Science and Technology English II

Exercise 209 “System” Meiji University 2021

EX_209_21.pptx 18 Slides December 7th.,2021

http://mikami.a.la9.jp/mdc/mdc1.htm

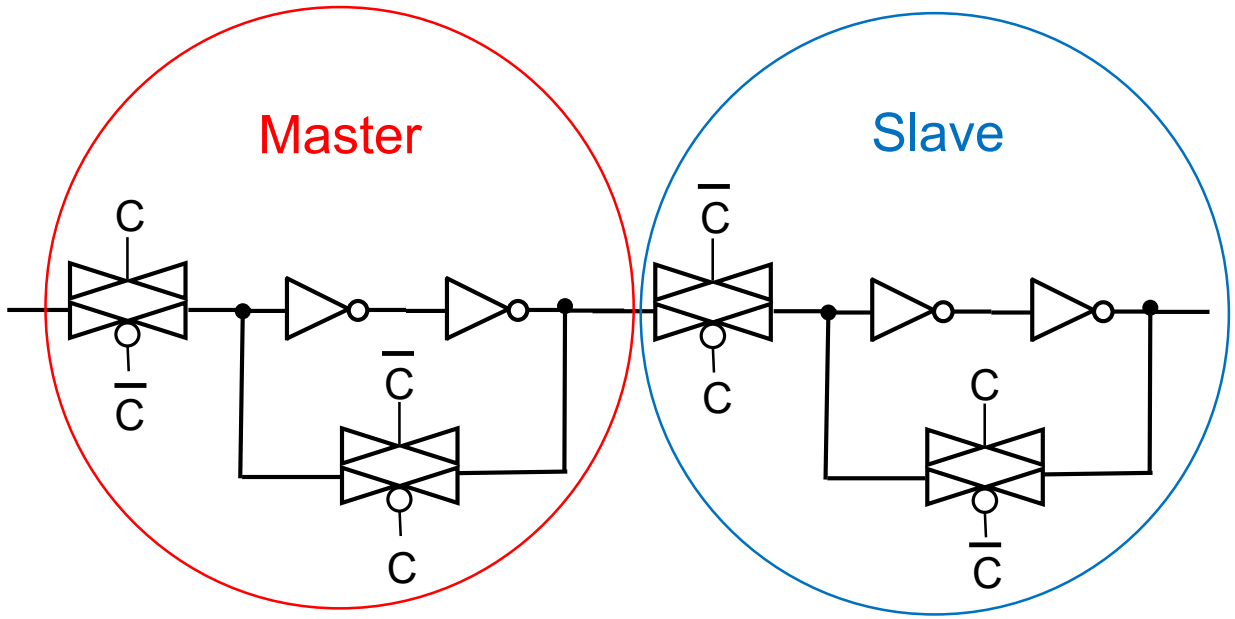
Renji Mikami

Renji_Mikami(at_mark)nifty.com [mikami(at_mark)meiji.ac.jp]

Day 208 Review

- Latch (非同期)から Flip-Flop (同期)までの変移
- 近代の設計は同期式設計ルールを前提としている

Clock		T4	T3	T2	T1	
DIN	D	C	C	B	B	
M T-Gate		閉	開	閉	開	閉
M Latch		C	C	B	B	—
S T-Gate		開	閉	開	閉	開
S Latch		C	B	B	—	—
FF-OUT		C	B	B		

