BAB VII
Logika Sekuensial

7.1 Pendahuluan

Deskripsi singkat

Mempelajari bermacam jenis flip-flop, SR-FF, D-FF, JK-FF, T-FF sebagai elemen dari untai sekuensial, memori, dan register. Mempelajari diagram keadaan suatu flip-flop, dan diagram pewaktuannya.

Manfaat

Dengan mempelajari BAB 7, ini mahasiswa dapat memahami prinsip kerja berbagai jenis flip-flop, dan menyusun flip-flop dengan gerbang dasar NAND atau NOR. aplikasinya.

Relevansi

BAB 7 ini memberikan dasar-dasar untuk merancang dan menganalisis untai sekuensial dalam system digital, yang menjadi dasar pada system system pengolah/komputasi berteknologi digital modern saat ini. BAB 7 sangat relevan terhadap bab berikutnya, dan matakuliah lanjut seperti perancangan system digital, system mikroprosesor, arsitektur computer.

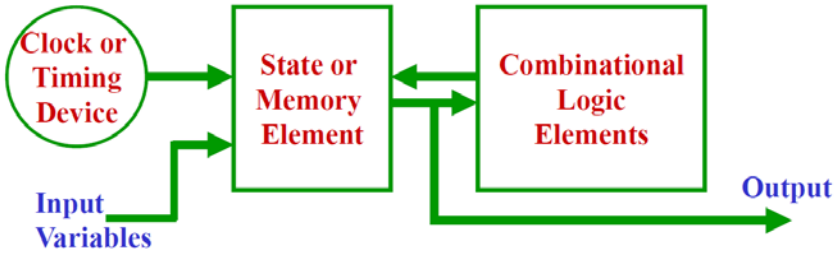
Learning Outcomes

- Mahasiswa mampu merancang untai berbagai jenis flip-flop.
- Mahasiswa mampu menganalisa (trouble shooting) untai flip-flop.
- Mahasiswa mampu merancang flip-flop, dan subtitusi atau konversi antar jenis flip-flop.

Kita telah mempelajari untai Logika Kombinasional, suatu untai logika kombinasional akan memiliki keadaan keluaran tergantung hanya dari keadaan inputnya. Sekarang kita alihkan perhatian pada *logika sekuensial*, yang merupakan bagian (porsi) terbesar dari untai digital pada sistem komputer komputer modern. Logika sekuensial berbeda dengan logika kombinasional dalam beberapa hal/cara yaitu:

- Keluarannya tergantung tidak hanya pada input tetapi juga pada logika keadaan internal (*internal state*).
- Keadaan output logika sekuensial tidak akan/harus berubah bila masukannya berubah, tetapi harus disinkronkan dengan suatu /beberapa sinyal pemicu (triggering event).
- Logika sekuensial sering disinkronkan (*synchronized*) atau dipicu (*triggered*) oleh deretan pulsa yang regular pada suatu jalur masukan serial, selanjutnya sinyal masukan ini disebut sebagai denyut atau ***clock***. Clock dibangkitkan oleh suatu piranti pewaktuan.

Presentasi Logika Sekuensial (*Sequential Logic Representation*)



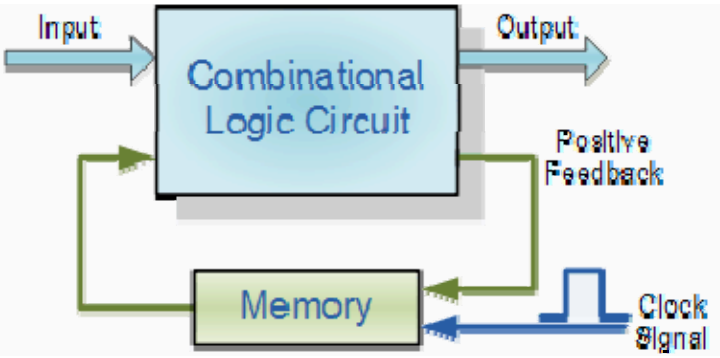
Gambar 7.1 Presentasi Logika Sekuensial

Pada Gambar 7.1 merupakan presentasi suatu untai logika sekuensial, yang memiliki beberapa elemen penyusun. Logika sekuensial memiliki masukan dan keluaran seperti halnya untai logika kombinasional, namun juga memiliki mekanisme pewaktuan (*timing*) atau sinkronisasi (*synchronizing*) dan sifat pengingat (elemen memori) untuk mengingat/mencatat nilai suatu keadaan (*state*). (ada yang menyebutkan sebagai state register).

Keluaran logika sekuensial dapat (biasanya ada) memiliki bagian / elemen kombi nasional. Keluaran logika sekuensial tergantung pada pewaktuan (*timing*) dan elemen pencatat keadaan (*state elemen*) serta variable variable masukannya. Ini berarti keadaan keluaran biasanya berubah sebagai fungsi dari elemen pewaktuan (*timing*).

Untai logika sekuensial pada umumnya merupakan piranti dua keadaan (*two state*) atau (*Bistable*) yang keluarannya terdiri atas dua keadaan yaitu logika 1 atau logika 0. Keadaan keluaran tersebut akan diingat (*latched*) selamanya sampai untai memperoleh sinyal atau pulsa pemicu (*trigger pulse*) pada masukannya yang menyebabkan keluarannya berubah keadaan.

Sequential Logic Representation



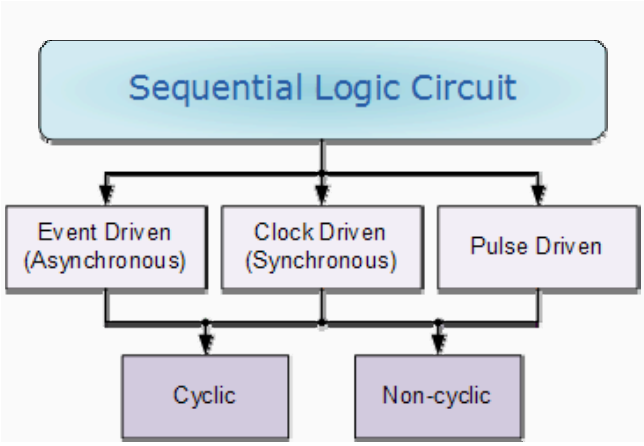
Istilah "Sekuensial" berarti sesuatu terjadi secara berurutan (*sekuens*), keadaan saat ini (*present state*) akan diikuti keadaan selanjutnya (next state). Pada untai Logika Sekuensial, keadaan berikutnya akan dipicu oleh suatu sinyal clock. Untai kogika sekuensial yang sederhana dapat disusun berdasarkan untai dua keadaan (Bistable) seperti Flip-flop, Penaut (Latches) dan Pencacah (Counter). Elemen untai sekuensial dapat disusun berdasar gerbang gerbang dasar seperti NAND dan/atau NOR.

Klasifikasi Logika Sekuensial (*Classification of Sequential Logic*)

Gerbang-gerbang logika dasar merupakan blok pembangun (*building blocks*) suatu sirkuit kombinasional, sedangkan memori (*latches*) dan flip-flop merupakan blok pembangun Sirkuit Logika Sekuensial. Sirkuit Logika Sekuensial sendiri dapat untuk membangun edge-triggered flip-flop, maupun Sirkuit Sekuensial yang lebih kompleks

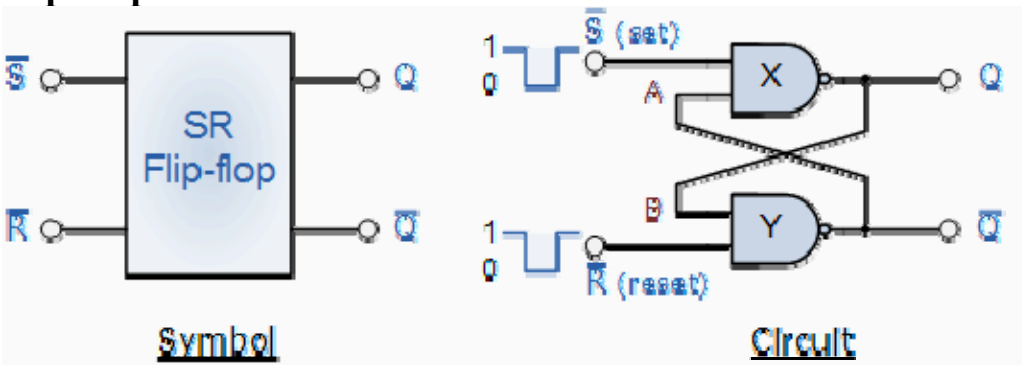
seperti register penyimpan (*storage register*), shift registers, piranti memori (*memory device*) atau pencacah (*counter*). Untai logika sekuensial dapat dibagi dalam tiga kategori utama yaitu;

1. *Event Driven* – untai asinkron (*asynchronous circuits*) yang state-nya akan segera berubah bila di-enable.
2. *Clock Driven* – untai sinkron (*synchronous circuits*) yang disinkronkan dengan suatu sinyal *clock* tertentu.
3. *Pulse Driven* - yang merupakan kombinasi dari duanya yang merespon pulsa-pulsa pemicu. (which is a combination of the two that responds to triggering pulses.)



