan KMAP:

Kombinasi literal yang memiliki *output* 1 adalah 000, 100, 101, dan 111. Maka:

$F(A,B,C) = m_0 + m_4 + m_5 + m_7 = \sum (0, 4, 5, 7)$

F(A,B,C) = A'B'C' + AB'C' + AB'C + ABC

Untuk menyederhanakan ekspresi di atas, digunakan Karnaugh's *Map Method*

F(A,B,C) = A'B'C' + AB'C' + AB'C + ABC [bentuk kanonik]

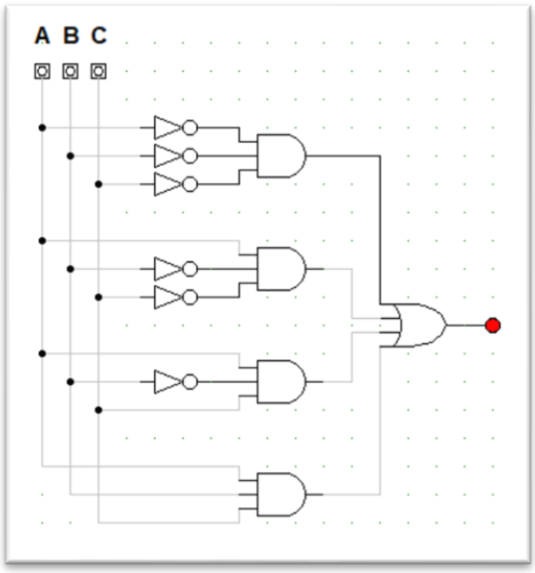
Peta Karnaugh:

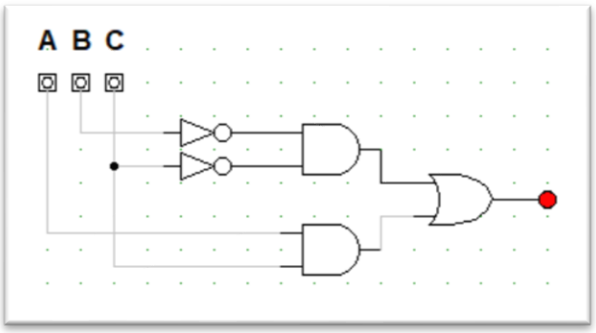
A \ BC	00	01	11	10
0	1	0	0	0
1	1	1	1	0

Hasil:

F(A,B,C) = B'C' + AC [bentuk baku]

Perbandingan rangkaian logika:





Tabel kebenaran:

A	B	C	A'B'C'	AB'C'	AB'C	ABC	B'C'	AC	F(A,B,C)
0	0	0	1	0	0	0	1	0	1
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0	1
1	0	1	0	0	1	0	0	1	1
1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	1	1

Penyederhanaan POS menggunakan KMAP:

Kombinasi yang memiliki *output* 0 adalah 001, 010, 011, dan 110. Maka:

$F(A,B,C) = M_1 . M_2 . M_3 . M_6 = \prod (1, 2, 3, 6)$

F(A,B,C) = (A+B+C') . (A+B'+C) . (A+B'+C') . (A'+B'+C)

Untuk menyederhanakan ekspresi di atas, digunakan Karnaugh's *Map Method*

F(A,B,C) = (A+B+C') . (A+B'+C) . (A+B'+C') . (A'+B'+C) [bentuk kanonik]

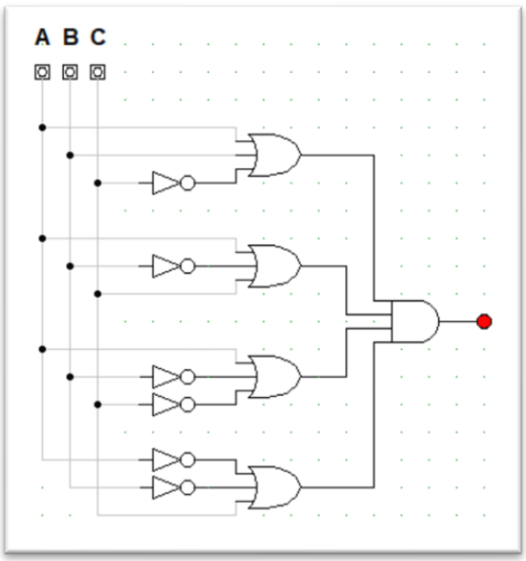
Peta Karnaugh:

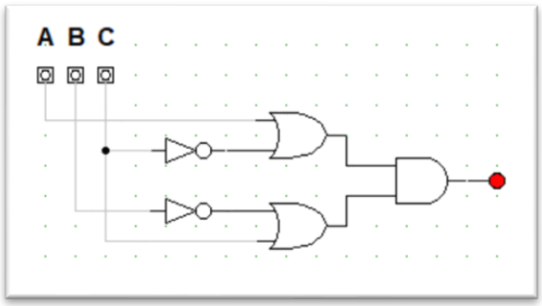
A \ BC	00	01	11	10
0	0	1	1	1
1	0	0	0	1

Hasil:

F(A,B,C) = (A+C') . (B'+C) [bentuk baku]

Perbandingan rangkaian logika:





Tabel kebenaran:

A	B	C	A+B+C'	A+B'+C	A+B'+C'	A'+B'+C	A+C'	B'+C	F(A,B,C)
0	0	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	0	1	0
0	1	0	1	0	1	1	1	0	0
0	1	1	1	1	0	1	0	1	0
1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1

Materi 4: ADDER dan KOMPARATOR

Penjumlahan Biner

Sama seperti penjumlahan desimal (0-9), bilangan biner juga dapat dijumlahkan dengan cara yang sama namun memiliki penulisan yang berbeda. Pada penjumlahan biner, terdapat *summary* (S) yang berarti hasil dan *carry* (C) yang berarti pembawa.

Penjumlahan biner dilakukan dari LSB (Least Significant Biner). Untuk 0 + 0, 1 + 0, dan 0 + 1, dilakukan penjumlahan seperti pada umumnya, yaitu dengan hasil/*summary* (S) 0, 1, dan 1 secara berurutan dan tidak menghasilkan *carry output* (C_{out}) sehingga *carry* ialah 0. Untuk 1 + 1, maka hasilnya adalah 0 dan *carry output* menjadi 1. Maka, penjumlahan tersebut dapat ditulis menjadi 1 + 1 = 10 dengan biner di kanan adalah *summary* dan biner di kiri adalah *carry output*. Hasil 10 sendiri adalah penulisan biner dengan nilai sebesar 2 yang mana seperti penjumlahan desimal, 1 + 1 = 2.

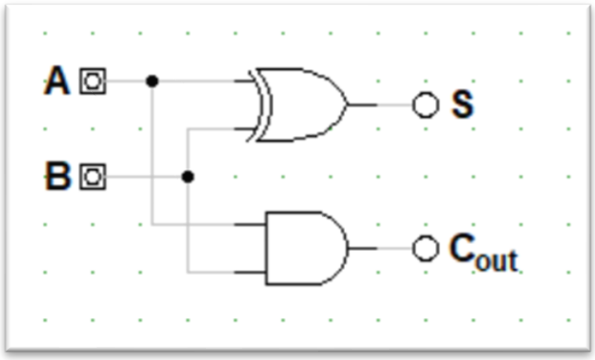
Penjumlahan juga dapat dikombinasikan dengan *carry input* (C_{in}) sehingga penjumlahan akan saling terhubung melalui *carry input* dan *carry output*.

Adder

Adder merupakan rangkaian logika kombinasional yang berfungsi untuk melakukan penjumlahan bilangan biner. *Adder* juga merupakan rangkaian ALU (*Arithmetic and Logic Unit*). *Adder* dibagi menjadi tiga yaitu.

1. *Half Adder*

   Ialah rangkaian *adder* yang paling sederhana yang mana *adder* ini tidak memiliki *carry input*. *Adder* ini hanya bisa digunakan untuk penjumlahan 1-Bit.



Tabel kebenaran:

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

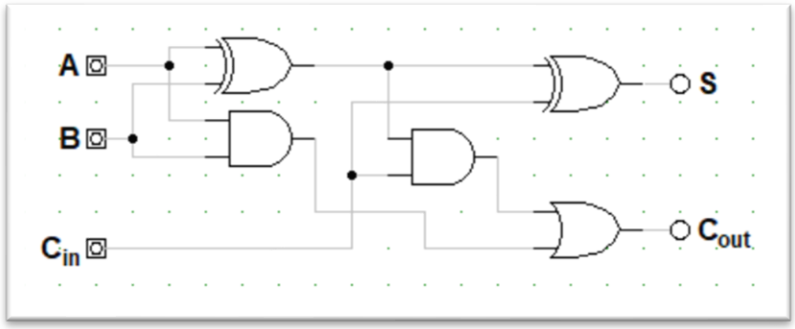
Ekspresi output:

S = A⊕B

C_{out} = A . B

2. *Full Adder*

   Ialah rangkaian *adder* yang memiliki *carry input*. *Adder* ini bisa digunakan untuk penjumlahan 1-Bit dan lebih.



Tabel kebenaran:

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

   Ekspresi output:

   $S = A'B'C_{in} + A'B\,C_{in}' + AB'\,C_{in}' + AB\,C_{in}$

   $C_{out} = A'B\,C_{in} + AB'\,C_{in} + AB\,C_{in}' + AB\,C_{in}$

3. *Full Adder* Paralel

   Ialah rangkaian *Full Adder* yang disusun paralel. *Adder* ini dapat tersusun dari *Full Adder* 1-Bit yang diparalelkan, *Full Adder* 4-Bit yang diparalelkan, dan seterusnya sesuai kebutuhan. Rangkaian ini digunakan untuk penjumlahan Bit yang besar dan supaya rangkaian lebih terorganisir.