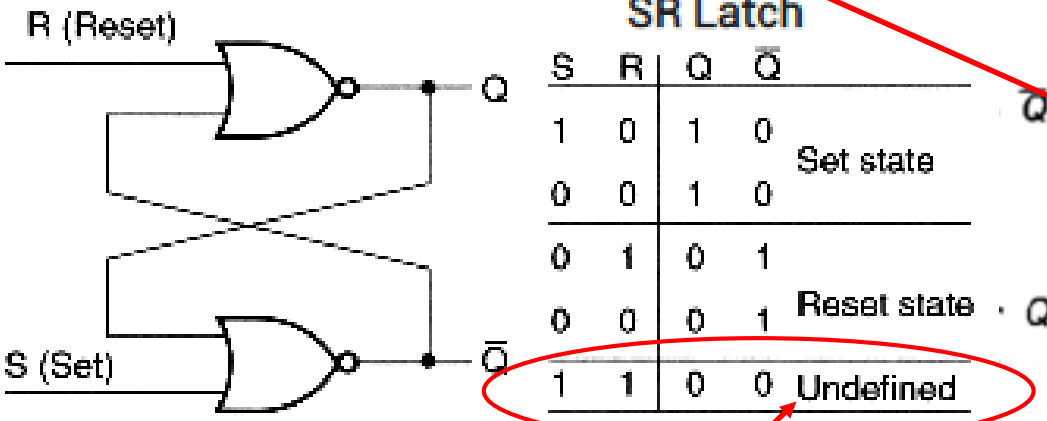


EX_208-1

- SR Latch

Set (1) とReset (0) ができるラッチしかし S と R が同時に 1 になってはいけない(禁止)

inhibit



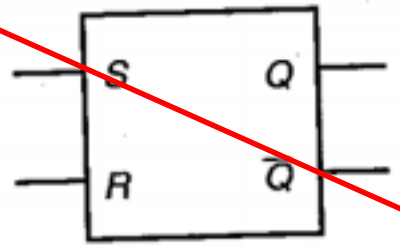
(a) Logic diagram

SR Latch

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	
0	0	0	1	Reset state
1	1	0	0	Undefined

(b) Function table

Figure 6.3 NOR-based SR flip-flop.



(b) Logic symbol

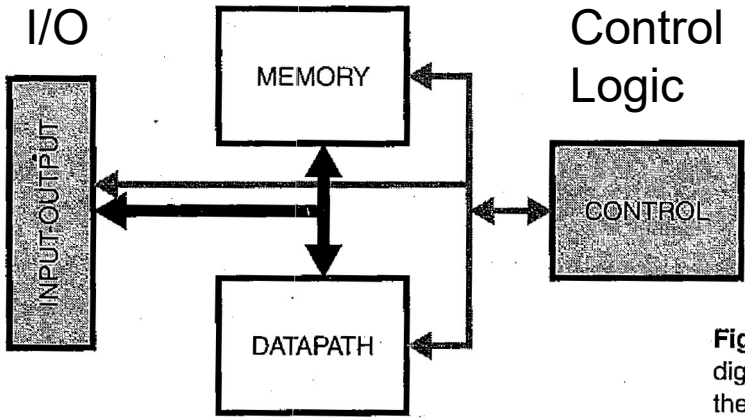
S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