Gambar 7.2 Kategori logika sekuensial

SR Flip-Flop



Gambar 7.3 Simbol Untai SR Flip Flop

Contoh paling sederhana piranti logika sekuensial adalah SR-FF atau SR-Latch.

Untai ini adalah suatu piranti *non-clocked* yang sederhana, terdiri atas dua gerbang NAND 2-masukan yang dihubungkan secara silang (*crossconnected*). Seperti diperlihatkan pada Gambar 7.3. (Namun dapat juga diwakili dengan kombinasi jenis gerbang yang lainnya misalnya gerbang NOR)

SR-FF memiliki keluaran yang tergantung pada keadaan saat ini (*current state / present state*) dan juga input-input nya

Bentuk untai SR-FF menggunakan logika masukan aktif rendah (*active low*) atau *negative-true*. Hal ini berarti, perubahan keadaan berdasar transisi ke logika 0 (diawal keadaan 0), bukan ke logika 1.

Untai SR-FF memiliki tiga keadaan yang disepakati, yaitu keadaan *set, reset* dan mengingat (*hold*)

Keadaan *Set*:

Keadaan ***set*** terjadi bila keadaan keluaran $Q = 1$, dan $\bar{Q} = 0$.

Keadaan *Reset*:

Keadaan ***reset*** terjadi bila keadaan keluaran $Q = 0$, dan $\bar{Q} = 1$.

Keadaan keluaran Q dan $\bar{Q}$, selalu berlawanan. Suatu SR-FF adalah *bistable.* Ini berarti akan mempertahankan (*latched*) keadaan keluaran (set atau reset) sampai suatu keadaan masukan memaksa untuk berubah.

Keadaan Mengingat(*hold*):

Keadaan mengingat (*hold*) terjadi bila SR-FF mendapat masukan $\bar{S}$ = 1 dan $\bar{R}$=1, keadaan ini akan menyebabkan keluaran tidak berubah.

Siklus Set SR Flip-Flop (*Set Cycle*)

Mencermati kembali, bahwa Gerbang NAND dua masukan, akan memiliki keluaran 0, jika dan hanya jika kedua masukannya berlogika 1.

Anggap bahwa FF dalam kondisi reset atau keluaran Q= 0. Juga karena FF tidak akan aktif (berubah keadaan) jika kedua masukannya berlogika 1.

Analisis cara kerja SR-FF untuk siklus *set* sebagai berikut:

Asumsikan:

Anggap kondisi FF dalam keadaan *reset* , Q = 0, dan $\bar{Q}$ = 1, dan masukannya $\bar{S}$ = $\bar{R}$ = 1.

Maka siklusnya adalah sebagai berikut:

1. Masukan $\bar{S}$ diaktifkan ($\bar{S} \rightarrow 0$)
2. Keluaran Q (NAND X) harus → 1 (Sifat keluaran gerbang NAND akan bernilai 0 jika kedua inputnya bernilai 1)
3. Oleh karena itu kedua input NAND Y akan bernilai 1, maka keluaran $\bar{Q} \rightarrow 0$
4. Input yang lain dari NAND X sekarang menjadi 0, bila masukan $\bar{S}$ berubah menjadi tinggi atau “1”, maka Q akan tetap (mengingat) logika “1” , karena input yang lain mendapat umpan balik (feed back) dari NAND Y masih bernilai “0”.
5. Sebaliknya , karena kedua input gerbang NAND Y sekarang masih bernilai 1, maka $\bar{Q}$ akan mengingat = 0.

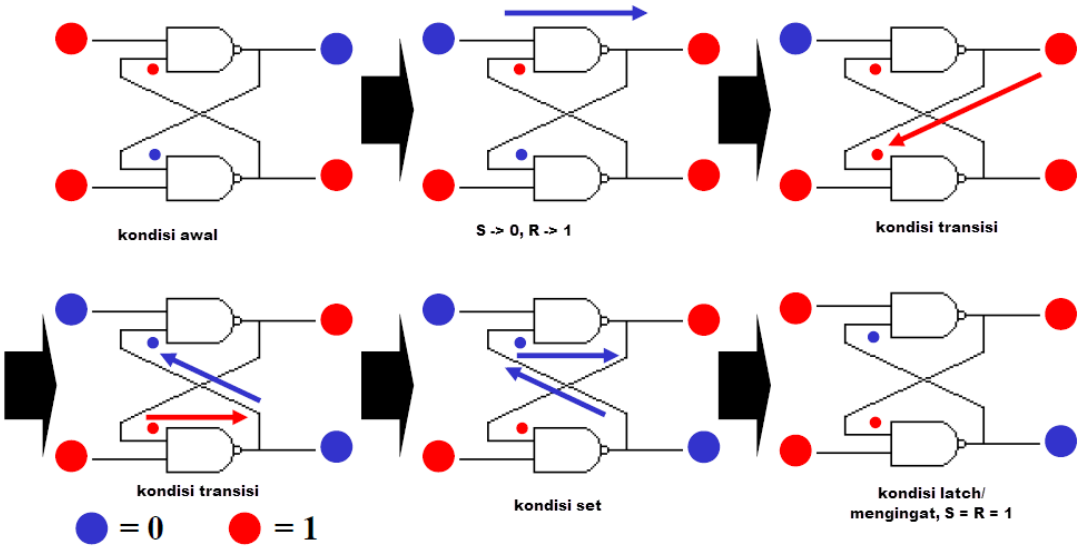
6. Siklus kebalikannya (*set-ke-reset*) adalah dengan cara yang identik, namun masukan $\bar{R}$ menjadi rendah ($\bar{R} \rightarrow 0$)

Tabel 7.1 Tabel transisi SR-FF

State	Inputs		Current Output		New Output		Description
	$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	Q	$\bar{Q}$	
Set	0	1	1 atau 0	0 atau 1	1	0	Set Q » 1
	1	1	1	0	1	0	Tak berubah (*Latched*)
Reset	1	0	1 atau 0	0 atau 1	0	1	Reset Q » 0
	1	1	0	1	0	1	Tak berubah (*Latched*)
Invalid	0	0	0	1	1*	1*	$Q = \bar{Q} = 1$
	0	0	1	0	1*	1*	$Q = \bar{Q} = 1$

*) Kondisi invalid $Q = \bar{Q} = 1$ adalah tidak sesuai dengan kesepakatan bahwa Q dan $\bar{Q}$

Penjelasan cecara grafis suatu siklus set



Keadaan Invalid (*Forbidden*)

Kondisi invalid $Q = \bar{Q} = 1$ adalah tidak sesuai dengan kesepakatan bahwa Q dan $\bar{Q}$, selalu berlawanan. Kondisi invalid ini dipicu oleh $\bar{S} = \bar{R} = 0$, sehingga kedua gerbang NAND $Q = \bar{Q} = 1$.

Jika $\bar{S}$ dan $\bar{R}$ tiba-tiba menjadi berlogika 1 secara simultan, maka kesemua (4) masukan dari kedua gerbang NAND adalah 1 dan kedua keluarannya menjadi 0 (!).