Komparator

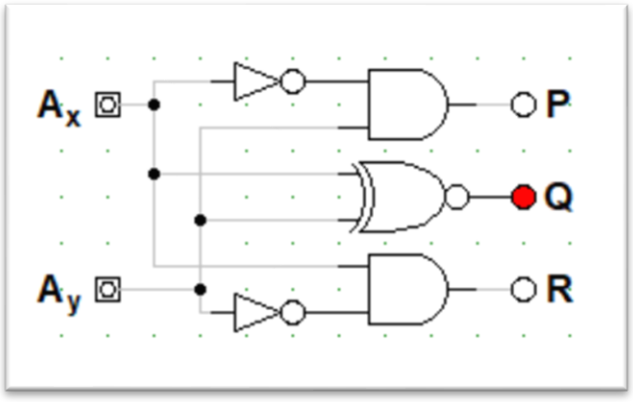
Komparator adalah rangkaian logika kombinasional yang berfungsi untuk membanding keadaan logika *input-input*nya. Komparator biner dibagi menjadi dua yaitu sebagai berikut.

1. *Non-equality comparator*

   Ialah komparator yang memberikan *output* 1 apabila keadaan *input*nya memiliki logika yang berbeda-beda

2. *Equality comparator*

   Ialah komparator yang memberikan *output* 1 apabila keadaan *input*nya memiliki logika yang sama



Tabel kebenaran:

A_x	A_y	P	Q	R
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Ekspresi output:

P $= A_x'A_y$ $[A_x < A_y]$ (*Non-equality comparator*)

Q $= A_x'A_y' + A_xA_y$ $[A_x = A_y]$ (*Equality comparator*)

R $= A_xA_y'$ $[A_x > A_y]$ (*Non-equality comparator*)

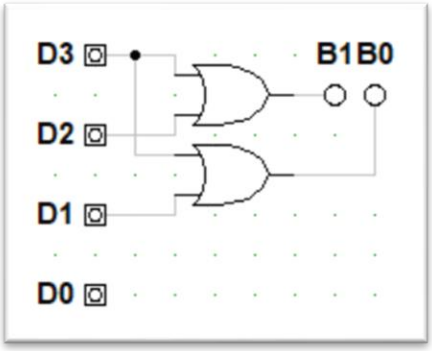
Materi 5: ENKODER dan DEKODER

Encoder

Encoder adalah alat yang berfungsi untuk mengkodekan data input menjadi data bilangan dengan format tertentu. Encoder dalam rangkaian digital merupakan rangkaian kombinasional yang memiliki input dalam format tertentu dan ouput dalam format bilangan biner. Sifat dari encoder adalah memiliki banyak input dan sedikit output.

Pada encoder, kombinasi input hanya boleh memiliki satu input yang bernilai 1. Jika lebih, output dari encoder akan menjadi tidak berarti. Contoh encoder: 10-to-4 encoder/Decimal to BCD encoder, 8 line to 3 line encoder, dan 4 line to 2 line encoder.

Contoh: *Encoder* Desimal ke Biner 4 to 2



Kombinasi input hanya boleh memiliki satu input yang bernilai 1 supaya output menjadi berarti.

Tabel kebenaran:

D0	D1	D2	D3	B1	B0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Hasil:

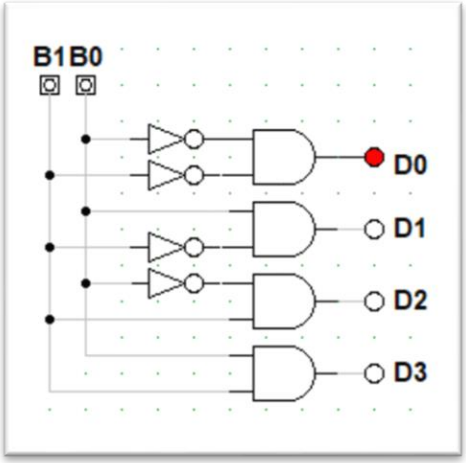
Format bilangan desimal	Format bilangan biner
0	00
1	01
2	10
3	11

Decoder

Decoder juga adalah alat yang berfungsi untuk mengkodekan data input menjadi data bilangan dengan format tertentu. Yang membedakannya dengan encoder adalah decoder dalam praksisnya biasa digunakan untuk mengembalikan proses encoding sehingga didapatkan bentuk informasi aslinya. Decoder dalam rangkaian digital merupakan rangkaian kombinasional yang memiliki input dalam format bilangan biner dan ouput dalam format tertentu. Sifat dari decoder adalah memiliki sedikit input dan banyak output.

Pada decoder, setiap kombinasi input hanya akan menghasilkan salah satu output bernilai 1. Namun, terdapat pengecualian untuk jenis BCD to 7 segment dimana kombinasi input decoder tersebut dapat menghasilkan nilai 1 di lebih dari satu output. Contoh decoder: 4-to-10 decoder/BCD to Decimal decoder, BCD to 7 segment decoder, 3 line to 8 line decoder, dan 2 line to 4 line decoder.

Contoh: *Decoder* Biner ke Desimal sederhana



Kombinasi input hanya akan menghasilkan satu output yang bernilai 1.

Tabel kebenaran:

B1	B0	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Hasil:

Format bilangan biner	Format bilangan desimal
00	0
01	1
10	2
11	3

Materi 6: MULTIPLEKSER dan DEMULTIPLEKSER

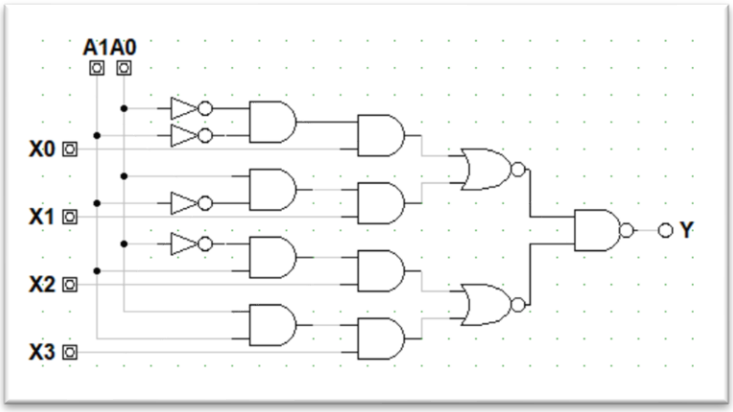
Multiplexer

Multiplexer adalah suatu rangkaian kombinasional yang menyeleksi *data input* untuk diteruskan pada *output* menggunakan *selector* (yang juga merupakan *input*). Rangkaian ini sering disebut data *selector* dan disingkat mux. Mux memiliki dua jenis *input*, yaitu *data input* dan *selector input*. *Selector input* akan memilih *data input* mana yang akan diteruskan ke *output*, sedangkan *data input* sendiri membawa informasi yang akan diteruskan (seperti apakah berupa low logic atau high logic). Kondisi logika dari *data input* yang terpilih oleh *selector* akan ditampilkan pada *output*.

Mux memiliki ciri khas yaitu hanya memiliki satu *output*. Pada beberapa mux, terdapat *input* tambahan berupa enabler *input*, yaitu *input* yang menentukan apakah mux beroperasi atau tidak beroperasi (seperti tombol on/off). Beberapa jenis mux antara lain 2x1 Multiplexer, 4x1 Multiplexer, 8x1 Multiplexer, dan 16x1 Multiplexer.

Terdapat hubungan antara jumlah *data input* dan *selector input*, yaitu $n = 2^s$, dengan n adalah jumlah *data input* dan s adalah jumlah *selector input*. Maka dari itu, jumlah *data input* dari multiplexer selalu merupakan hasil dari perpangkatan

Contoh: *Multiplexer* 4x1



Tabel kebenaran:

A1	A0	X0	X1	X2	X3	Y
0	0	0	×	×	×	0
0	0	1	×	×	×	1
0	1	×	0	×	×	0
0	1	×	1	×	×	1
1	0	×	×	0	×	0
1	0	×	×	1	×	1
1	1	×	×	×	0	0
1	1	×	×	×	1	1

Pada rangkaian ini, A0 dan A1 adalah *selector input*, X0, X1, X2, dan X3 adalah *data input*, serta Y adalah *output*. Ketika suatu *data input* dipilih oleh *selector*, maka *output* akan menampilkan kondisi logika dari *data input* tersebut, sedangkan kondisi logika *data input* lain yang tidak terpilih adalah tidak dipedulikan/keadaan *don't care* (ditandai dengan ×).

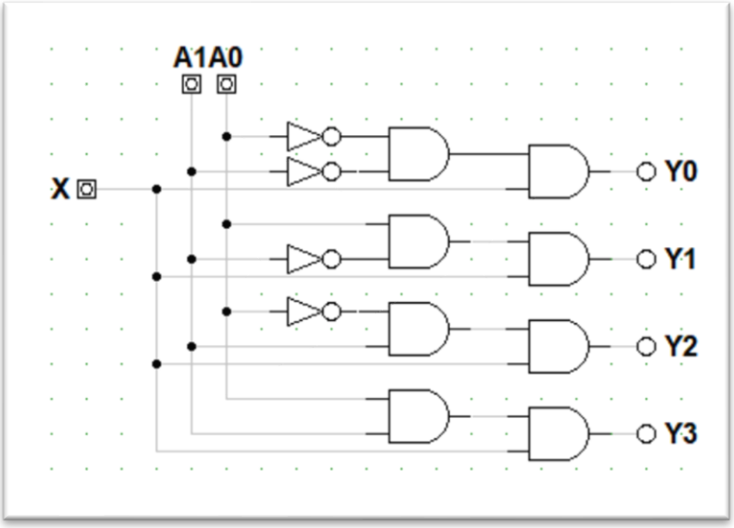
Demultiplexer

Demultiplexer adalah suatu rangkaian kombinasional yang mendistribusikan *data input* untuk diteruskan pada *output* yang diinginkan menggunakan *selector* (yang juga merupakan *input*). Rangkaian ini sering disebut data distributor dan disingkat demux. Demux memiliki dua jenis *input*, yaitu *data input* dan *selector input*. *Selector input* akan memilih *output* mana yang akan menerima informasi dari *data input*, sedangkan *data input* sendiri membawa informasi yang akan diteruskan (seperti apakah berupa low logic atau high logic). *Output* yang terpilih oleh *selector* akan menampilkan kondisi logika dari *data input*.

