

===== Rangkuman PSD =====

☺ Sukses UAS ☺

1. Floating Point & Negative Number

1.1 Floating Point (IEEE 754)

$$\pm M \times B^e$$

- Komponen Floating Point:

M = Mantissa

B = Base/Basis

E = Eksponen

Contoh = M = 2.73 B = 10 E = 2

1 bit	8 bits	23 bits
SIGN	EKSPONEN / BIAS	MANTISSA / FRACTION

- Convert Dec to Bin (Float 32bit):

1. Ubah Decimal menjadi Binary (Ex : 19.510 = 10011.10000...)
2. Ubah bentuk tadi menjadi satu koma 1.--- (Ex : 1.00111 * 2⁴)
3. Tentukan **sign**, apakah 0 atau 1. Jika 0 maka (+) jika 1 maka (-)
4. Tentukan **Ekponen**, menentukannya dari langkah 2. Yaitu pangkat dari 2nya
5. Tentukan **mantissanya**, mantissa sama dengan bentuk binary dari step 1 namun dibuat 23 bits dengan cara menambahkan 0 dibelakang hingga 23 bits. Contoh , 100 1110 0000 0000 0000 0000
6. Tentukan **bias**, bias adalah 127 + eksponen, kemudian ubah menjadi binary. Contoh, 127 + 4 = 131 == 10000011
7. Tentukan **fraction**, sama dengan mantissa, tapi pada fraction kita menghilangkan angka 1 pertama pada mantissa. Contoh, 001 1100 0000 0000 0000 0000

8. Setelah ke 7 step diatas sudah dilakukan maka tinggal menyusul Sign, Bias, dan fraction sesuai urutan table.

- Convert Bin (32 bits) to dec:

1. Tentukan **positive/negative**, dengan melihat sign/bit pertama dari IEEE
2. Menentukan **eksponen**, dengan cara merubah bias yang terdapat di IEEE menjadi decimal lalu kurangi dengan 127.
3. Menentukan **mantissa**, lihat fraction dari soal. Setelah itu tambahkan "1." didepan fraction tsb
4. Buat bentuk decimal dengan cara, hilangkan semua 0 dibelakang angka 1 terakhir pada mantissa, lalu kalikan dengan 2^{eksponen} . Lalu ubah menjadi decimal.

- Floating Point (IEEE) addition / subs:

1. Tentukan **sign**, **eksponen**, dan **mantissa** dari soal pada setiap IEEE.
2. Kalikan mantissa dengan 2^{eksponen}
3. Cari **selisih eksponen** untuk melihat shifting.
4. Melakukan **shift**, atau menggeser masing-masing mantissa dari step 2 ke kiri sebanyak shifting. Agar 2^{eksponen} sama sehingga dapat dikurang/ditambah
5. Setelah itu tentukan lagi **sign**, **bias**, dan **fraction** yang baru setelah dari step 4. Kemudian susun sesuai urutan.

1.2 Negative Number & 1s or 2s Complement

- Bilangan negative di representasikan dengan angka 1 pada awal bit pada binary.

- **Sign Magnitude**

- o Sign magnitude adalah suatu cara untuk merepresentasikan sign dari sebuah bit. Sign tersebut berada di awal yang berguna untuk menentukan positive/negative.
- o Contoh: $+127 = 01111111$ $-127 = 11111111$
- o Zero : $+0 : 0000000$ $-0 : 10000000$

- 1s Complement

- Rumus : $-\text{bin} = 2^{\text{bits}} - \text{decimal}(\text{dari bin}) - 1$
- Contoh : tentukan -12 dalam 1s complement

Jawab, $12 = 00001100$

$$-00001100 = 2^8 - 12 - 1$$

$$= 243$$

$$= 11110011_{1s}$$

- Rumus Cepat = merubah 1 \rightarrow 0 dan 0 \rightarrow 1 di bentuk binary, $00001100 \rightarrow 11110011$
- Nilai dari 1s complement positive adalah tetap, tidak ada perubahan dari bentuk binary.
- Contoh $14 = 00001110_{1s}$
 $-14 = 11110001_{1s}$

- 2s Complement

- Mirip dengan 1s complement, hanya perlu menambahkan 1 pada 1s complement.
- Untuk bentuk positive tetap dengan binarynya, tidak ada perubahan.
- Contoh $-14 = 11110001_{1s} + 1 = 11110010_{2s}$