Hasil kondisi rancu “race condition” terlihat dimana kondisi keluaran sebelumnya bernilai 0, satu dari masukan NAND mendapat masukan 0, dan memaksa kedua keluarannya bernilai 1

Hasil kondisi “race” ini tidak dapat diprediksi, karena itu $\bar{R}=\bar{S}=0$ bersama dihindari (*forbidden*), karena keluarannya tidak stabil!

Namun demikian kondisi ini, tidak menyebabkan untai ini menjadi rusak. Beberapa buku teks menyebutkan dengan istilah invalid, atau “forbidden “ (terlarang), atau “race condition” tidaklah sepenuhnya benar, sebaliknya dalam keadaan tertentu secara praktis dapat dimanfaatkan sebagai piranti untuk mendeteksi kegagalan (*fault detector*) suatu sirkuit.

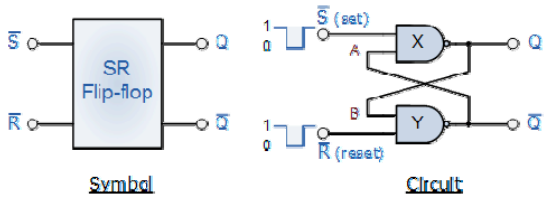
Penggunaan SR-FF

Beberapa jenis FF yang lebih kompleks dapat disusun berdasar SR-FF yang sederhana. Pada kenyataannya, bahwa SR-FF sederhana merupakan basis dari semua piranti logika sekuensial (*sequential logic*). Beberapa sirkuit pada komputer moderen berdasar atau untai lain yang berelasi dengan SR-FF. Piranti-piranti tersebut misalnya:

Register penyimpan data (data storage register) yang sering disederhanakan dengan istilah ***register***.

- Pencacah Biner (*binary counter*)
- Register Geser (*shift register*)
- Semua jenis penampilkan status (status indicators)
- Berbagai tipe memori computer (kecuali DRAM, yang paling banyak digunakan sebagai elemen memori, bukanlah berbasis melainkan kapasitor sebagai penggantinya)

Flip-Flops dan “Memory”



Q can be set to 1 or 0. Thus the output Q can be said to "remember" one bit.

Bilamana SR-FF (atau beberapa jenis FF yang lebih eksotis/kompleks akan kita pelajari selanjutnya) memiliki keluaran Q yang dapat berubah ke 1 atau ke 0, dan keadaan keluaran tersebut akan dipertahankan hingga hadir suatu input yang berlawanan sebagai pemicu (*triggered*) . (memiliki sifat *bistable*)

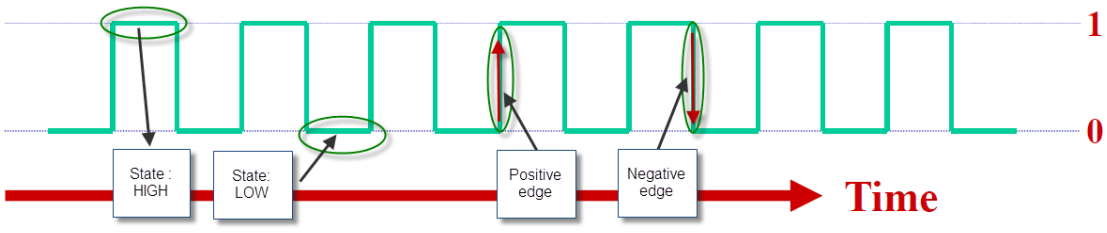
Oleh karena itu FF atau Latch memiliki sifat "mengingat" *(remembering*) bilangan biner satu-bit. Dapat di set ke 1 atau 0.

FF menjadi salah suatu pilihan teknologi memori penyimpan (*storage memory*). Hal ini, kita dapat menggunakan FF untuk menyimpan bilangan biner dalam computer.

Untai logika dengan Clock (*Clocked Circuits*)

Istilah-istilah dalam suatu system Clock diperlihatkan seperti pada Gambar berikut;

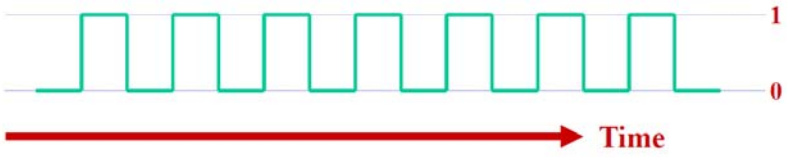
- State TINGGI (*HIGH*), State RENDAH (*LOW*)
- Transisi RENDAH ke TINGGI = Positive Edge Trigger (*leading edge*)
- Transisi TINGGI ke RENDAH = Negative Edge Trigger (*trailing edge*)



The majority of all sequential logic circuits are clocked logic circuits.

• Clocked circuits are circuits that are regulated by a clocking element, which determines when state changes are made in the sequential logic circuit.
• In a clocked sequential circuit, in general, the circuit can only change states on a "tick" of the clock element. Thus we say that all operations in this type of circuit are "clocked."

• We refer to a circuit as a "clocked circuit" when all the sequential elements in the circuit change states in synchronization to a train of pulses.
• Such a "pulse train" is shown below.
• The clock pulses change regularly from 0 to 1 and back.



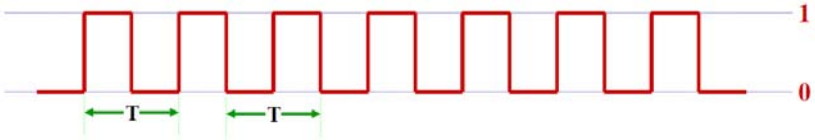
Periode T, adalah jarak diantara dua titik yang identik, yaitu dari transisi naik ke transisi naik berikutnya atau dari transisi turun ke transisi turun berikutnya.

Suatu clock hanya memiliki dua keadaan: 0 dan 1, dan memiliki perubahan diantara dua keadaan tersebut.

Duty Cycle adalah waktu yang diperlukan masing-masing keadaan, yaitu saat keadaan tinggi dan saat keadaan rendah.

The period T of a sequential logic clock is the distance between identical points on the pulse train, e.g., two rising edges or two falling edges.
– The clock has only two states: 0 and 1.
– The clock alternates between the states.
• The amount of time that the clock spends in each of the two states is called the duty cycle.
• The clock below has a 50/50 duty cycle; it spends equal time each period in the 1 and 0 states.

Suatu clock dikatakan memiliki duty cycle 50/50, bila waktu keadaan tinggi sama dengan waktu keadaan rendah dalam satu periode.



Suatu clock tidak selalu memiliki duty cycle 50/50, berikut suatu sinyal clock yang memiliki keadaan "1" nya 35%, dan keadaan "0" nya 65% dari periode sinyal. Maka sinyal clock tersebut dikatakan "35/65". Dengan cara yang sama jika sinyal dikatakan

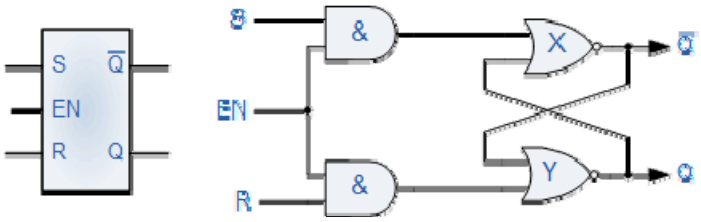
memiliki duty cycle “70/30” berarti sinyal tersebut memikili keadaan tinggi 70% dan keadaan rendah 30% dari periode sinyal.

• The clock below does not have a 50/50 duty cycle. It stays in the “1” state about 35% of the time, and in the “0” state about 65% of the time.
• Thus we say that the clock has a “35/65” duty cycle. In the same way, a clock can have a 70/30 cycle time (i.e., it stays in the “1” state 70% of the time), and so forth.
• Note that the period T is defined in the same way as before.

Clocked Flip-Flops

All ff's have the same basic configuration:
– Both true and false outputs ("Q" and "Q-not").
– "Set" is when Q=1.
– Triggered by "set" and "reset" inputs.
• The most useful ff's are not simple asynchronous (non-clocked) ff's, however, but synchronous ("clocked") ff's.
• Clocked ff's are very similar to non-clocked ff's -- the main difference is that in addition to a "set" or "reset" input to cause the outputs to change, there must also be the presence of a clock signal in its true state (normally 1).
• Thus clocked ff's do not change states, regardless of the set or i il h " l k i k "
reset inputs, until the "clock ticks."