Demux memiliki ciri khas yaitu hanya memiliki satu *data input*. Pada beberapa demux, juga terdapat *input* tambahan berupa enabler *input*, yaitu *input* yang menentukan apakah demux beroperasi atau tidak beroperasi (seperti tombol on/off). Beberapa jenis demux antara lain 1x2 Demultiplexer, 1x4 Demultiplexer, 1x8 Demultiplexer, dan 1x16 Demultiplexer.

Terdapat hubungan antara jumlah *data input* dan *selector input*, yaitu $m = 2^s$ dengan *m* adalah jumlah *output* dan *s* adalah jumlah *selector input*. Maka dari itu, jumlah *output* dari demultiplexer selalu merupakan hasil dari perpangkatan 2.

Contoh: *Demultiplexer* 1x4



Tabel kebenaran:

A1	A0	X	Y0	Y1	Y2	Y3
0	0	0	0	×	×	×
0	0	1	1	×	×	×
0	1	0	×	0	×	×
0	1	1	×	1	×	×
1	0	0	×	×	0	×
1	0	1	×	×	1	×
1	1	0	×	×	×	0
1	1	1	×	×	×	1

Pada rangkaian ini, A0 dan A1 adalah *selector input*, X adalah *data input*, serta Y0, Y1, Y2, dan Y3 adalah *output*. Ketika suatu *output* terpilih oleh *selector*, maka *output* tersebut akan menampilkan kondisi logika dari *data input*, sedangkan kondisi logika *output* lain yang tidak terpilih adalah tidak dipedulikan/keadaan *don't care* (ditandai dengan ×). Pada saat keadaan *don't care*, keadaan logika *output* tersebut secara *default*/bawaan adalah berlogika 0.

Materi 6: FLIP-FLOP

Rangkain Sekuensial

Rangkaian sekuensial merupakan rangkaian kombinasional yang dipadukan dengan unit *Memory*. *Output* selanjutnya (Qn+1) dari rangkaian sekuensial dipengaruhi oleh keadaan-keadaan *input* dan keadaan-keadaan *output* pada saat yang sama (Qn). Unit *Memory* menyimpan keadaan-keadaan *output* saat itu (Qn) lalu menyalurkannya kembali menjadi *input* dari rangkaian sekuensial (disebut *feedback*).

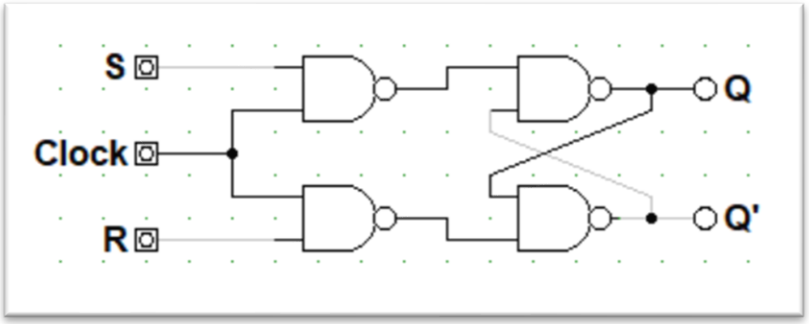
Rangkaian sekuensial dapat bekerja secara sinkronus maupun asynchronous. Sinkronus berarti *input* disalurkan masuk ke rangkaian secara serentak/waktu yang sinkron sedangkan asinkronusberarti *input* disalurkan masuk ke rangkaian secara independen/waktu yang tidak sinkron. Cara kerja asinkronus lebih sering dihindari karena berpotensi terjadinya perubahan *input* yang tidak disengaja (yang mana langsung mengubah keadaan *output*) dan sukar dicari kesalahannya. Cara kerja sinkronus memiliki *input* tambahan yaitu *control input* yang bisa berupa Enable maupun Clock. Pada saat *control input* berlogika rendah, maka tidak akan terjadi perubahan pada *output* walaupun keadaan *input* berubah-ubah.

Flip-Flop

Flip – Flop atau FF adalah rangkaian sekuensial yang dapat menyimpan informasi sebesar 1 bit sehingga FF adalah unit *Memory* terkecil. *Output* dari FF berjumlah dua yang keduanya saling berkebalikan/komplementer yaitu Q dan Q'. FF memiliki *output* "*Memory*" yang berarti rangkaian akan menampilkan *output* sebelumnya.

Flip-Flop RS Clocked

Flip – Flop RS Clocked atau FF RSC adalah FF sinkron yang merupakan dasar dari semua jenis FF. Rangkaian ini terdiri dari *input* Set (S) dan Reset (R). Berikut rangkaian logika dan tabel kebenarannya

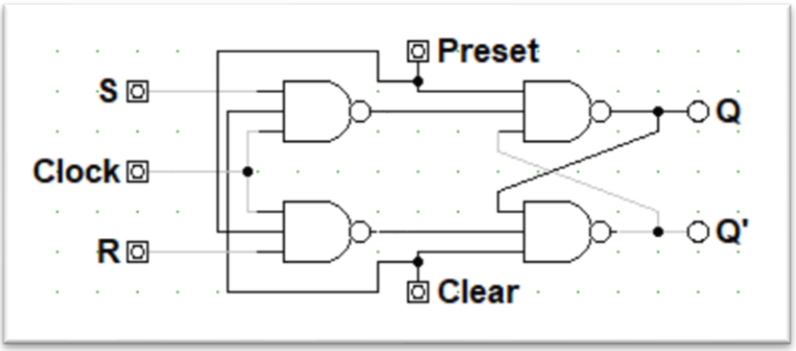


Clock	S	R	Q	Q'
0	×	×	*Memory*	
1	0	0	*Memory*	
1	0	1	0	1
1	1	0	1	0
1	1	1	*Not Used*	

FF RSC memiliki kelemahan yaitu saat S = R = 1 maka Q dan Q' bernilai 1. Hal ini tentu bertentangan terhadap definisi dari Q dan Q' yaitu saling berkebalikan. Maka dari itu, keadaan *input* tersebut tidak dipakai (*Not Used)*.

Flip-Flop RS Clocked* dengan *Preset* dan *Clear

Flip – Flop ini merupakan modifikasi dari FF RSC dimana ditambakan *input* Preset dan Clear. Kedua *input* tersebut bekerja secara asinkronus terhadap *output* saat itu sehingga tidak terpengaruh oleh clock. Modifikasi ini bertujuan supaya kita dapat mengubah terlebih dahulu keadaan *output* sebelum dioperasikan pada FF RSC. Berikut rangkaian logika dan tabel kebenarannya.



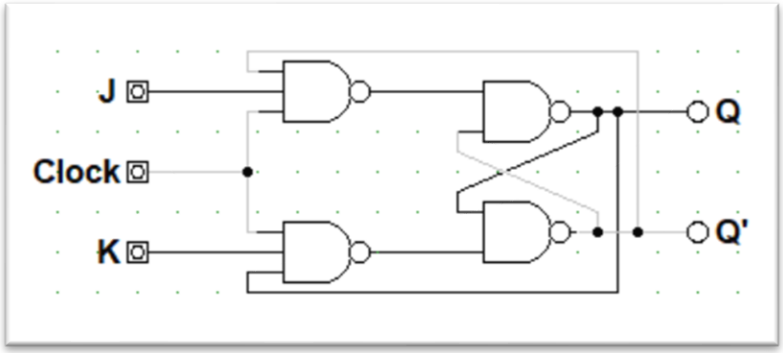
Preset	Clear	Clock	S	R	Q	Q’
1	1	0	×	×	*Memory*	
1	1	1	0	0	*Memory*	
1	1	1	0	1	0	1
1	1	1	1	0	1	0
1	1	1	1	1	*Not Used*	

Preset	Clear	Clock	S	R	Q	Q’
0	1	×	×	×	1	0
1	0	×	×	×	0	1
0	0	×	×	×	*Not Used*	

Preset dan Clear bekerja secara inverted. Keadaan Preset dan Clear harus dikembalikan menjadi 11 supaya FF RSC bisa dioperasikan. Seperti pada FF RSC, keadaan 00 perlu dihindari karna akan mengakibatkan keadaan Q dan Q’ bernilai 1.

Flip-Flop JK

Flip – Flop JK atau FF JK adalah modifikasi dari FF RSC dengan tujuan supaya membuat *input* S=R=1 dapat digunakan. *Input* S berubah menjadi J dan *input* R berubah menjadi K. Di FF ini, saat J=K=1, akan terjadi *toggle* yaitu keluaran menampilkan komplemen dari memori (Qn’). Keadaan *toggle* ini hanya akan terjadi bila *Clock On’s time* lebih besar daripada *Flip – Flop’s propagation delay*. Berikut rangkaian logika dan tabel kebenarannya.

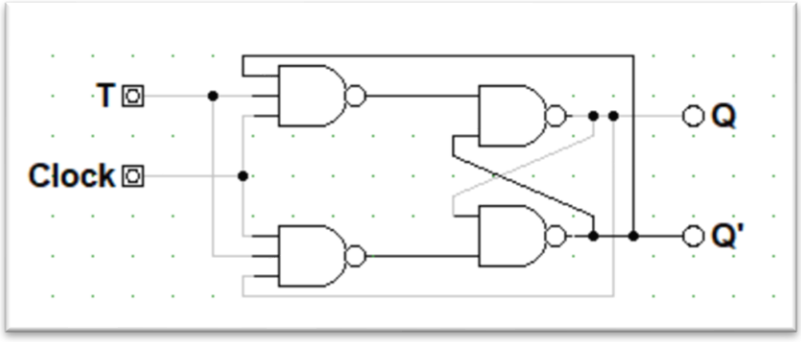


Clock	J	K	Q	Q’
0	×	×	*Memory*	
1	0	0	*Memory*	
1	0	1	0	1
1	1	0	1	0
1	1	1	*Toggle**	

Toggle= Toggle belum sempurna*

Flip-Flop T

Flip – Flop T atau FF T adalah modifikasi dari FF JK dimana hanya digunakan satu *input* yang lalu dihubungkan ke setiap *terminal input* dari J dan K. Dengan ini didapat FF JK yang hanya memiliki *input* 00 dan 11. Fungsi dari FF T adalah sebagai rangkaian *toggle*. Seperti pada FF JK, keadaan *toggle* ini hanya akan terjadi bila *Clock On’s time* lebih besar daripada *Flip – Flop’s propagation delay*. T dari nama FF ini berarti “*Toggle*”. Berikut rangkaian logika dan tabel kebenarannya.