- 2s Complement Add / Subs

- Dalam 2s complement subs tidak ada pengurangan seperti decimal, yang ada adalah menjumlahkan dengan negatifnya yang dapat di rumuskan $A - B = A + (-B)$. Dengan merubah B menjadi $-B_{2s}$.
- Overflow adalah suatu kondisi ketika dilakukannya operasi yang hasil dari operasi tersebut tidak merepresentasikan nilai decimalnya dengan sesuai.
 Contoh, $5 + 6 = 0101_{2s} + 0110_{2s} = 10110_{2s}$
 $= -5 ??$ Maka Overflow

2s Complement Addition/Subtraction (2/3)

- Examples: 4-bit system

+3	0011
+ +4	+ 0100
----	-----
+7	0111
----	-----

-2	1110
+ -6	+ 1010
----	-----
-8	11000
----	-----

+6	0110
+ -3	+ 1101
----	-----
+3	10011
----	-----

+4	0100
+ -7	+ 1001
----	-----
-3	1101
----	-----

1s Complement Addition/Subtraction (2/2)

- Examples: 4-bit system Any overflow?

+3	0011
+ +4	+ 0100
----	-----
+7	0111
----	-----

+5	0101
+ -5	+ 1010
----	-----
-0	1111
----	-----

-2	1101
+ -5	+ 1010
----	-----
-7	10111
----	-----
	+ 1
	1000

-3	1100
+ -7	+ 1000
----	-----
-10	10100
----	-----
	+ 1
	0101

2. Sequential Circuit

2.1 Pengertian

Sequential circuit berbeda dengan combinational. Letak perbedaannya adalah jumlah inputnya, sequential memiliki 2 jenis variable input yaitu state dan data. Sedangkan input dari combinational hanya tergantung pada data.

2.2 Latch

- Terdapat 2 jenis latch yaitu S-R dan D
- S-R Latch, atau yang bisa disebut Set(S) Reset(R) ini dapat merubah dan mempertahankan input yang ada. Seperti behavior yang ada dibawah ini

S	R	Q	Q+	Ket.
0	0	0	0	No Change
0	1	0	0	Reset
1	0	0	1	Set
1	1	Invalid		Both Go Low

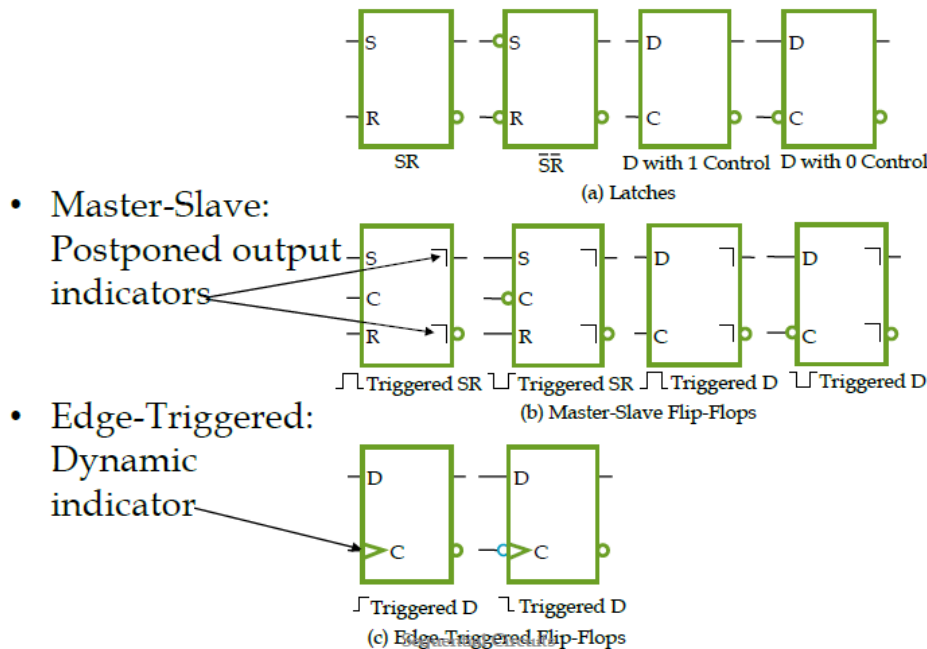
- D Latch, atau yang disebut Data Latch memiliki masukan berupa D. D latch dapat menghindari invalid output seperti saat keadaan input S-R Latch 1 – 1. Berikut behavior dari D Latch