(c) Characteristic table

inhibit

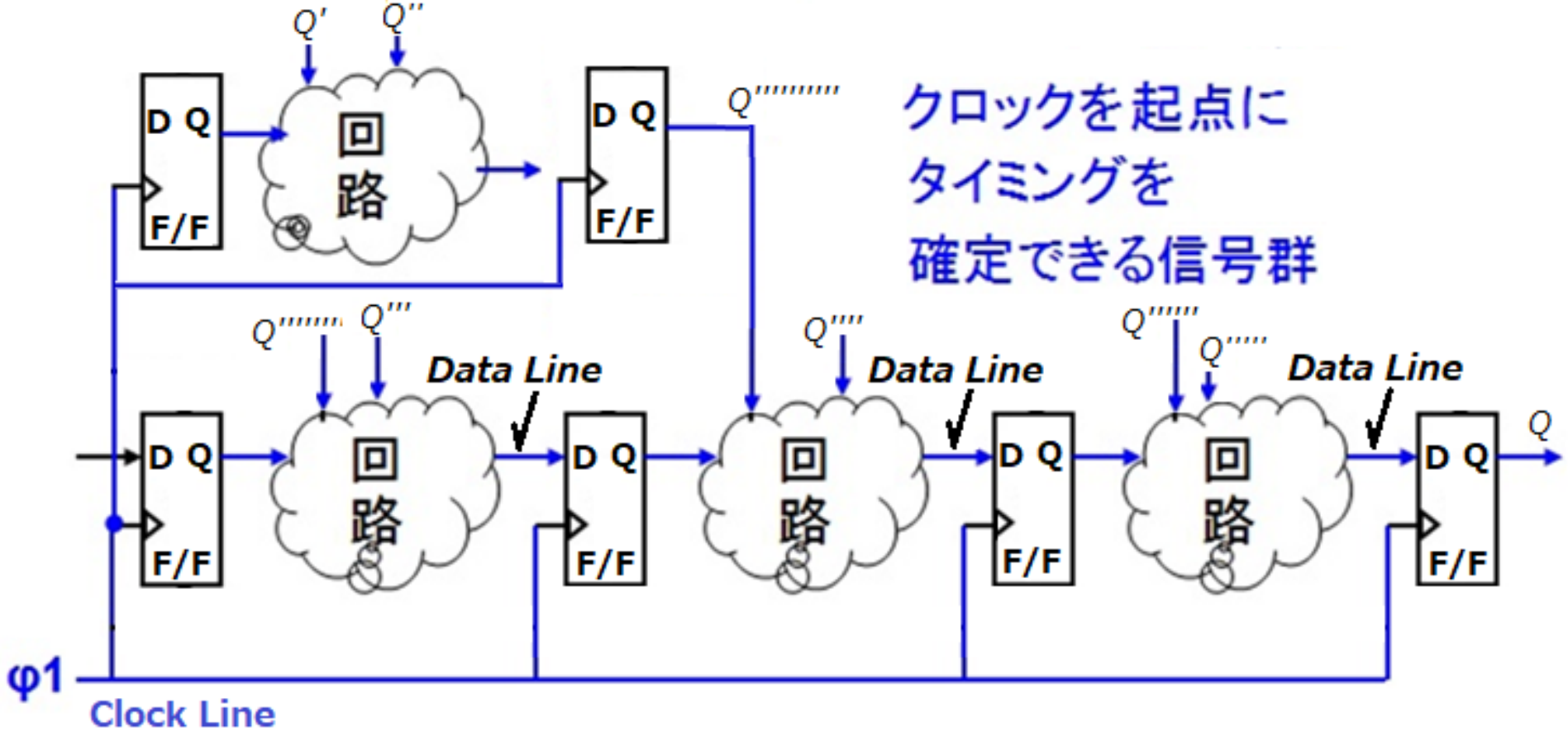
Day 209 Perspective

- 計算機 (Computer) を考えるポイント
- 計算機の基本 – チューリングマシン-ノイマン型コンピュータ
  - Instruction Flow – Sequential (<->Not Dataflow)
- MPUの構成要素
  - Memory, *Data Path*, Control Logic, I/O
- 計算機ハードの高速化
  - Path Delay Minimization
    - 信号経路の遅延時間の削減
  - Clock Up
    - 動作速度の高速化
  - Pipeline
    - 演算回路の多段化による並行処理