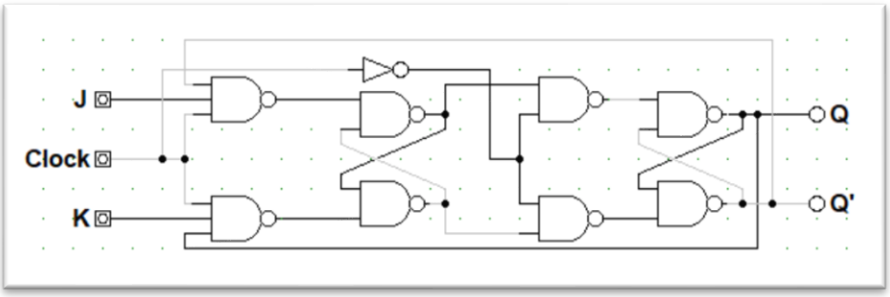


Clock	T	Q	Q’
0	×	*Memory*	
1	0	*Memory*	
1	1	*Toggle**	

Toggle= Toggle belum sempurna*

Flip-Flop JK Master-Slave

Flip – Flop JK Master-Slave atau FF JKMS adalah modifikasi dari FF JK. Seperti yang sudah dijelaskan diatas, keadaan *toggle* hanya akan terjadi bila *Clock On’s time* lebih besar daripada *Flip – Flop’s propagation delay*. Pada praksisnya, hal tersebut sukar dilakukan. Jika tidak memenuhi syarat tersebut, maka keluaran akan menjadi *race around*, yaitu keadaan di mana *output* Q dan Q’ akan terus menerus bernilai 01, 10, 10 10, … sampai clock bernilai 0. Keadaan *race around* juga bisa disebut “*toggle* yang tak terkendali’. Maka dari itu, alternatif lain supaya keadaan *toggle* dapat bekerja adalah dengan menggunakan modifikasi Master-Slave. Berikut rangkaian logika dan tabel kebenarannya.

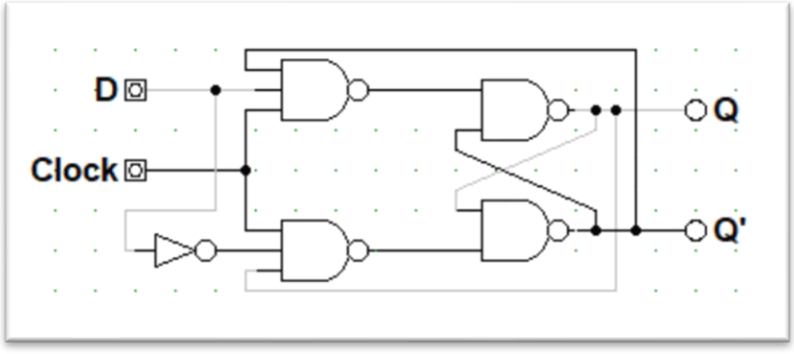


Clock	J	K	Q
0	×	×	*Memory*
1	0	0	*Memory*
1	0	1	0
1	1	0	1
1	1	1	*Toggle*

Master-Slave atau Tuan-Budak adalah modifikasi dimana dibuat dua FF yang mirip lalu ditentukan FF yang sebagai *master* dan FF yang sebagai *slave*. FF master hanya akan bekerja bila clock bernilai 1 sedangkan FF slave akan bekerja bila clock bernilai 0. Modifikasi Master-Slave membuat rangkaian menjadi bersifat *negative-edge triggered*. Dengan itu, keadaan *race around* bisa diatasi.

Flip-Flop D

Flip – Flop D atau FF D adalah modifikasi dari FF JK dimana hanya digunakan satu *input* yang akan disambungkan ke *terminal input* J secara normal dan *terminal input* K secara berkebalikan. Dengan ini didapat FF JK yang hanya memiliki *input* 01 dan 10 (yang menjadi *input* D = 0 dan D = 1). D pada FF ini adalah singkatan dari “Data”. Berikut rangkaian logika dan tabel kebenarannya.



Clock	D	Q	Q’
0	×	*Memory*	
1	0	0	1
1	1	1	0

Materi 8: COUNTER

Counter

Counter atau pencacah adalah rangkaian sekuensial memiliki fungsi untuk melakukan penghitungan/pencacahan berdasarkan banyaknya *pulse* dari clock. Karena *counter* akan menghitung sesuatu secara bertahap/sekuensial (yaitu berdasarkan keadaan jumlah *pulse* yang diberikan), maka diperlukan rangkaian memori untuk menyimpan informasi jumlah *pulse* sehingga *counter* adalah terdiri dari satu atau lebih rangkaian flip-flop. Flip-flop yang biasa digunakan yaitu JK Flip-flop.

Hampir seluruh peralatan elektronik yang menerapkan sistem digital memiliki unsur *counter* di dalamnya. Kegunaan *counter* secara umum ialah:

- Operasi aritmetika
- Pembagi frekuensi
- Pengurut alamat

Counter dapat memiliki rentang clock yang beragam sesuai dari bagaimana *counter* didesain. Jumlah clock yang sering dipakai adalah 2^n yaitu 0-3, 0-7, 0-15, dst. tetapi kita juga bisa merancang s*up*aya clock dapat menghitung diluar dari 2^n seperti 0-2 dan 2-6.

Clock pada *counter* bekerja secara *negative trigger/falling edge* sehingga perubahan *output* terjadi saat keadaan clock berubah dari *high* (1) menjadi *low* (0).

Jenis *Counter* Berdasarkan Distribusi *Pulse*

Counter dibagi menjadi dua yaitu asinkronus dan sinkronus. *Counter* asinkronus atau sering disebut *ripple counter* adalah *counter* yang clocknya hanya disambung pada Flip-flop LSB/urutan pertama, yang kemudian *output* dari Flip-flop tersebut akan menjadi clock untuk Flip-flop berikutnya. Dengan ini, *pulse* dari clock disebarkan secara merambat dan tidak serentak. *Counter* sinkronus adalah *counter* yang clocknya disambung ke setiap Flip-flop yang digunakan sehingga *pulse* dari clock disebarkan secara serentak ke setiap Flip-flop.

Secara praksis, *counter* asinkronus memiliki *propagatin delay* yang lebih lama daripada *counter* sinkronus. Hal ini dikarenakan pada *counter* asinkronus, *pulse* harus bergerak dari Flip-flop pertama, lalu Flip-flop kedua, lalu Flip-flop ketiga, dst. Sedangkan pada *counter* sinkronus, *pulse* bergerak secara serentak dan langsung ke setiap Flip-flop. Pada rangkaian digital yang semakin besar dan kompleks, kekurangan dari *counter* asinkronus akan dapat dirasakan.

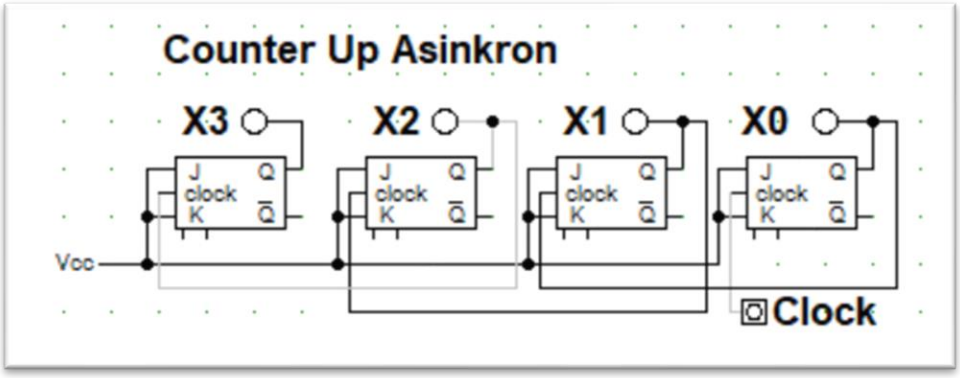
Jenis *Counter* Berdasarkan Urutan Hitung

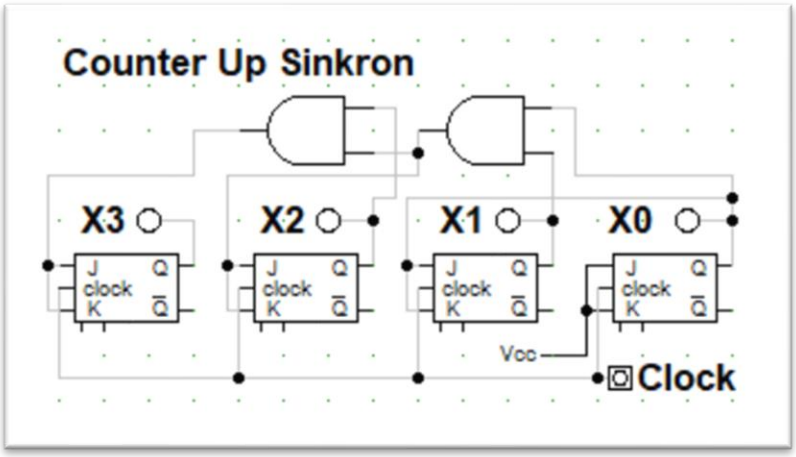
Counter dibagi menjadi *up counter*/*counter* naik dan *down counter*/*counter* turun. *Up counter* adalah *counter* yang melakukan penghitungan dari kecil ke besar. Contohnya pada *up counter* 2 Bit maka akan dilakukan penghitungan 0, 1, 2, dan 3. *Down counter* adalah *counter* yang melakukan penghitungan dari besar ke kecil. Contohnya pada *down counter* 2 Bit maka akan dilakukan penghitungan 3, 2, 1, dan 0. Penggunaan *up* dan *down counter* ini sesuai dengan kebutuhan.