Q	D	Q+	Ket.
0	0	No Change	Mempertahankan keadaan sebelumnya
0	1	1	Set
1	0	0	Clear
1	1	No Change	Mempertahankan keadaan sebelumnya

2.3 Flip Flops



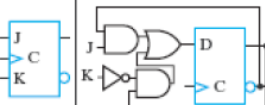

- Timing Problems, terdapat 2 jenis timing yaitu Master Slave dan Edge-Triggered. Perbedaannya adalah kondisi dari setiap jenis. Ketika kondisi timing adalah master slave maka ketika clock bernilai 1 maka perubahan akan selalu terjadi selama clock 1. Berbeda dengan edge triggered yang terbagi menjadi 2 yaitu positive dan negative. Positive edge triggered adalah perubahan next state saat kondisi clock naik menjadi 1, negative adalah kebalikannya, next state dapat berubah saat clock turun menjadi 0. **PERHATIKAN CLOCK, APAKAH MASTER SLAVE / EDGE TRIGGERED / POSITIVE / NEGATIVE**

Standard Symbols for Storage Elements



30

- Flip Flop Behavior

Type	Symbol	Logic Diagrams	Characteristic Table				Characteristic Equation	Excitation Table				
D		See Figure 5-12	D	Q(t+1)	Operation	$Q(t+1) = D(t)$	Q(t+1)		D	Operation		
			0	0	Reset		0	0	Reset			
			1	1	Set		1	1	Set			
SR		See Figure 5-9	S	R	Q(t+1)	Operation	$Q(t+1) = S(t) + \overline{R(t)} Q(t)$	Q(t)	Q(t+1)	S	R	Operation
			0	0	Q(t)	No change		0	0	0	X	No change
			0	1	0	Reset		0	1	1	0	Set
			1	0	1	Set		1	0	0	1	Reset
			1	1	?	Undefined		1	1	X	0	No change
JK		J	K	Q(t+1)	Operation	$Q(t+1) = J(t) \overline{Q}(t) + \overline{K}(t) Q(t)$	Q(t)	Q(t+1)	J	K	Operation	
		0	0	Q(t)	No change		0	0	0	X	No change	
		0	1	0	Reset		0	1	1	X	Set	
		1	0	1	Set		1	0	X	1	Reset	
		1	1	Q̄(t)	Complement		1	1	X	0	No Change	
T		T	Q(t+1)	Operation	$Q(t+1) = T(t) \oplus Q(t)$	Q(t+1)		T	Operation			
		0	Q(t)	No change		Q(t)	0	No change				
		1	Q̄(t)	Complement		Q̄(t)	1	Complement				

- **Identifikasi Circuit (PR 7)**

Steps:

- Tentukan Variable Input/Output
- Tentukan Persamaan
 - Jenis FF
 - Persamaan input ke FF
- Membuat State Table
 - Buatlah state table berdasarkan persamaan yang sudah didapat sebelumnya, identifikasi next state dan outputnya
- Membuat State Diagram
 - Setelah membuat state table, pindahkan isi dari state table kedalam bentuk State Diagram (**Mealy**/Moore)
- Mencari Functionom
 - Buatlah K-Map dari State Table untuk menentukan functionnya
- Buatlah Clock Pulse (Jika diminta)

- **5 Steps Sequential**

- Spesification
- Table
- Diagram
- Kmap/Functionom
- Mapping & Verif

3. Register

3.1 Pengertian

Register merupakan suatu alat/gate yang dapat digunakan untuk mentransfer maupun menyimpan(load) data.

- Counter
 - Ripple, output dari FF menjadi clock berikutnya
 - Synchronous, 1 Clock untuk semua FF. (Satu untuk semua)

3.2 Register Transfer

$K1 : R2 \leftarrow R1$ Dapat dibaca ketika input K1 aktif/bernilai 1, maka Register 1 (R2) akan mentrasfer data ke Register 2 (R2). Contoh dapat dilihat pada PR 10.

3.3 Register Cell

Penggunaan 1 cell untuk 1 bit. Dapat digunakan untuk memperbanyak diri dibandingkan membuat satu register yang utuh.