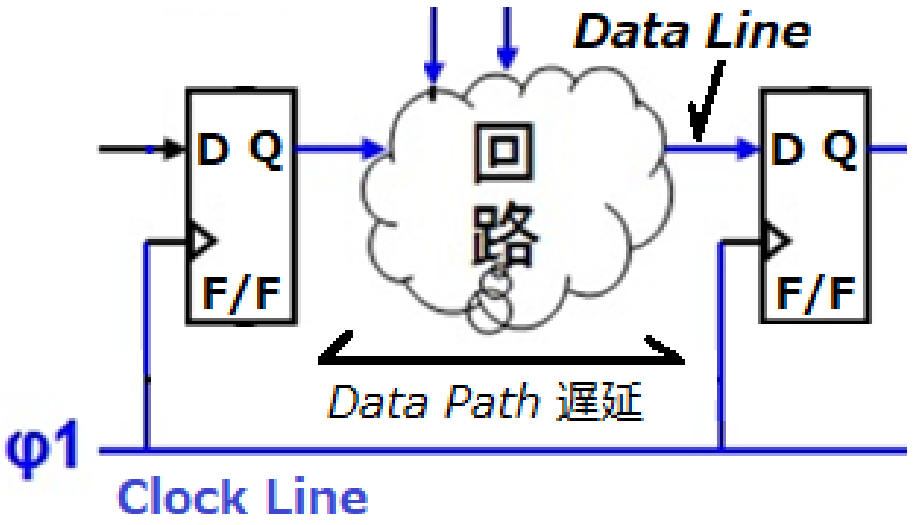
近代システムの同期系

- 同期回路は Clock Line と Data Line から 構成される
- どこかの F/F 出力が回路に入力される構成ではタイミングが確定



Path Delay Minimization

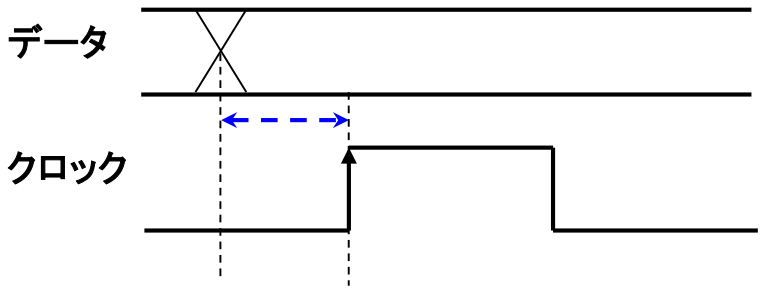
- Data Line の信号は、F/F の Setup Time 以内に届かないといけない
- クロック・ラインの Clock Speed が系の処理速度を決める
- Clock Speed は、その系で最大の遅延のデータラインに制約される
- 系の中の最大の遅延が発生するパスをクリティカル・パスという