Contoh-contoh *Counter*

Counter Up 4-bit: Asinkron dan Sinkron

Rangkaian logika:



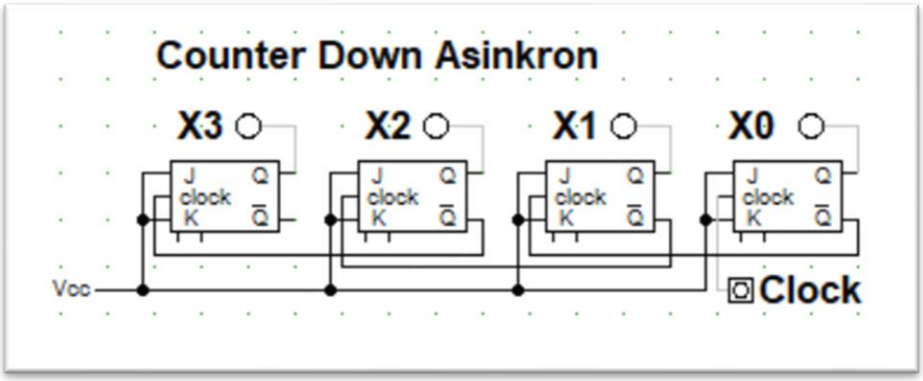


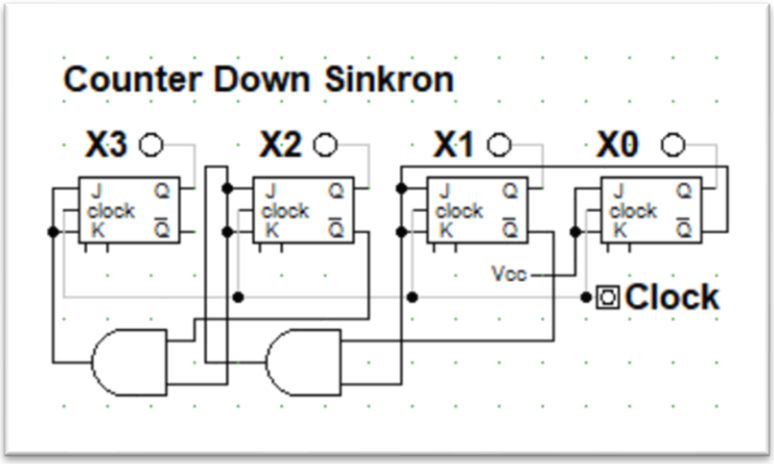
Tabel kebenaran:

Pulse	Mod-16	X3	X2	X1	X0	Desimal
0	0	0	0	0	0	0
1	1	0	0	0	1	1
2	2	0	0	1	0	2
3	3	0	0	1	1	3
4	4	0	1	0	0	4
5	5	0	1	0	1	5
6	6	0	1	1	0	6
7	7	0	1	1	1	7
8	8	1	0	0	0	8
9	9	1	0	0	1	9
10	10	1	0	1	0	10
11	11	1	0	1	1	11
12	12	1	1	0	0	12
13	13	1	1	0	1	13
14	14	1	1	1	0	14
15	15	1	1	1	1	15
16	0	0	0	0	0	0

Counter Down 4-bit: Asinkron dan Sinkron

Rangkaian logika:





Tabel kebenaran:

Pulse	Mod-16	X3	X2	X1	X0	Desimal
0	0	0	0	0	0	0
1	1	1	1	1	1	15
2	2	1	1	1	0	14
3	3	1	1	0	1	13
4	4	1	1	0	0	12
5	5	1	0	1	1	11
6	6	1	0	1	0	10
7	7	1	0	0	1	9
8	8	1	0	0	0	8
9	9	0	1	1	1	7
10	10	0	1	1	0	6
11	11	0	1	0	1	5
12	12	0	1	0	0	4
13	13	0	0	1	1	3
14	14	0	0	1	0	2
15	15	0	0	0	1	1
16	0	0	0	0	0	0

Materi 9: SHIFT REGISTER

Register

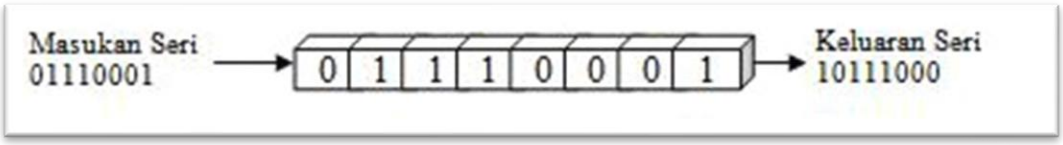
Register adalah rangkaian sekuensial yang dapat menyimpan data dalam bentuk *binary* dalam jumlah yang banyak. *Register* terdiri dari susunan Flip-flop JK atau Flip-flop D. Cara kerja *register* ialah menyusun secara khusus Flip-Flop tadi sehingga sifat Flip-flop yang hanya dapat menyimpan data 1 Bit dapat menyimpan data yang lebih banyak lagi. Jumlah bit yang dapat disimpan sama dengan jumlah Flip-flop yang digunakan.

Shift Register

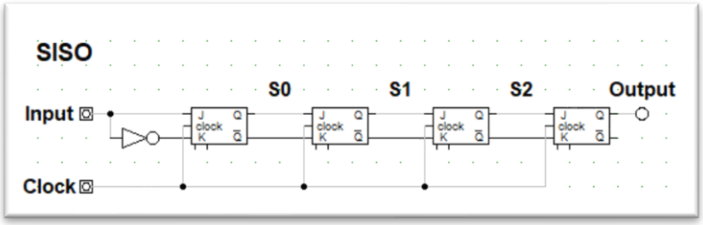
Shift register adalah jenis dari *register* di mana ia dapat menyimpan dan *shifting*/menggeser data. *Shift* ini dapat dilakukan ke arah LSB (kanan) atau ke arah MSB (kiri). Penggeseran data dilakukan menggunakan *clock*. Setiap kali *pulse* dilakukan maka data akan bergeser. Fungsi geser yang dimiliki *shift register* dimanfaatkan untuk menyederhanakan rangkaian logika. *Register* ini juga dapat digunakan untuk mengubah format data dari seri ke paralel atau dari paralel ke seri.

***Shift Register: Serial In Serial Out* (SISO)**

Rangkaian ini adalah jenis dari *Shift Register* yang mana *input* masuk secara seri dan *output* ditampilkan secara seri pula. *Output* yang *register* adalah *output* dari Flip-flop terakhir. Jenis SISO ini tidak mengubah format data



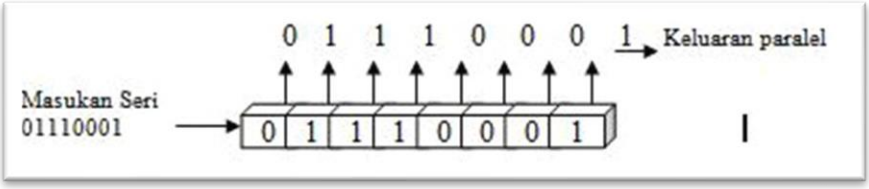
Dikarenakan *input* masuk secara seri, maka data masuk secara satu per satu dan *clock pulse* digunakan untuk mengijinkan data masuk sekaligus melakukan *shifting*. Contoh: jika ingin memasukkan data 1011 pada *Shift Register* SISO 4 bit, maka dilakukan *input* 1 → *pulse* → *input* 1 → *pulse* → *input* 0 → *pulse* → *input* 1 → *pulse*. *Clock* pada SISO bekerja secara *falling edge*. Berikut rangkaian logika dan tabel kebenarannya.



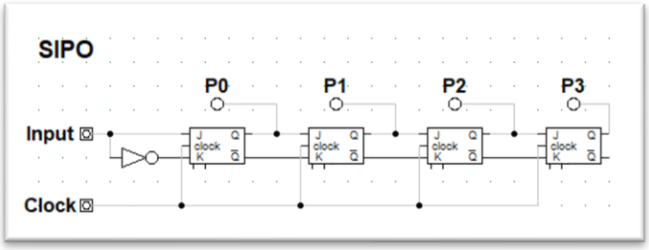
Pulse	*Input*	S0	S1	S2	*Output*
0	#1	*Initial*	*Initial*	*Initial*	*Initial*
1	#2	#1	*Initial*	*Initial*	*Initial*
2	#3	#2	#1	*Initial*	*Initial*
3	#4	#3	#2	#1	*Initial*
4	#5	#4	#3	#2	#1
5	#6	#5	#4	#3	#2

***Shift Register: Serial In Paralel Out* (SIPO)**

Rangkaian ini adalah jenis dari *Shift Register* yang mana *input* masuk secara seri dan *output* ditampilkan secara paralel. Maka dari itu, setiap Flip-flop ditampilkan *output*nya. Jenis SIPO ini mengubah format data dari seri ke paralel.



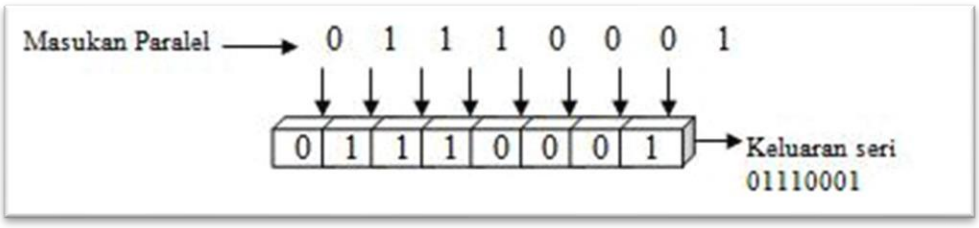
Sama seperti jenis SISO, dikarenakan *input* masuk secara seri, maka data masuk secara satu per satu dan *clock pulse* digunakan untuk mengijinkan data masuk sekaligus melakukan *shifting*. *Clock* pada SIPO bekerja secara *falling edge*. Berikut rangkaian logika dan tabel kebenarannya.