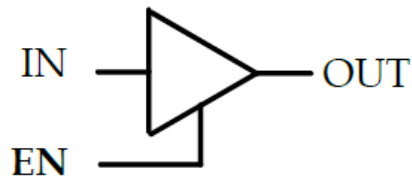
Not Encoded

- Terdapat 3 Control Input yaitu:
 - Load, Shift, Add
 - Nilai 1 hanya boleh ada 1 dalam setiap kondisi seperti untuk 2 bit maka = (0,0) (0,1) (1,0)
- Not Encoded Step:
 - Spesifikasi
 - Data Input
 - Control Input
 - Hold State
 - Register Transfer
 - State Table
 - 2 Dimensi
 - 1 Dimensi

- Kmaps
 - Untuk memudahkan membuat KMaps gunakan table 1 Dimensi
 - Cari Fungsi
- Buat Circuit dari Fungsi

3.4 Three – State Bus

- Symbol



- Truth Table

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

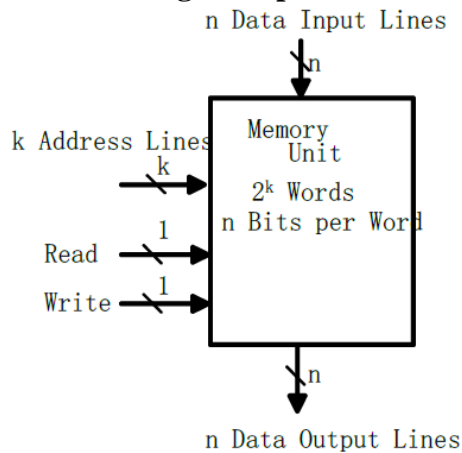
- Fungsi dari buffer adalah untuk memutus/menghentikan arus yang akan melewati buffer. Ketika nilai EN = 0 maka output akan Hi-Z atau tidak ada data yang lewat. Berkebalikan ketika nilai EN = 1, maka IN/Input akan melewati Buffer tanpa merubah data. (Dapat diibaratkan D Flip Flop)

3.5 Serial/Parallel Adder

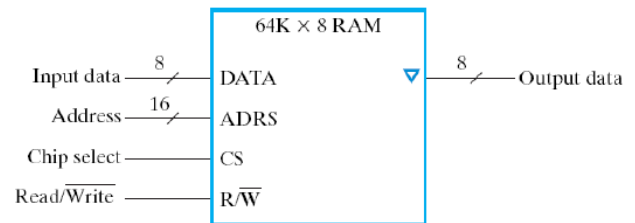
- ☹

4. Memory

4.1 Block Diagram Spesification



Symbol for a 64K x 8 RAM Chip



$64K = 2^{16}$, Maka ADDRESS = 16

64K x 8 RAM, Maka Input/Outputnya = 8

4.2 Larger Memory

Membuat Block Memory yang besar dengan Block Memory yg lebih kecil, seperti:

Buatlah Diagram 256K x 8 RAM dengan 64K x 8 RAM

Steps:

- Anggap 256K x 8 RAM == (A)K x (B) RAM
- Bagi (A1) dengan (A2). Keterangan: A1 adalah Diagram besar A2 adalah block diagram yang lebih kecil. $256 / 64 = 4$. Nilai 4 ini akan menjadi barisnya. (Sebenarnya bebas untuk membuat 4 ini menjadi baris/kolom, asalkan konsisten dalam membuat circuitnya)
- Bagi (B1) dengan (B2). Maka $8 / 8 = 1$. Nilai 1 ini akan menjadi kolomnya. (Ketika menjadikan 1 ini kolom maka nilai 4 di langkah ke-2 hrs menjadi baris. Konsisten membuat baris dan kolom, langkah ke-2 dan ke-3 saling berkebalikan).
- Buatlah decoder sebagai CS (Chip Select) atau yang biasa diketahui sebagai Enable. Ukuran decoder adalah n-to-step2. Maksud dari ini adalah ketika hasil pembagian seperti contoh step 2 adalah 4 maka decodernya adalah 2 to 4. Keluaran dari decodernya menuju ke CS setiap row diagram. Sehingga ketika decoder bekerja, satu baris Block Diagram akan menyala.
- Buatlah R/W (Read/Write) yang menuju ke setiap block memory.

- Buatlah Address dan Data input/output dari block diagram sesuai yang telah di pelajari pada Materi 4.1 yaitu Block Diagram Spesification
- Jangan lupa buat **BUS** di Outputnya
- Contoh, Membuat 256K x 32 RAM dengan 64K x 8 RAM