The Clocked R-S Flip-Flop
SR-FF dengan clock



Gambar 7.x SR-FF dengan Clock (Clocked SR-FF)

The simplest clocked ff is the clocked R-S FF, shown above (NAND version).
• In addition to the set and reset inputs, the clock input is present.
• Since when clock is low (0), neither set or reset input affect the circuit, we say that the clock "gates" the set or reset signal to the RS FF.
• In this case, the set or reset input must be high (1) to set or reset the ff when the clock goes true (0 1).
• Having set and reset 1 at the same time is forbidden as for the RS FF; simultaneous set and reset true causes a race condition when clock is high.

Tabel transisi SR-FF dengan-clock

State	Clock/ Enable (state)	Inputs		Current Output		New Output		Description
		S	R	Q	$\bar{Q}$	Q	$\bar{Q}$	
	0	*	*	1 atau 0	0 atau 1	Sama	sama	Disable
Set	1	1	0	1 atau 0	0 atau 1	1	0	Set Q » 1
	1	0	0	1	0	sama	sama	Tak berubah (*Latched*)
Reset	1	0	1	1 atau 0	1 atau 0	0	1	Reset Q » 0
	1	0	0	0	1	0	1	Tak berubah (*Latched*)
Invalid	1	1	1	1 atau 0	0 atau 1	1*	1*	$Q = \bar{Q} = 1$* (*invalid*)

*Kondisi invalid sama seperti pada SR-FF yang tanpa pemicu (*non-clocked*).

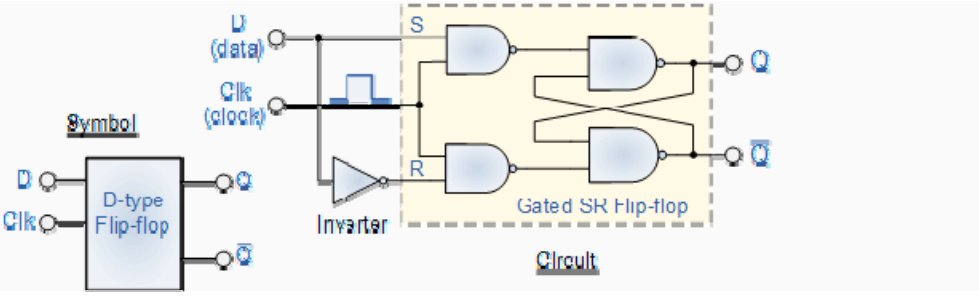
Modern computers use much combinational logic, but they are synchronized and controlled by sequential logic activated by a clock or pulse train.
• Thus the clocked RS FF is an example of sequential logic that could be used in a modern computer circuit.
• However, more sophisticated flip-flops are normally utilized in modern computers.
• One of these is the D flip-flop.

Komputer-komputer modern menggunakan banyak untai logika kombinasional, tetapi untai-untai tersebut di sinkronkan dan di control secara logika sekuensial yang diaktifkan oleh suatu clock atau deretan pulsa pulsa (*pulse train*).

SR-FF ber-clock adalah salah satu bentuk logika sekuensial yang dapat digunakan dalam untai computer modern. namun demikian, banyak bentuk bentuk flip-flop yang lebih canggih lain yang biasanya digunakan dalam computer modern. Salah satunya adalah D-FF.

D flip-flop

Sirkuit Flip-Flop D (D flip-flop Circuit)



Gambar 7.x D-FF ber-clock

The D FF is a bistable circuit with only one input plus the clock. The term "refers to "data " D" data."

Since the D FF is limited to one input, there is no chance that a race condition will occur.

- The clocked D FF can be created simply by replacing the "reset" input with an inverted "set" input in the clocked RS FF.

• In this configuration, we can see that if the D input = 1, when the clock is high, the ff will go into the "set" state ("set" input=1, "reset"=0).
• When the D input = 0, then the ff goes into the "reset" state when clock goes high ("set" input=0, "reset"=1). When clock = 0, the ff is idle.
• There is no "forbidden" state here -- the D FF output is defined (see truth table, next page) for ALL inputs.

Truth Table for the D Flip-flop

Tabel Kebenaran D-FF

Masukan		Present state		Next state		Deskripsi
D	Clk	Q	$\bar{Q}$	Q	$\bar{Q}$	
1 atau 0	↓ » 0	1 atau 0	0 atau 1	sama	sama	Memory tak berubah
0	↑ » 1	0	1	0	1	Reset Q » 0
1	↑ » 1	0	1	1	0	Set Q » 1
0	↑ » 1	1	0	0	1	Reset Q » 0
1	↑ » 1	1	0	1	0	Set Q » 1

Catatan: ↓ dan ↑ menunjukan arah pulsa clock, D-FF seperti diatas adalah dianggap terpicu tebing (*edge triggered*). (sesungguhnya diawal state tinggi)

"Master-Slave," or Delay Flip-Flops

Sering diperlukan suatu FF yang keluarannya *tidak berubah secara mendadak* (*immediately*) bila state internalnya berubah dari keadaan "set" (Q=1) ke "reset" (Q=0), demikian juga sebaliknya.

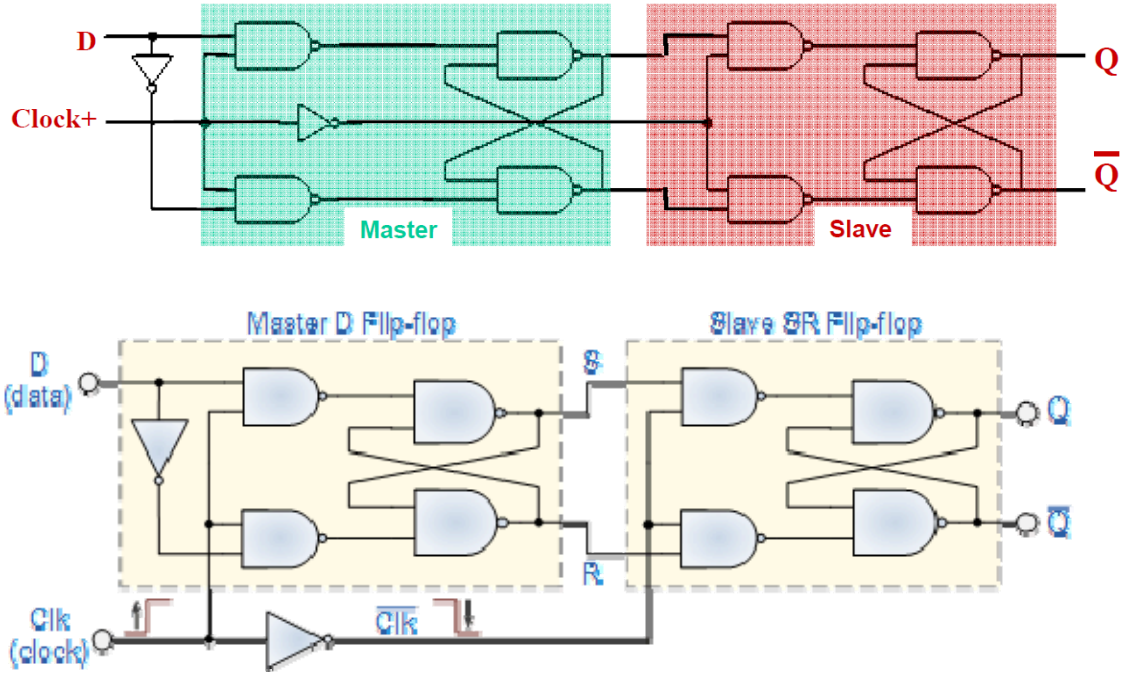
FF yang demikian disebut suatu "master-slave flip-flop" (FF Induk-budak) atau "delay flip-flop" (FF tertunda).

.It is often desirable to have a flip-flop whose output does not change immediately when its internal state is altered from "set" (Q=1) to "reset" (Q=0), or vice-versa.

• This sort of ff is called a "master-slave" or "delay" ff.