Carry 生成回路がクリティカル・パスになりやすい

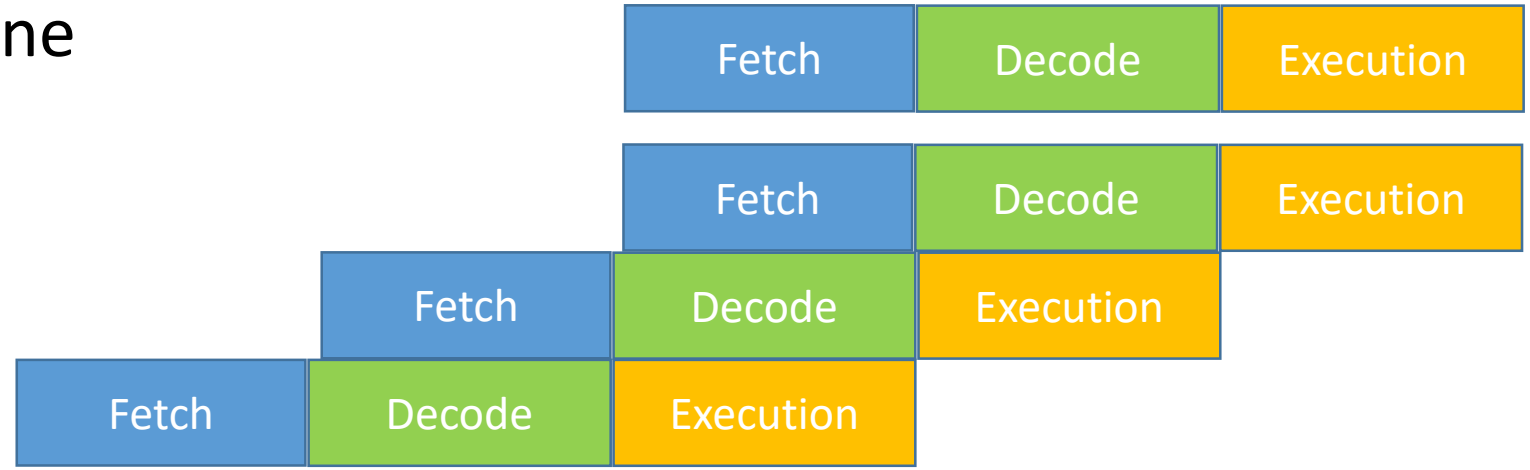
- **セットアップ時間（SETUP）** *Review EX_207*

クロックの立上り（または立下がり）のエッジの前にデータが確定していなければいけない時間



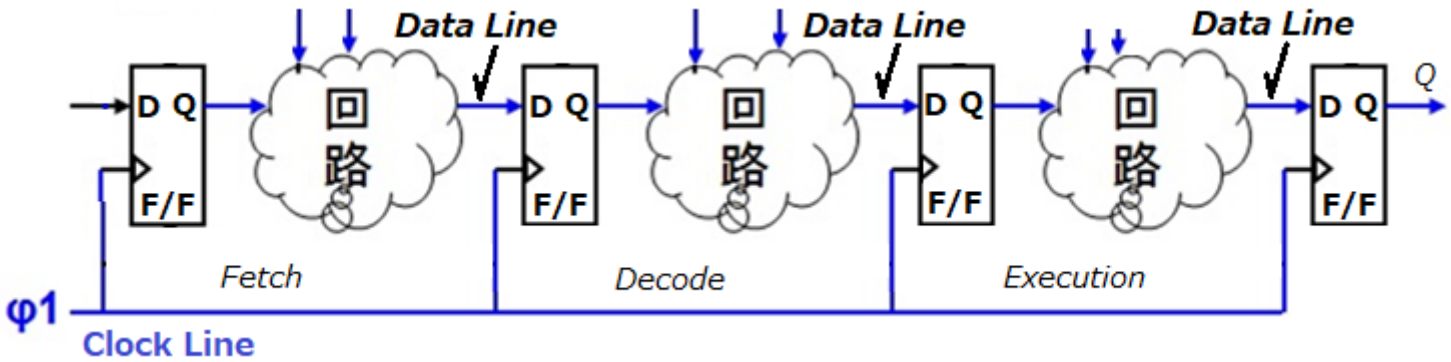
Pipeline

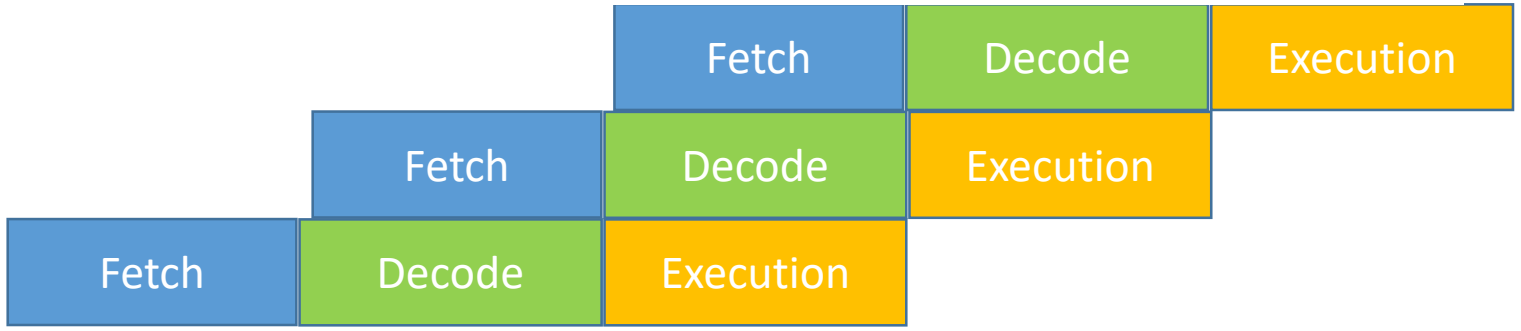
- MPU の基本動作は Fetch – Decode – Execution
  - Fetch Cycle : Memory から命令データを持ってくる
  - Decode Cycle : 命令データをデコード(解読)する
  - Execution Cycle : デコードした命令を実行する
  - このサイクルが終了後、次の命令をもってくる(Fetch Cycleの実行)
- このサイクルを並行して行うのがパイプライン
- Non Pipeline
- Pipeline



Pipeline

- Non Pipeline では1連の命令が3サイクル実行時間で処理
- Pipeline では、1サイクルごとに次の命令をFetchして処理
- 3 Stage Pipeline では、x3 倍の処理能力 !?