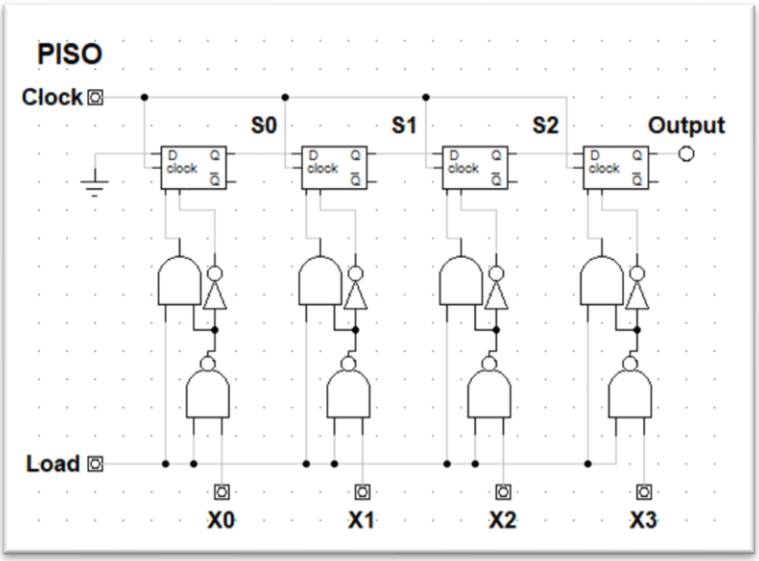
Pulse	*Input*	P0	P1	P2	P3
0	#1	*Initial*	*Initial*	*Initial*	*Initial*
1	#2	#1	*Initial*	*Initial*	*Initial*
2	#3	#2	#1	*Initial*	*Initial*
3	#4	#3	#2	#1	*Initial*
4	#5	#4	#3	#2	#1
5	#6	#5	#4	#3	#2

Shift Register: Paralel In Serial Out* (PISO) dengan *Load

Rangkaian ini adalah jenis dari *Shift Register* yang mana *input* masuk secara paralel dan *output* ditampilkan secara seri. Maka dari itu, seluruh data dapat secara bersamaan dan dimasukkan dalam *register*. Jenis SIPO ini mengubah format data dari paralel ke seri. Terdapat input Load yang berfungsi supaya input masuk ke register secara serentak. Saat Load berlogika 0, input-input ditahan untuk masuk ke register. Saat Load berlogika 1, input-input dapat masuk ke register. Clock hanya bisa digunakan saat Load berlogika 0



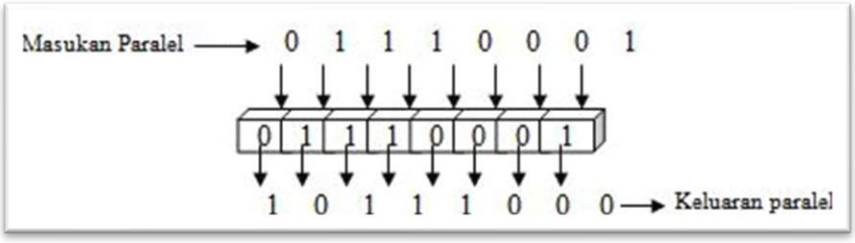
Register ini tetap dapat melakukan *shift* menggunakan *clock* tetapi *input* yang baru akibat pergeseran adalah 0 secara konstan. Contoh: saat keadaan *register* 1011 lalu dilakukan *shift*, maka hasilnya ialah 0101. *Clock* pada PISO bekerja secara *leading edge*. Berikut rangkaian logika dan tabel kebenarannya.



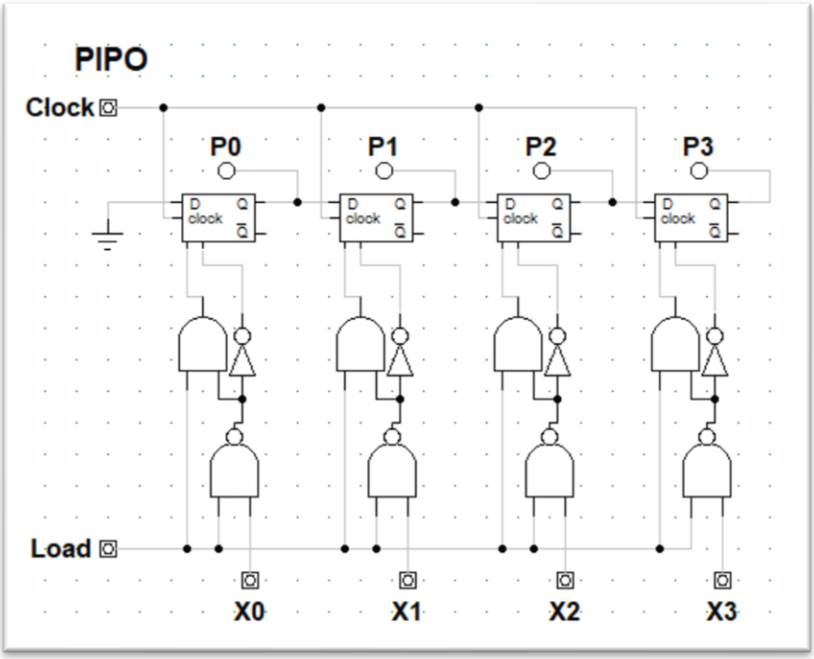
Pulse	*Load*	X0	X1	X2	X3	S0	S1	S2	*Output*
0	0	X	X	X	X	*Initial*	*Initial*	*Initial*	*Initial*
0	1	#1	#2	#3	#4	#1	#2	#3	#4
1	0	X	X	X	X	0	#1	#2	#3
2	0	X	X	X	X	0	0	#1	#2
3	0	X	X	X	X	0	0	0	#1
4	0	X	X	X	X	0	0	0	0
5	0	X	X	X	X	0	0	0	0

Shift Register: Paralel In Paralel Out* (PIPO) dengan *Load

Rangkaian ini adalah jenis dari *Shift Register* yang mana *input* masuk secara paralel dan *output* ditampilkan secara paralel pula. Maka dari itu, seluruh data dapat secara bersamaan dan dimasukkan dalam *register*. Jenis PIPO ini tidak mengubah format data. Terdapat input Load yang berfungsi supaya input masuk ke register secara serentak. Saat Load berlogika 0, input-input ditahan untuk masuk ke register. Saat Load berlogika 1, input-input dapat masuk ke register. Clock hanya bisa digunakan saat Load berlogika 0



Sama seperti PISO, *register* ini tetap dapat melakukan *shift* menggunakan *clock* tetapi *input* yang baru akibat pergeseran adalah 0 secara konstan. *Clock* pada PIPO bekerja secara *leading edge*. Berikut rangkaian logika dan tabel kebenarannya.



Pulse	*Load*	X0	X1	X2	X3	P0	P1	P2	P3
0	0	X	X	X	X	*Initial*	*Initial*	*Initial*	*Initial*
0	1	#1	#2	#3	#4	#1	#2	#3	#4
1	0	X	X	X	X	0	#1	#2	#3
2	0	X	X	X	X	0	0	#1	#2
3	0	X	X	X	X	0	0	0	#1
4	0	X	X	X	X	0	0	0	0
5	0	X	X	X	X	0	0	0	0

Materi 10: PENGENALAN ARDUINO

Arduino secara umum

Arduino adalah sebuah minimum sistem mikrokontroler bersifat open-source yang banyak digunakan untuk membagun sebuah project elektronika. Platform Arduino berisi dua yaitu hardware berupa board dan sebuah software atau IDE (Integrated Development Environment) yang berjalan pada komputer, digunakan untuk menulis dan mengisikan perogram ke board Arduino.

Sejarah singkat

Konsep awal Arduino lahir dari tesis Hernando Barragan berjudul "*Arduino-La rivoluzione dell'open Hardware*" atau dalam bahasa inggrisnya "*Arduino – The Revolution of Open Hardware*". Berlandaskan tesis Hernando ini, Massimo Banzi berharap bisa membuat suatu platform yang jauh lebih murah, sederhana dan mudah untuk digunakan dibandingkan *Basic Stamp* yang merupakan mikrokontroler yang umum sebelum adanya Arduino.

Massimo Banzi meneruskan ambisinya dengan mulai mengembangkan Arduino. Tim inti pengembangan Arduino terdiri dari Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, dan David Mellis. Mereka mengupayakan 4 hal dalam Arduino ini, yaitu:

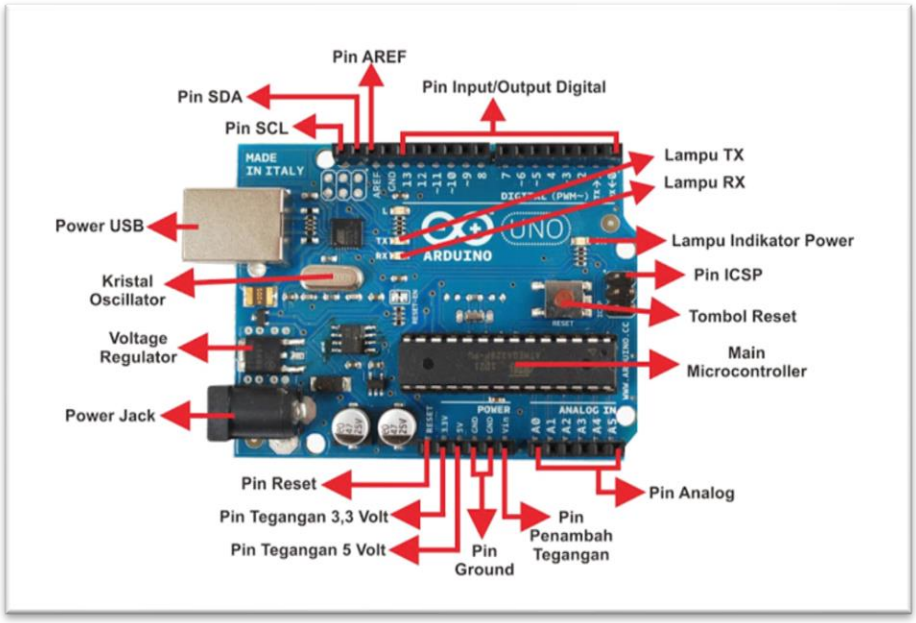
1. Harga terjangkau
2. Dapat dijalankan diberbagai sistem operasi, Windows, Linux, Mac, dan sebagainya.
3. Sederhana, dengan bahasa pemograman yang mudah bisa dipelajari orang awam
4. Open Source, hardware maupun software.