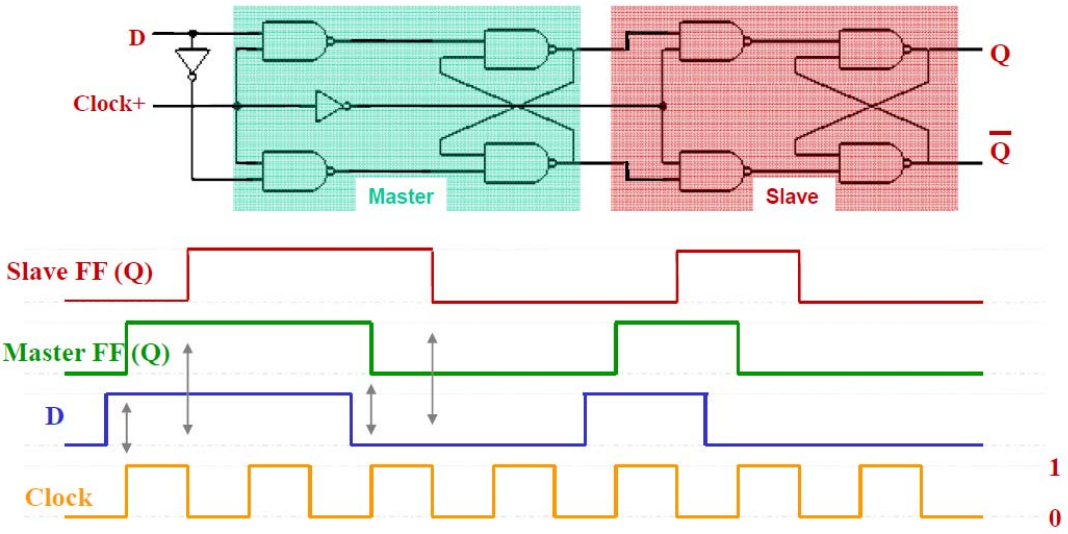
• The idea behind the master-slave ff is to have a "master" (i.e., controlling) ff change states on one edge of a clock pulse (normally the leading edge) and have a second ff connected to the first change to the same state as the "master" on the trailing edge, or "backside" of a clock pulse.

• In this way, the internal state of the ff changes one-half clock cycle prior to the time in which the changed state appears on the circuit outputs.



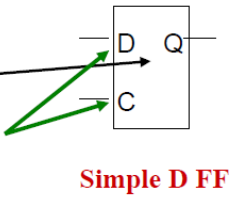
D FF converted to master-slave type.

• The slave (basically a clocked RS FF) always mirrors the state of the master.
• The slave circuit changes state 1/2 cycle after the master.
• The device still operates as a D FF; no indeterminate state.

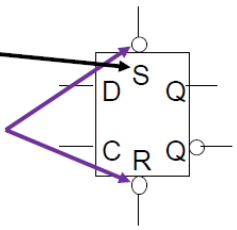


D Flip-Flop Symbols

- **Flip-flop detail is not usually shown in diagrams.**
- **One symbol for a D FF is shown to the right.**
- **There is no small circle on either input. Therefore, "1" is the active state (when clock and D = 1, output will → 1).**
- **D FF's with asynchronous set and reset are also available.**
- **Circles on S and R inputs mean that set and reset are "negative-true" signals (active at level 0).**
  - **"Q-not" output is also available.**
- **Set and reset have the same problems discussed before: If S = R = 0, output may be indeterminate.**



Simple D FF

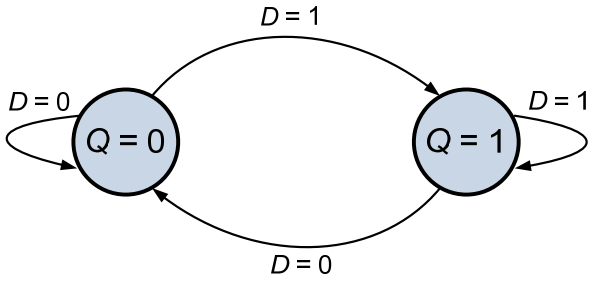


D FF With Asynchronous S/R

Flip-flop detail is not usually shown in diagrams.
• One symbol for a D FF is shown to the right.
• There is no small circle on either input. Therefore, "1" is the active state (when clock and D = 1, output will → 1).

• D FF's with asynchronous set and reset are also available.
• Circles on S and R inputs mean that set and reset are "negative-true" signals (active at level 0).

– “Q-not” output is also available.
• Set and reset have the same problems discussed before: If S = R = 0, output may be indeterminate.
Prentasi D-FF dengan State Diagram



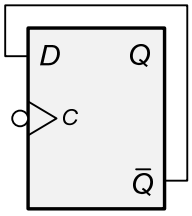
JK flip-flop

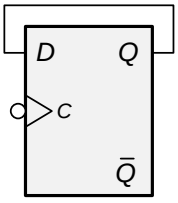
Pemenu JK-FF adalah Jack Kilby, yang bertujuan untuk menutupi kekurangan SR-FF yang memiliki kondisi invalid (ketika kedua keluarannya $Q=\bar{Q}$)
Ada 4 keadaan pada JK-FF

1. Toggle, $Q_{n+1} = \bar{Q}_n$
2. Keadaan mengingat (*hold*), $Q_{n+1} = Q_n$,
3. Keadaan reset, Q = 0, dan keadaan
4. Keadaan set , Q = 1

1. Piranti JK-FF, berdasar D-FF

JK-FF dapat disusun berdasar D-FF, perhatikan Gambar xx(a), keluaran $\bar{Q}_n$, di umpan balikan kemasukkan *D* atau $D=\bar{Q}_n$, maka keadaan berikutnya (*next state*) adalah $Q_{n+1} = \bar{Q}_n$, atau bersifat *toggle* (keadaan berikutnya berkebalikan dengan keadaan sekarang) Pada konfigurasi Gambar xx(b), bahwa $D= Q_n$, atau $Q_{n+1} = Q_n$, maka untai akan mengingat (keadaan berikutnya sama dengan keadaan sekarang).

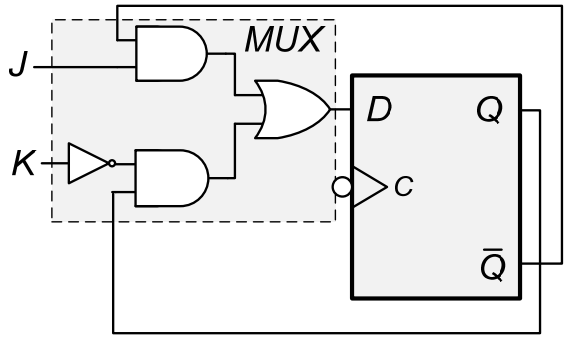




(a) $D = \bar{Q}$, atau $Q_{n+1} = \bar{Q}_n$ (b) $D = \bar{Q}$, atau $Q_{n+1} = Q_n$

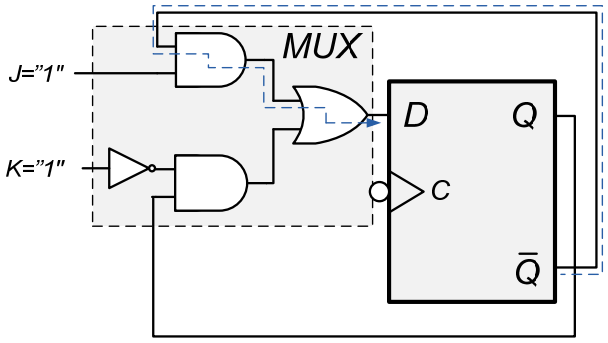
Gambar xx D-FF keadaan *Toggle* dan *Mengingat*

Kedua konfigurasi diatas dapat digabung dengan suatu untai multipleks (MUX), seperti pada Gambar xx. MUX tersebut dimaksudkan untuk memilih kapan untai bersifat *toggle* dan kapan bersifat mengingat (*hold*). Kedua keadaan tersebut dapat dipilih melalui masukan *J* dan *K*.