Pipiline Hazard

*データパス回路*で 3 Stage 処理が終わる前に次の命令を先読みFetchするが、条件分岐などで異なる命令アドレスに飛ぶと先読みした命令がキャンセルになる。パイプラインに再度命令をセットしなおすことになる。乱れるパイプラインという。

この制御は、*コントロールロジック回路*で行う

Adder

- 数を正負の整数とし、加算、減算、乗算を考える
- 整数を Binary で表現、2の補数とすれば、負の数を扱える
- これで減算は、負の数の加算で置き換えできる
- 乗算は加算の繰りかえしで置き換えできる
- よって加算器が基本となる

(乗算はシフト操作により高速に計算できる)

- Adder には、Half Adder (半加算器)と Full Adder (全加算器)があり、Carry - キャリー(桁上げ)によって多ビット化できる。

N(4)bit Adderの記述と階層設計(VHDL)

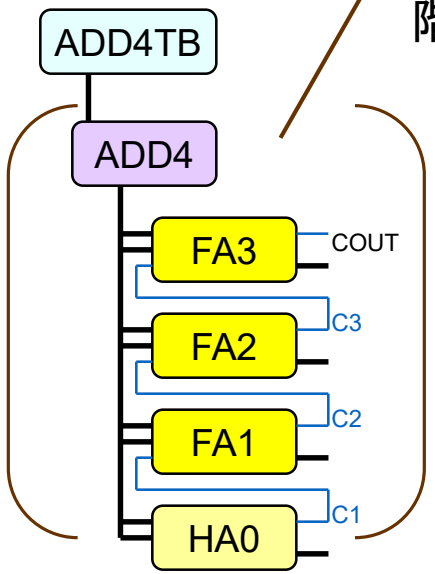

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ADD4 is
   port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
          B : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (3 downto 0);
          COUT : out  STD_LOGIC);
end ADD4;

architecture Behavior of ADD4 is
        signal C1, C2, C3 : std_logic;

component HA
        port (X,Y : in  STD_LOGIC;
              S,C : out  STD_LOGIC);
               end component;
component FA
        port (A, B, CI : in  STD_LOGIC;
        SUM,CO : out  STD_LOGIC);
               end component;

begin
        HA0 : HA port map(X => A(0) , Y => B(0) , S => Y(0)  ,C => C1);
        FA1 : FA port map(A => A(1), B => B(1), CI => C1, SUM => Y(1)  , CO => C2);
        FA2 : FA port map(A => A(2), B => B(2), CI => C2, SUM => Y(2)  , CO => C3);
        FA3 : FA port map(A => A(3), B => B(3), CI => C3, SUM => Y(3)  , CO => COUT);
end Behavior;
```

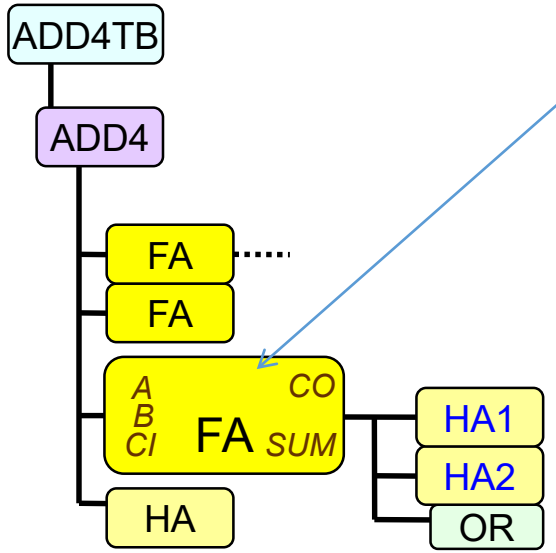


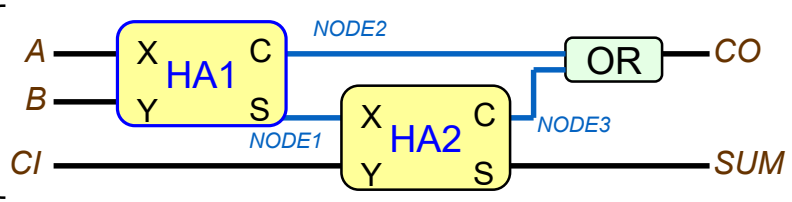
ADD4の
VHDL
階層記述

ソースを見ると、
論理機能の記述が
ありません。
HA0とFA1から
FA3までの接続
だけが記述して
あります。
代入演算子は、
=>ですから、
コンカレント
記述です。
つまり
インスタンス
の接続だけを
示しています。
FAの機能は、
さらに下位階層
で記述します。

トポロジー記述です
I/O, Instance,
Network で
構成されています。

Full Adder の階層設計





FA(Full Adder) をさらに
階層化し、下の階層で
記述するHA(Half Adder)
の接続と or (論理和)で
記述している。
ここで、HA1とHA2は
インスタンスの接続と
論理演算で記述している