Banzi dan rekan-rekan berpikir bahwa akan jauh lebih baik bila platform Arduino ini dibuat open source agar proses pengembangannya bisa dilakukan secara cepat. Bukti perkembangan Arduino dapat kita rasakan sekarang

Untuk sejarah nama Arduino sendiri diambil dari nama bar di kota Ivrea, Italia Utara yang sering dikunjungi oleh para pengembang Arduino, yaitu "Bar Di Re Arduino" yang artinya Bar Raja Arduino. Arduin sendiri adalah nama raja yang pernah memerintah kota Ivrea pada tahun 1002 Masehi.

Bagian-bagian pada Arduino

Berikut adalah nama-nama bagian Arduino serta penjelasannya (dengan Arduino Uno sebagai model)



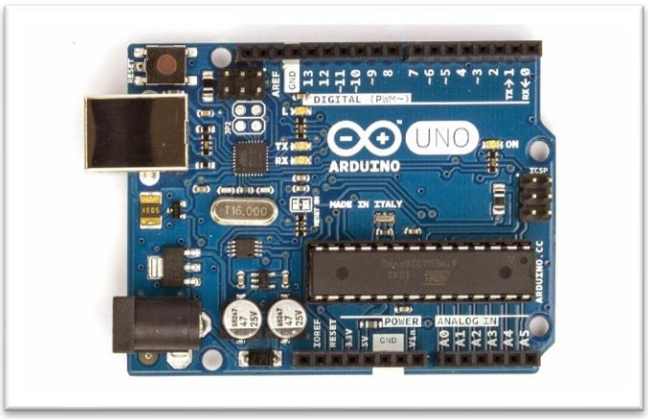
1. Power USB, berfungsi dalam 3 hal yaitu:
    - Media pemberi tegangan listrik ke Arduino
    - Media tempat memasukkan program dari komputer ke Arduino
    - Sebagai media untuk komunikasi serial antara komputer dan Arduino R3 maupun sebaliknya.

2. Kristal Oscillator, berfungsi sebagai jantung Arduino yang membuat dan mengirimkan detak ke mikrokontroler agar beroperasi setiap detaknya.
3. Voltage Regulator, berfungsi menstabilkan tegangan listrik yang masuk ke Arduino.
4. Power Jack, berfungsi sebagai media pemberi tegangan listrik ke Arduino apabila tak ingin menggunakan Power USB.
5. Pin Reset, berfungsi untuk mereset Arduino agar program dimulai dari awal. Cara penggunannya yaitu dengan menghubungkan pin reset ini langsung ke ground.
6. Pin Tegangan 3,3 Volt, berfungsi sebagai pin positif untuk komponen yang menggunakan tegangan 3,3 volt.
7. Pin Tegangan 5 Volt, berfungsi sebagai pin positif untuk komponen yang menggunakan tegangan 5 volt. Pin 5 volt sering juga disebut pin VCC.
8. Pin Ground (GND), berfungsi sebagai pin negatif pada tiap komponen yang dihubungkan ke Arduino.
9. Pin Penambah Tegangan (VIN), berfungsi sebagai media pemasok listrik tambahan dari luar sebesar 5 volt bila tak ingin menggunakan Power USB atau Power Jack.
10. Pin Analog, berfungsi membaca tegangan dan sinyal analog dari berbagai jenis sensor untuk diubah ke nilai digital
11. Main Microcontroller, berfungsi sebagai otak yang mengatur pin-pin pada Arduino.
12. Tombol Reset, berfungsi untuk mengulang program dari awal dengan cara menekan tombol.
13. Pin ICSP (In-Circuit Serial Programming), berfungsi untuk memprogram mikrokontroler seperti Atmega328 melalui jalur USB Atmega16U2.
14. Lampu Indikator Power, berfungsi sebagai indikator bahwa Arduino sudah mendapatkan suplai tegangan listrik yang baik.
15. Lampu TX (transmit), berfungsi sebagai penanda bahwa sedang terjadi pengiriman data dalam komunikasi serial.
16. Lampu RX (receive), berfungsi sebagai penanda bahwa sedang terjadi penerimaan data dalam komunikasi serial.

17. Pin Input/Output Digital, berfungsi untuk membaca nilai logika 1 dan 0 atau mengendalikan komponen output lain seperti LED, relay, atau sejenisnya. Pin ini termasuk paling banyak digunakan saat membuat rangkaian.
    Untuk pin yang berlambang "~" artinya dapat digunakan untuk membangkitkan PWM (Pulse With Modulation) yang fungsinya bisa mengatur tegangan output. Biasanya digunakan untuk mengatur kecepatan kipas atau mengatur terangnya cahaya lampu.
18. Pin AREF (Analog Reference), berfungsi mengatur tegangan referensi eksternal yang biasanya berada di kisaran 0 sampai 5 volt.
19. Pin SDA (Serial Data), berfungsi untuk menghantarkan data dari modul I2C atau yang sejenisnya.
20. Pin SCL (Serial Clock), berfungsi untuk menghantarkan sinyal waktu (clock) dari modul I2C ke Arduino

Jenis-jenis Arduino

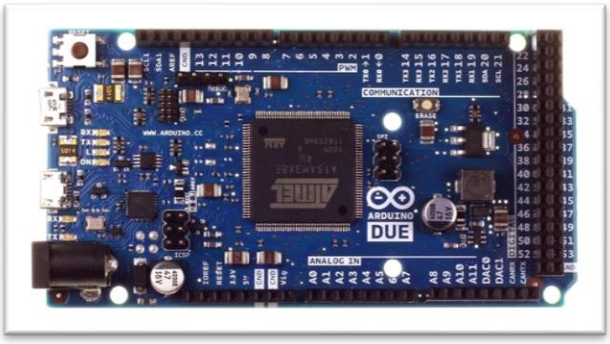
1. Arduino Uno



Ialah Arduino yang paling populer dan ramah pemula. Menggunakan USB type A to B
Spesifikasi:

Microcontroller	ATMEGA328P
Digital Input/Output pins	14 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	6
Flash Memory	32 KB dengan 0,5 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	68,6 mm x 53,4 mm
Berat	25 g

2. Arduino Due

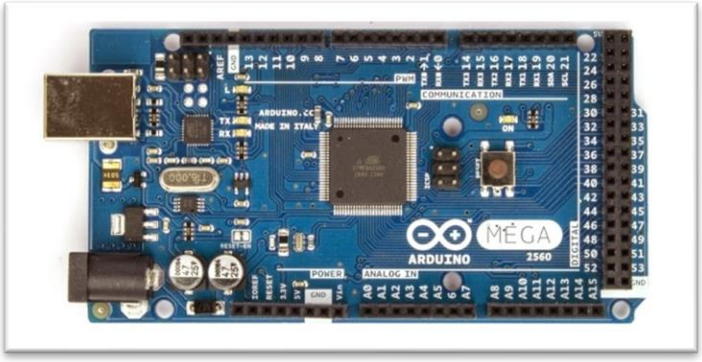


Ialah Arduino yang menggunakan chip yang lebih tinggi yaitu ARM Cortex CPU. Menggunakan micro USB

Spesifikasi:

Microcontroller	AT91SAM3X8E
Digital Input/Output pins	54 (dengan menyediakan 12 PWM Input/Output)
Analog Input pins	12
Flash Memory	512 KB
Kecepatan clock	84 MHz
Dimensi (panjang x lebar)	101,52 mm x 53,3 mm
Berat	36 g

3. Arduino Mega



Ialah Arduino yang mirip dengan Uno namun dengan chip lebih baik dan pin I/O lebih banyak. Menggunakan USB type A to B
Spesifikasi:

Microcontroller	ATMEGA2560
Digital Input/Output pins	54 (dengan menyediakan 15 PWM Input/Output)
Analog Input pins	16
Flash Memory	256 KB dengan 8 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	101,52 mm x 53,3 mm
Berat	37 g

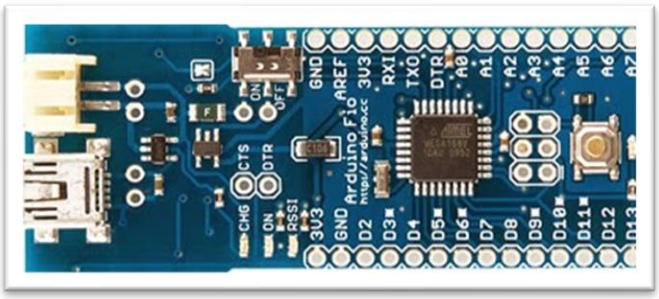
4. Arduino Leonardo

Ialah Arduino yang bisa disebut saudara kembar dari Uno. Menggunakan micro USB

Spesifikasi:

Microcontroller	ATMEGA32U4
Digital Input/Output pins	20 (dengan menyediakan 7 PWM Input/Output)
Analog Input pins	12
Flash Memory	32 KB dengan 4 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	68,6 mm x 53,3 mm
Berat	20 g