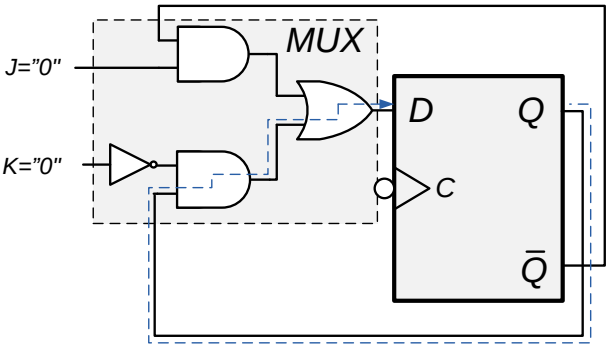


Gambar xx. Untai MUX (*Multipleks*) dapat digunakan untuk memilih fungsi untuk masukan *D*

Masukan D-FF, adalah $D = J.\bar{Q} + \bar{K}.Q$



(a) J = K = 1 , maka $D = \bar{Q}$, atau $Q_{n+1} = \bar{Q}_n$ (*Toggle*)



(b) $J = K = 0$, maka $D = \bar{Q}$, atau $Q_{n+1} = Q_n$ (*Mengingat*)

Keadaan Toggle

Pada Gambar (a) *J*=1, menyebabkan gerbang AND (bagian atas dalam MUX) menjadi *enable* , sedang *K*=1, menyebabkan gerbang AND (bagian bawah dalam MUX) menjadi *disable*. Oleh karena itu $\bar{Q}$ akan diteruskan ke masukan *D*, maka $D = \bar{Q}$ atau $Q_{n+1} = Q_n$, hal ini untai keseluruhan menjadi flip-flop bersifat *Toggle*

Keadaan Mengingat:

Pada Gambar (b) *J*=0, menyebabkan gerbang AND (bagian atas dalam MUX) menjadi *disable* , sedang *K*=0, menyebabkan gerbang AND (bagian bawah dalam MUX) menjadi *enable*. Oleh karena itu Q akan diteruskan ke masukan *D*, maka $D = Q$ atau $Q_{n+1} = Q_n$, hal ini untai keseluruhan menjadi flip-flop bersifat *mengingat*

Kondisi Reset:

Jika $J = 0$ dan $K = 1$. $J = 0$ memaksa keluaran gerbang AND (bagian atas) selalu bernilai "0", apapun kondisi $\bar{Q}$. $K = 1$ juga memaksa keluaran gerbang AND (bagian bawah) selalu berlogika 0, apapun kondisi Q.

Perhatikan persamaan $D = J.\bar{Q} + \bar{K}.Q$

Jika $J = 0$ dan $K = 1$,

Maka keadaan $D = 0.\bar{Q} + 0.Q = 0.(\bar{Q} + Q) = 0$.

Keadaan ini menyebabkan keluaran FF berikutnya Q=0, atau (*reset*).

Kondisi Set:

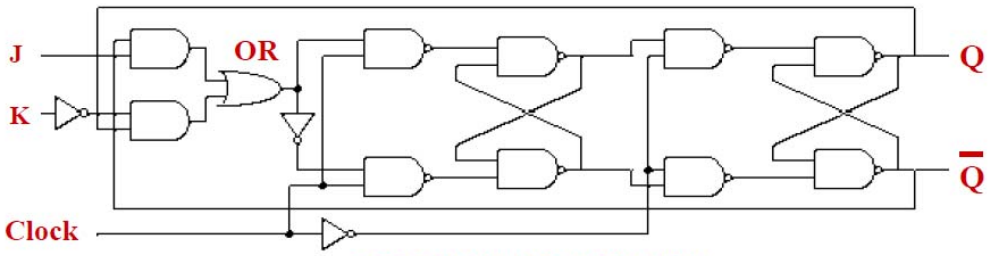
Perhatikan persamaan $D = J.\bar{Q} + \bar{K}.Q$

Jika $J = 1$ dan $K = 0$,

Maka keadaan $D = 1.\bar{Q} + 1.Q = 1.(\bar{Q} + Q) = 1$.

Keadaan ini menyebabkan keluaran FF berikutnya Q=1, atau (*set*).

Internals of the J-K Flip-Flop



J-K Master-Slave Flip Flop

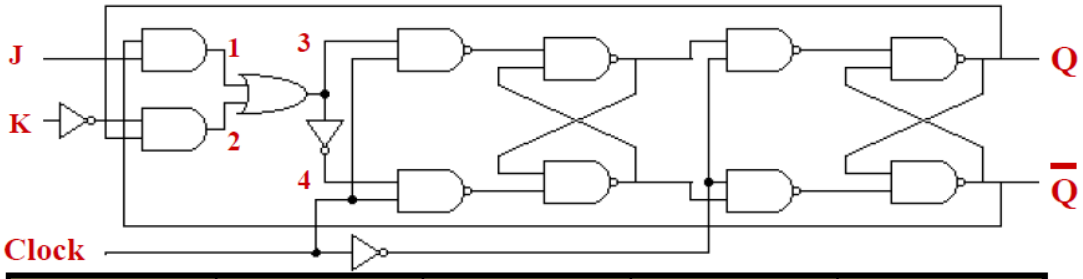
- **A master-slave J-K FF can be designed as shown above.**
- **The key states are J=K=1, for either output state (set or reset).**
- **If Q = 1 ("Set") and J = K = 1, output of the OR = 0, so the ff will reset.**
- **Likewise, if Q = 0 ("Reset") and J = K = 1, OR = 1, and the ff will be set.**
- **(For J=1 and K=0, or J=0 and K=1) normal set or reset occurs.)**
- **Then for J = K = 1, when the clock ticks Q → the opposite state.**

A master-slave J-K FF can be designed as shown above.
• The key states are J=K=1, for either output state (set or reset).
- If Q = 1 ("Set") and J = K = 1, output of the = 0, so the ff will reset.

• Likewise, if Q = 0 ("Reset") and J = K = 1, OR = 1, and the ff will be set.
• (For J=1 and K=0, or J=0 and K=1) normal set or reset occurs.)
• Then for J = K = 1, when the clock ticks Q → the opposite state.

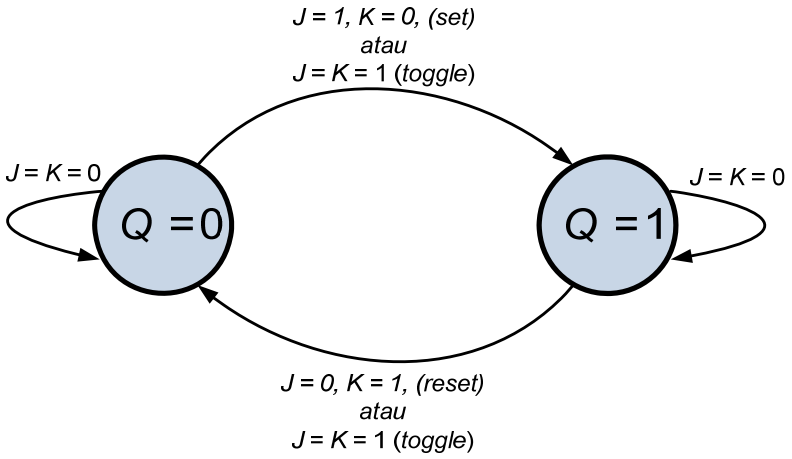
Untai di atas dapat disederhanakan seperti Gambar berikut

JK Flip Flop Truth Table



Inputs		Outputs		AND Outputs		OR Outputs		New Outputs	
J	K	Q	$\overline{Q}$	1	2	3	4	Q	$\overline{Q}$
0	0	0	1	0	0	0	1	Same	Same
0	0	1	0	0	1	1	0	Same	Same
0	1	0	1	0	0	0	1	Same	Same
0	1	1	0	0	0	0	1	0	1
1	0	0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0	Same	Same
1	1	0	1	1	0	1	0	1	0
1	1	1	0	0	0	0	1	0	1

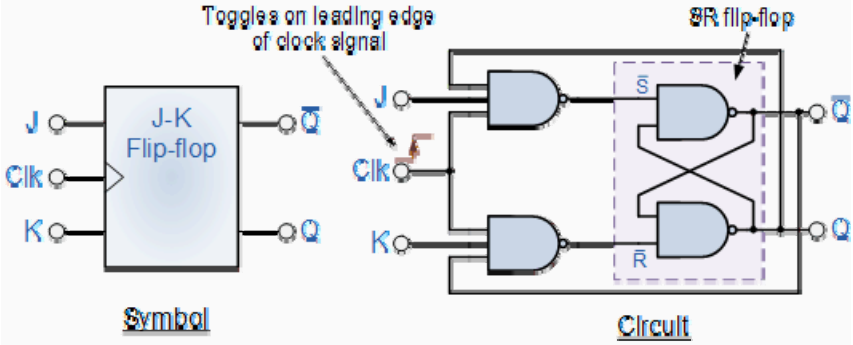
Pesentasi JK-FF dengan Diagram Keadaan



Gambar xx Diagram Keadaan JK-FF

Tambahan Aja/Untuk Latihan

2. JK Flip-flop berdasar SR-FF (*non-cloked*)



Gambarnya!!!(diatas) Q = yang diatas

Both the S and the R inputs of the previous SR bistable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack Kilby. Then this equates to: J = S and K = R.