```
entity FA is
    Port ( A,B,CI : in  STD_LOGIC;
           SUM,CO : out  STD_LOGIC);
end FA;

architecture Behavior of FA is
                signal NODE1, NODE2, NODE3 : std_logic;
                component HA
                                port (X,Y : in  STD_LOGIC;
        S,C : out  STD_LOGIC);
                end component;
begin
                HA1 : HA port map(X =>A, Y => B, S =>NODE1, C =>NODE2);
                HA2 : HA port map(X =>NODE1, Y => CI, S => SUM, C
=>NODE3);
                CO <= NODE2 or NODE3;
end Behavior;
```

トポロジー記述です
I/O, Instance,
Network で
構成されています。

EX209 英語解説

- 原本を読み解いていきます。
- ホームページにある原本DICS Chapter7 (DICS_207.pdf)の赤線部を速読してください。
- この原本は、コピペができません。(勉強用にコピペする場合は、打ち直ししたEX_209.pdf 内の文を使ってください。)

- レポートは自分の言葉でたくさん書いてください

Exercise: EX_209

- ホームページにある原本 DICS Chapter 7 DICS_207.pdf http://mikami.a.la9.jp/meiji/ste/DICS_207.pdf 原本の以下指定ページ(資料配布の場合は、その資料を使用)を速読し設問に和文または英文で答えよ。(本資料の文はコピペできるので翻訳サイトを使ってもよいが、その前に必ずDICS_207.pdf の原本英文を通読すること。)
- EX_209-1 : P384 STE-102-701 Line 32 ~ P385 STE-102-702 Line 19まで
  - Datapaths in Digital Processor Architectures ~
  - この範囲を読んで要約してください。
- EX_209-2 : P386 STE-102-703 Line 11~14 P388 STE-102-705 Line 1
  - 7.3.1 The Binary Adder: Definitions~
  - この範囲を読んで要約してください。

•提出はClass Web “レポート” にて木曜まで

•毎回のレポートは、最低A4 1ページ以上は書いてください。余白には、今回の授業の内容、資料についての感想や要望を記入してください。

EX_209-1(1) : P384 STE-102-701 Line 32 ~ P385 STE-102-702 Line 3

このpdfはコピペできます

- Datapaths in Digital Processor Architectures
- An analysis of the components of a simple processor puts the different classes of digital circuits and their usage in perspective. Such a processor could be the brain of a personal computer (PC) or the heart of a compact disc player. A typical block diagram is shown in Figure 7.1 and is composed of a number of building blocks that occur in one form or another in almost every digital processor.
- The datapath is the core of the processor; it is where all computations are performed. The other blocks in the processor are support units that store either the results produced by the datapath or help to determine what will happen in the next cycle. A typical datapath consists of an interconnection of basic combinational functions, such as logic (AND, OR, EXOR) or arithmetic operators (addition, multipli- cation, comparison, shift). Intermediate results are stored in registers. The design of the arithmetic operators is the topic of this chapter.

EX_209-1(2) : P385 STE-102-702 Line 4 ~ 19

- The control module determines what actions happen in the processor at any given point in time. A controller can be viewed as a finite state machine (FSM). It consists of registers and logic, and is hence a sequential circuit. The logic can be implemented in different ways—either as an interconnection of basic logic gates, often called random logic, or in a more structured fashion using programmable logic arrays (PLAs) and instruction memories.