5. Arduino Fio

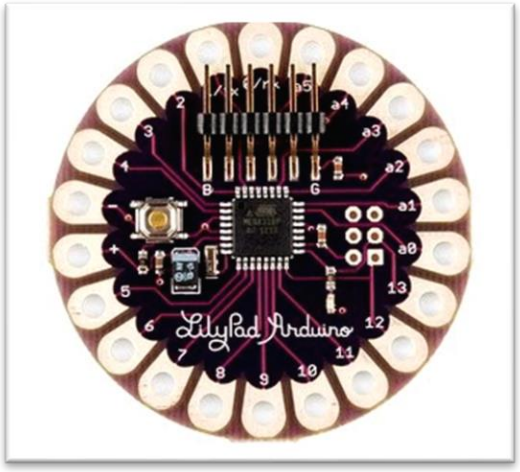


Ialah Arduino yang memiliki Socket XBee yang diperkenankan untuk proyek nirkabel

Spesifikasi:

Microcontroller	ATMEGA328P
Digital Input/Output pins	14 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	8
Flash Memory	32 KB dengan 2 KB digunakan oleh bootloader
Kecepatan clock	8 MHz
Dimensi (panjang x lebar)	28 mm x 65 mm
Berat	9 g

6. Arduino Lilypad

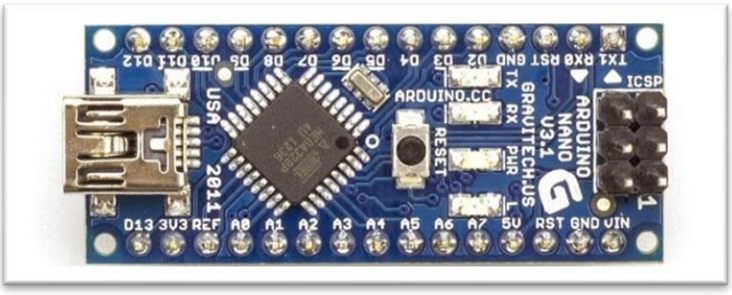


Ialah Arduino yang memiliki bentuk unik yaitu lingkaran. Menggunakan micro USB

Spesifikasi:

Microcontroller	ATMEGA168 atau ATMEGA328V
Digital Input/Output pins	14 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	6
Flash Memory	16 KB dengan 2 KB digunakan oleh bootloader
Kecepatan clock	8 MHz
Dimensi (diameter)	50 mm
Berat	8 g

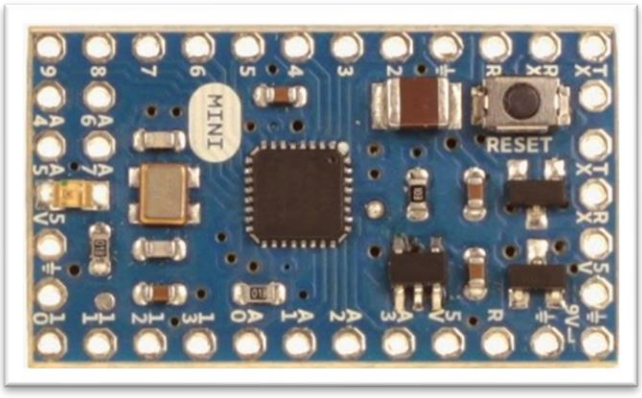
7. Arduino Nano



Ialah Arduino yang berukuran kecil namun memiliki banyak fasilitas. Menggunakan micro USB
Spesifikasi:

Microcontroller	ATMEGA168 atau ATMEGA328
Digital Input/Output pins	14 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	8
Flash Memory	16 KB (ATMEGA168) atau 32 KB (ATMEGA328) dengan 2 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (Panjang x lebar)	18,5 mm x 43 mm
Berat	7 g

8. Arduino Pro Mini

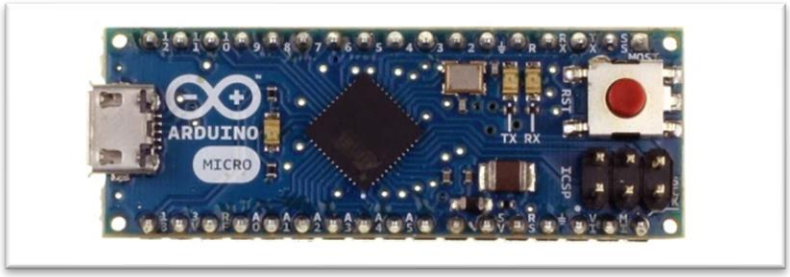


Ialah Arduino yang mirip dengan Nano namun tidak memiliki USB Adapter

Spesifikasi:

Microcontroller	ATMEGA168 atau ATMEGA328
Digital Input/Output pins	14 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	8
Flash Memory	16 KB (ATMEGA168) atau 32 KB (ATMEGA328) dengan 2 KB digunakan oleh bootloader
Kecepatan clock	8 MHz (model 3V) atau 16 MHz (model 5V)
Dimensi (Panjang x lebar)	17,8 mm x 33 mm
Berat	5,5 g

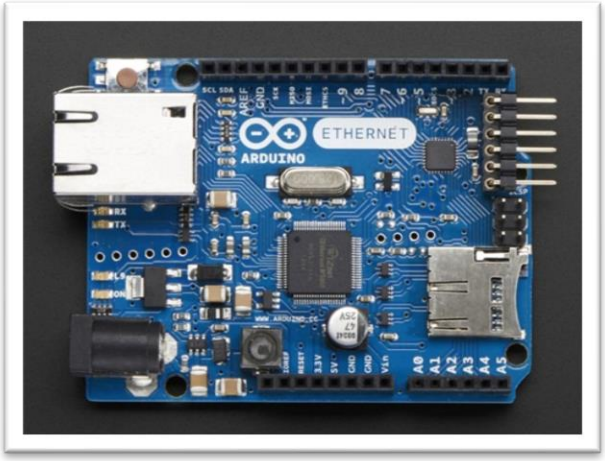
9. Arduino Mikro



Ialah Arduino mirip Nano dan Mini dengan fasilitas lebih banyak. Menggunakan micro USB
Spesifikasi:

Microcontroller	ATMEGA32U4
Digital Input/Output pins	20 (dengan menyediakan 7 PWM Input/Output)
Analog Input pins	12
Flash Memory	32 KB dengan 4 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	48 mm x 18 mm
Berat	13 g

10. Arduino Ethernet



Ialah Arduino yang sudah dilengakapi fasilitas ethernet.

Spesifikasi:

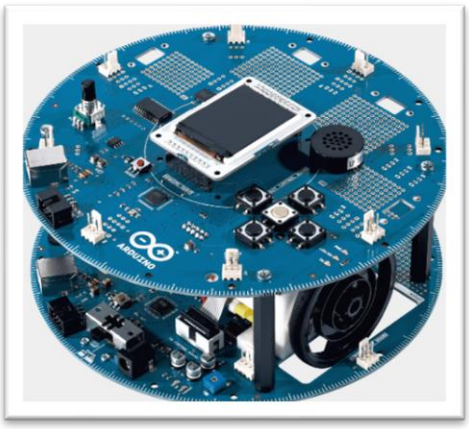
Microcontroller	ATMEGA328
Digital Input/Output pins	14 (dengan menyediakan 4 PWM Input/Output)
Analog Input pins	6
Flash Memory	32 KB dengan 0,5 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	68,6 mm x 53,3 mm
Berat	28 g

11. Arduino Esplora

Ialah Arduino yang sudah dilengkapi *joystick*, *button*, dan sebagainya, juga menyediakan sejumlah sensor *onboard built-in*, cocok untuk proyek yang berhubungan dengan *game*
Spesifikasi:

Microcontroller	ATMEGA32U4
Flash Memory	32 KB dengan 4 KB digunakan oleh bootloader
Kecepatan clock	16 MHz
Dimensi (panjang x lebar)	164,04 mm x 60 mm
Berat	53 g

12. Arduino Robot



Ialah Arduino dengan paket komplit yang sudah berbentuk robot. Arduino ini memfasilitasi setiap hal yang dibutuhkan pada robot seperti LCD, speaker, roda, sensor infrared, dan lain-lain.

Spesifikasi:

Control Board

Microcontroller	ATMEGA32U4
Digital Input/Output pins	5 (dengan menyediakan 6 PWM Input/Output)
Analog Input pins	4 dan 8 (ter-multiplexer)
Keypad	5
Flash Memory	32 KB dengan 4 KB digunakan oleh bootloader
Kecepatan clock	16 MHz

Motor Board

Microcontroller	ATMEGA32U4
Digital Input/Output pins	4 (dengan menyediakan 1 PWM Input/Output)
Analog Input pins	4
Flash Memory	32 KB dengan 4 KB digunakan oleh bootloader
Kecepatan clock	16 MHz

Secara keseluruhan Arduino Robot memiliki radius 185 mm dan tinggi 85 mm

Sensor

Adalah komponen yang digunakan untuk memberikan data ke Arduino. Sensor terbagi menjadi sensor digital dan analog. Sensor digital mengirimkan informasi dalam bentuk biner sedangkan sensor analog mengirimkan informasi dalam bentuk besaran yang kontinyu.

Berikut beberapa contoh sensor

1. Sensor Cahaya
2. Sensor Tekanan (mendeteksi ketegangan kawat)
3. Sensor Proximity (jarak)
4. Sensor Ultrasonik (menggunakan prinsip pantulan suara)
5. Sensor Kecepatan atau RPM (mengukur tegangan dari objek yang berputar)
6. Sensor Magnet
7. Sensor Penyandi (mengubah Gerakan linear atau radial menjadi sinyal digital)
8. Sensor Flow Meter (mengetahui flow dari suatu material liquid maupun solid)
9. Sensor Flame (mendeteksi nyala api dengan panjang gelombang 760 nm – 1100 nm)
10. Sensor Temperatur dan Kelembaban