Then the JK flip-flop is basically an SR flip-flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby

Clk sebagai masukan untuk enable (diawal state tinggi)

$\bar{S}= \overline{J.\bar{Q}.Clk}$
$\bar{R}= \overline{K.Q.Clk}$

Kondisi set

a) Asumsikan kondisi saat ini $Q = 0$ dan $\bar{Q} = 1$
Untuk menge-*set* maka $J = 1$, dan $K = 0$
Maka :
$\bar{S}= \overline{J.\bar{Q}.Clk}$
$\bar{S}= \overline{1.1.1} = 0$
Dan
$\bar{R}= \overline{K.Q.Clk}$
$\bar{R}= \overline{0.0.1} = 1$

Kondisi tersebut memenuhi syarat untuk menge-set SR-FF, $\bar{S} = 0$, $\bar{R}=1$ >> $Q = 1$ dan $\bar{Q} = 0$

b) Asumsikan kondisi saat ini $Q = 1$ dan $\bar{Q} = 0$
Untuk menge-*set* maka $J = 1$, dan $K = 0$
Maka :

$\bar{S}=\overline{J.\bar{Q}.Clk}$
$\bar{S}=\overline{1.0.1}\ =1$
dan
$\bar{R}=\overline{K.Q.Clk}$
$\bar{R}=\overline{0.1.1}\ =1$
Kondisi tersebut adalah mengingat kondisi sebelumnya untuk SR-FF, $\bar{S}$ = 1, $\bar{R}$=1 >> Q = 1 dan $\bar{Q}$ = 0

Kondisi reset

a) Asumsikan kondisi saat ini Q = 1 dan $\bar{Q}$ = 0

Untuk menge-*reset* maka J = 0, dan K = 1
Maka :
$\bar{S}=\overline{J.\bar{Q}.Clk}$
$\bar{S}=\overline{0.0.1}\ =1$
dan
$\bar{R}=\overline{K.Q.Clk}$
$\bar{R}=\overline{1.1.1}\ =0$
Kondisi tersebut adalah kondisi *reset* untuk SR-FF, $\bar{S}$ = 1, $\bar{R}$=0 >> Q = 0 dan $\bar{Q}$ = 1

b) Asumsikan kondisi saat ini Q = 0 dan $\bar{Q}$ = 1

Untuk menge-*reset* maka J = 0, dan K = 1
Maka :
$\bar{S}=\overline{J.\bar{Q}.Clk}$
$\bar{S}=\overline{0.1.1}\ =1$
Dan
$\bar{R}=\overline{K.Q.Clk}$
$\bar{R}=\overline{1.0.1}\ =1$

Kondisi tersebut adalah mengingat kondisi sebelumnya untuk SR-FF, $\bar{S}$ = 1, $\bar{R}$=1 >> Q = 0 dan $\bar{Q}$ = 1 (*reset*)

Kondisi Toggle

a) Asumsikan kondisi saat ini Q = 1 dan $\bar{Q}$ = 0

Untuk bersifat *toggle* maka J = 1, dan K = 1
Maka :
$\bar{S}=\overline{J.\bar{Q}.Clk}$
$\bar{S}=\overline{1.0.1}\ =1$
dan
$\bar{R}=\overline{K.Q.Clk}$
$\bar{R}=\overline{1.1.1}\ =0$
Kondisi tersebut adalah kondisi *reset* untuk SR-FF, $\bar{S}$ = 1, $\bar{R}$=0 >> Q = 0 dan $\bar{Q}$ = 1 (dan ini berkebalikan dengan kondisi sebelumnya)

b) Asumsikan kondisi saat ini Q = 0 dan $\bar{Q}$ = 1

Untuk bersifat *toggle* maka $J = 1$, dan $K = 1$
Maka :
$\bar{S} = \overline{J.\bar{Q}.Clk}$
$\bar{S} = \overline{1.1.1} = 0$
dan
$\bar{R} = \overline{K.Q.Clk}$
$\bar{R} = \overline{1.0.1} = 1$
Kondisi tersebut adalah kondisi *set* untuk SR-FF, $\bar{S} = 0$, $\bar{R}=1$ >> $Q = 1$ dan $\bar{Q} = 0$ (dan ini juga berkebalikan dengan kondisi sebelumnya)

Kondisi Mengingat (*hold*)

a) Asumsikan kondisi saat ini $Q = 1$ dan $\bar{Q} = 0$

Untuk bersifat *toggle* maka $J = 0$, dan $K = 0$
Maka :
$\bar{S} = \overline{J.\bar{Q}.Clk}$
$\bar{S} = \overline{0.0.1} = 1$
dan
$\bar{R} = \overline{K.Q.Clk}$
$\bar{R} = \overline{0.1.1} = 1$
Kondisi tersebut adalah kondisi *mengingat* untuk SR-FF, $\bar{S} = 1$, $\bar{R}=1$ >> $Q = 1$ dan $\bar{Q} = 0$ (dan ini sama dengan kondisi sebelumnya)

b) Asumsikan kondisi saat ini $Q = 0$ dan $\bar{Q} = 1$

Untuk bersifat *toggle* maka $J = 0$, dan $K = 0$
Maka :
$\bar{S} = \overline{J.\bar{Q}.Clk}$
$\bar{S} = \overline{0.1.1} = 1$
dan
$\bar{R} = \overline{K.Q.Clk}$
$\bar{R} = \overline{1.0.1} = 1$
Kondisi tersebut adalah kondisi *mengingat* untuk SR-FF, $\bar{S} = 1$, $\bar{R}=1$ >> $Q = 0$ dan $\bar{Q} = 1$ (dan ini sama dengan kondisi sebelumnya)

The Truth Table for the JK Function

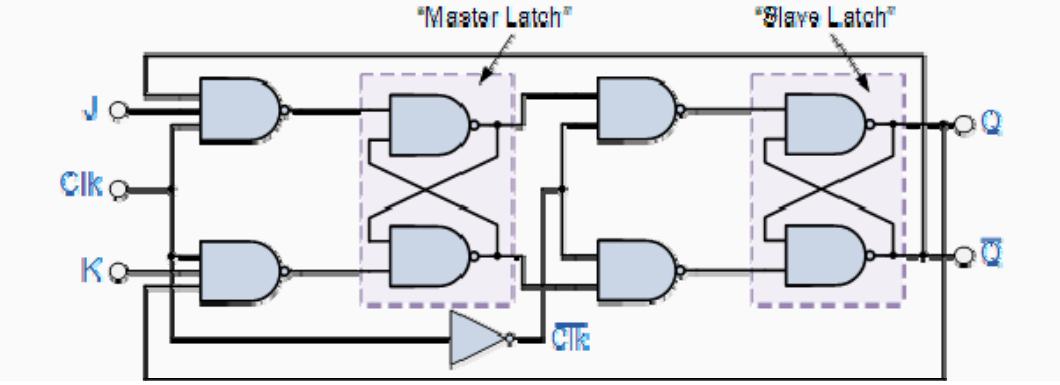
Tabel kebenaran / transisi JK-FF

	clk	Masukan		Present State		Next State		Deskripsi
		J	K	Q	$\bar{Q}$	Q	$\bar{Q}$	
Sama dengan SR-Latch	↑ » 1	0	0	0	0	sama	sama	Memory (tidak berubah)
	↑ » 1	0	0	0	1	sama	sama	
	↑ » 1	0	1	1	0	0	1	Reset Q » 0