- The memory module serves as centralized data storage. A broad range of different memory classes exist. The main difference between those classes is in the way data can be accessed, such as read-only versus read-write, sequential versus random access, or single-ported versus multiported access. Another way of differentiating between memories is related to their data-retention capabilities. Dynamic memory structures must be refreshed periodically to keep their data, while static memories keep their data as long as the power source is turned on. Finally, memory structures such as flash memories conserve the stored data even when the supply voltage is removed. A single processor might combine different memory classes. For instance, random access memory can be used to store data and read-only memory to store instructions.

EX_209-2(1) : P386 STE-102-703 Line 11~ P387 STE-102-704 ~Line 8

- 7.3.1 The Binary Adder: Definitions
- The truth table of a binary full adder is given in Table 7.1. A and B are the adder inputs, Ci is the carry input, S is the sum output, and Co is the carry output.The Boolean expressions for S and Co are given in Eq. (7.1).
- Table 7.1 Truth table for full adder.

$$S = A \text{ xor } B \text{ xor } Ci = A \text{ notB notCi} + \text{notA } B \text{ notCi} + \text{notA notB } Ci + ABCi$$

$$Co = AB + BCi + ACi$$

It is often useful from an implementation perspective to define *S* and *Co* as functions of some intermediate signals *G (Generate), D (Delete),* and *P (Propagate). G = 1 (D = 1)* ensures that a carry bit will be *generated (deleted)* at *Co* independent of *Cq*, while *P = 1* guarantees that an incoming carry will propagate to *Co*. Expressions for these signals can be derived from inspection of the truth table.

EX_209-2(2) : P387 STE-102-704 Line 9 ~

*G = AB, D = not A * notB , P = A + B (or P = A xor B)*

S and *Co* can be rewritten as functions of *P* and *G* (or *D*)

このpdfはコピペできます

Co(G,P) = G + PCi, S(G,P) = P xor Ci

- Notice that *G* and *P* are only functions of *A* and *B* and are not dependent upon *Ci*. In a similar way, we can also derive expressions for *S(D,P)* and *Co(D,P)*.
- An *N*-bit adder can be constructed by cascading *N* full-adder circuits in series, connecting $C_{o,k-1}$ to $C_{i,k}$ for *k* = 1 to *N*-1, and the first carry-in $C_{i,0}$ to 0 (Figure 7.3). This configuration is called a *ripple-carry adder* since the carry bit "ripples" from one stage to the other. The delay through the circuit depends upon the number of logic stages that must be traversed and is a function of the applied input signals. For some input signals, no rippling effect occurs at all, while for others the carry has to ripple all the way from the *least- significant bit (lsb)* to the *most-significant bit (msb).* The propagation delay of such a structure (also called the *critical path*) is defined as the *worst-case delay over all possible input patterns.*
- In the case of the ripple-carry adder, the worst-case delay happens when a carry generated at the least significant bit position propagates all the way to the most significant bit. The delay is then proportional to the number of bits in the input words *N* and is approximated by Eq. (7.4).

$$t_{adder} = (N\text{-}1)t_{carry} + t_{sum} \quad (7.4)$$

Memo

フォローアップURL (Revised)

http://mikami.a.la9.jp/meiji/MEIJI.htm

担当講師

三上廉司(みかみれんじ)

Renji_Mikami(at_mark)nifty.com

mikami(at_mark)meiji.ac.jp (Alternative)

http://mikami.a.la9.jp/_edu.htm