	↑ » 1	0	1	0	1	0	1	
	↑ » 1	1	0	0	1	1	0	Set Q » 1
	↑ » 1	1	0	1	0	1	0	
Aksi toggle	↑ » 1	1	1	0	1	1	0	Toggle
	↑ » 1	1	1	1	0	0	1	

Then the JK flip-flop is basically an SR flip-flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby

The Master-Slave JK Flip-Flop



The Truth Table for the JK Function

Tabel kebenaran / transisi JK-FF

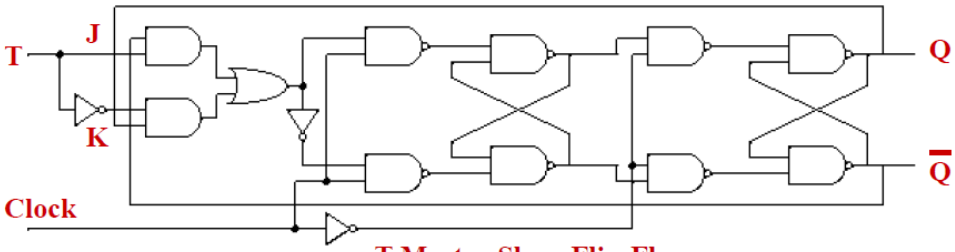
Sama dengan SR-Latch	clk	Masukan		Present State		Next State		Deskripsi
		J	K	Q	$\bar{Q}$	Q	$\bar{Q}$	
	↓ » 0	0	0	0	0	sama	sama	Memory (tidak berubah)
	↓ » 0	0	0	0	1	sama	sama	
	↓ » 0	0	1	1	0	0	1	Reset Q » 0

↓

T Flip-Flop

Navigation

The Toggle Flip-Flop



T Master-Slave Flip-Flop

The T FF is like a JK FF with J and K tied together (K input inverted).

Then if T = 1, and clock = 1, the ff "toggles" to the opposite state.

If T = 0, the ff does not change state on the clock "tick."

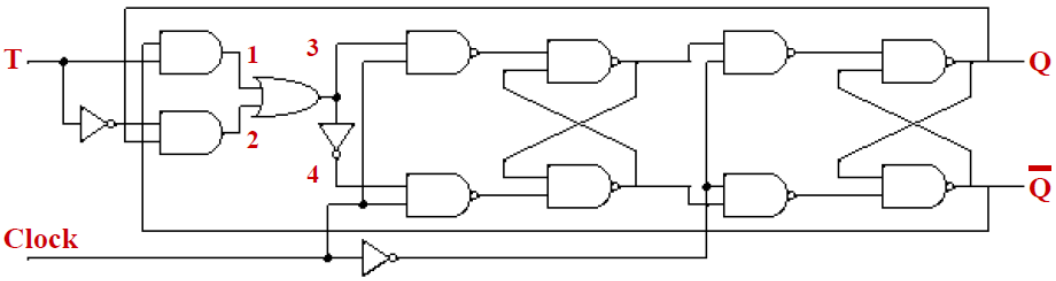
The T FF is a master-slave ff; output changes on the back edge of the clock.

Set T = 1 permanently, and the T FF toggles on every clock pulse.

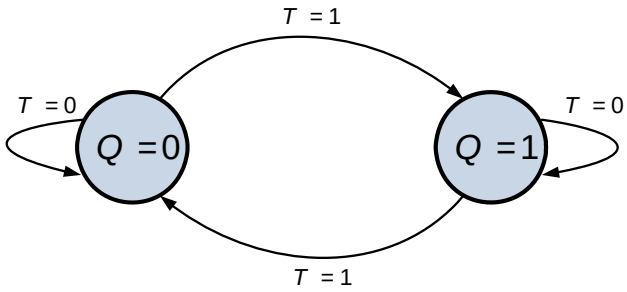
Note Q tied to the K input and Q-not tied to the J input. This "feedback," along with the connected J and K inputs, enables the T FF to work properly.

The T FF is like a JK FF with J and K tied together (K input inverted).
• Then if T = 1, and clock = 1, the ff "toggles" to the opposite state.
• If T = 0, the ff does not change state on the clock "tick."
• The T FF is a master-slave ff; output changes on the back edge of the clock.
• Set T = 1 permanently, and the T FF toggles on every clock pulse.
• Note Q tied to the K input and Q-not tied to the J input. This "feedback," along with the connected J and K inputs, enables the T FF to work properly.

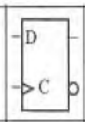
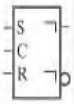
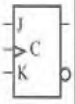
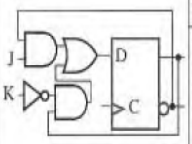
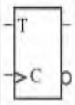
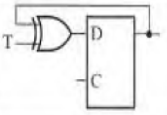
Tabel Kebenaran T-FF



Input T	Outputs		AND Outputs		OR Outputs		New Outputs	
	Q	$\overline{Q}$	1	2	3	4	Q	$\overline{Q}$
0	0	1	0	0	0	1	Same	Same
0	1	0	0	1	1	0	Same	Same
1	0	1	1	0	1	0	1	0
1	1	0	0	0	0	1	0	1

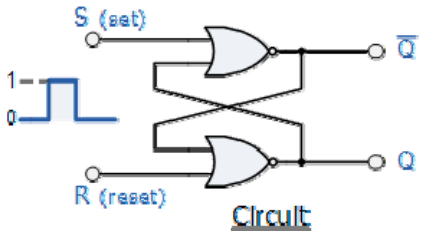


Gambar Diagram Keadaan T-FF

Type	Symbol	Logic Diagrams	Characteristic Table	Characteristic Equation	Excitation Table
D		See Figure 6-13	D | Q(t+1) | Operation 0 | 0 | Reset 1 | 1 | Set	$Q(t+1) = D(t)$	Q(t+1) | D | Operation 0 | 0 | Reset 1 | 1 | Set
SR		See Figure 6-10	S | R | Q(t+1) | Operation 0 | 0 | $Q(t)$ | No change 0 | 1 | 0 | Reset 1 | 0 | 1 | Set 1 | 1 | ? | Undefined	$Q(t+1) = S(t) + \bar{R}(t)\cdot Q(t)$	Q(t) | Q(t+1) | S | R | Operation 0 | 0 | 0 | X | No change 0 | 1 | 1 | 0 | Set 1 | 0 | 0 | 1 | Reset 1 | 1 | X | 0 | No change
JK			J | K | Q(t+1) | Operation 0 | 0 | $Q(t)$ | No change 0 | 1 | 0 | Reset 1 | 0 | 1 | Set 1 | 1 | $\bar{Q}(t)$ | Complement	$Q(t+1) = J(t)\cdot\bar{Q}(t) + \bar{K}(t)\cdot Q(t)$	Q(t) | Q(t+1) | J | K | Operation 0 | 0 | 0 | X | No change 0 | 1 | 1 | X | Set 1 | 0 | X | 1 | Reset 1 | 1 | X | 0 | No Change
T			T | Q(t+1) | Operation 0 | $Q(t)$ | No change 1 | $\bar{Q}(t)$ | Complement	$Q(t+1) = T(t) \oplus Q(t)$	Q(t+1) | T | Operation $Q(t)$ | 0 | No change $\bar{Q}(t)$ | 1 | Complement

SOAL LATIHAN BAB 7

1. Suatu SR-FF, tersusun oleh dua gerbang NOR, buatlah tabel transisi dari FF tersebut.



S	R	Q	$\bar{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	0	0
		(Invalid)	

Catatan Perhatikan posisi Q dan $\bar{Q}$!!!

Lengkapi tabel berikut

State	Inputs		Current Output		New Output		Description
	S	R	Q	$\bar{Q}$	Q	$\bar{Q}$	
Set	1	0	1 atau 0	0 atau 1			Set Q » 1
	0	0	1	0			Tak berubah (*Latched*)
Reset	0	1	1 atau 0	0 atau 1			Reset Q » 0
	0	0	0	1			Tak berubah (*Latched*)
Invalid	1	1	0	1			$Q = \bar{Q} = 0$

	1	1	1	0			$Q = \bar{Q} = 0$

2. Buatlah untai D-FF dengan JK-FF
3. Buatlah untai JK-FF dengan D-FF