มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

การสอบปลายภาคการเรียนที่ 1 ปีการศึกษา 2550

วิชา ENE 231 Digital Circuit and Logic Design วิศวกรรมอิเล็กฯ ปีที่ 2

สอบวันศุกร์ที่ 12 ตุลาคม 2550 **เวลา** 13.00-16.00 น.

คำสั่ง

1. ข้อสอบมีทั้งหมด 7 ข้อ 3 หน้า (รวมใบปะหน้า) คะแนนรวม 120 คะแนน
2. ให้ทำข้อสอบทุกข้อลงในสมุดคำตอบ
3. ห้ามนำเอกสารใด ๆ เข้าห้องสอบ
4. ไม่อนุญาตให้นำเครื่องคำนวณใด ๆ เข้าห้องสอบ
5. มีทฤษฎีต่างของ Switching Algebra, VHDL syntax และ Flip-flop Characteristic Equations จำนวน 7 หน้า

เมื่อนักศึกษาทำข้อสอบเสร็จ ต้องยกมือบอกกรรมการคุมสอบ
เพื่อขออนุญาตออกนอกห้องสอบ
ห้ามนักศึกษานำข้อสอบและกระดาษคำตอบออกนอกห้องสอบ
นักศึกษาซึ่งทุจริตในการสอบอาจถูกพิจารณาโทษสูงสุดให้พ้นสภาพการเป็นนักศึกษา

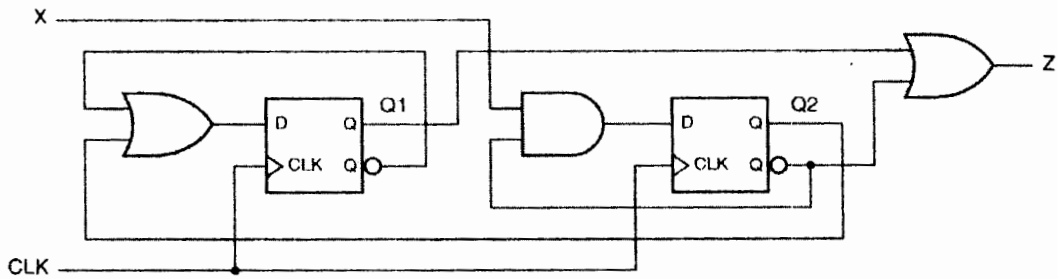
ชื่อ-สกุล.....รหัสนักศึกษา.....

(ผศ. ดร. พินิจ กำหอม)
ผู้ออกข้อสอบ
โทร. 0-2470-9075

ข้อสอบนี้ได้ผ่านการประเมินจาก
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ ฯแล้ว

(ผศ.ดร. วุฒิชัย อัศวินชัยโชติ)
หัวหน้าภาควิชาวิศวกรรมอิเล็กทรอนิกส์และโทรคมนาคม

1. วิเคราะห์วงจร synchronous finite state machine ในรูปข้างล่าง โดยให้เขียน excitation equations, excitation/transition table, และ state/output table (20 คะแนน)



2. ออกแบบวงจร synchronous finite state machine ที่มี state/output table ข้างล่างโดยให้ใช้ positive-edge-trigger D flip-flop ในการออกแบบและให้ใช้ state assignment ดังนี้ A=00, B=01, C=11, D=10

	X		Z
S	**0**	**1**	
A	B	D	0
B	C	B	0
C	B	A	1
D	B	C	0
	S*		

ให้แสดงผลของออกแบบในแต่ละขั้นตอนได้แก่ transition table, excitation table, excitation equations ที่ลดรูปแล้ว และ output equations ที่ลดรูปแล้ว (25 คะแนน)

3. ให้ออกแบบ state/output table **หรือ** state diagram ของ synchronous FSM ที่ทำหน้าที่ตามข้อกำหนดดังนี้ พร้อมเขียน VHDL อธิบาย (25 คะแนน)
   เอ้าท์พุทของระบบจะเป็น '1' ถ้าอินพุท A ของมันใน 2 สัญญาณนาฬิกาก่อนมีค่าเป็น 00 หรือ 11

4. เขียน VHDL สำหรับ 8-bit 4-input MUX (7 คะแนน)

5. เขียน VHDL สำหรับ 8-bit Registers ที่มีขา asynchronous reset (8 คะแนน)

6. สร้างวงจรบวกเลข 2's complement 4 บิทโดยนอกจากผลบวกแล้วให้มีเอ้าท์พุท overflow_flag เพื่อแสดงว่าค่าเอ้าท์พุท overflow หรือไม่ (15 คะแนน)

7. จากตารางความจริงข้างล่าง

Row #	Inputs A B C D	Output F
0	0 0 0 0	1
1	0 0 0 1	1
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	1
5	0 1 0 1	0
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	- (don't care)
9	1 0 0 1	0
10	1 0 1 0	0
11	1 0 1 1	1
12	1 1 0 0	1
13	1 1 0 1	1
14	1 1 1 0	0
15	1 1 1 1	1

(a.) ให้หา minimal POS โดยใช้ Quine-McCluskey (แสดงขั้นตอนการทำ) (15 คะแนน)

(b.) หา complexity ของวงจรที่ได้จากข้อ (a.) (5 คะแนน)

Switching Algebra Postulates and Theorems

1. Closure Properties
   a. **Postulate 1a (P1a):** If X and Y are in the domain, that is, take on only the values {0,1}, then (X+Y) is also in the domain.
   b. **Postulate 1b (P1b):** If X and Y are in the domain, that is, take on only the values {0,1}, then (X·Y) is also in the domain.
2. Identity Properties
   a. **Postulate 2a (P2a):** $X + 0 = X$
   b. **Postulate 2b (P2b):** $X \cdot 1 = X$
3. Commutative Properties
   a. **Postulate 3a (P3a):** $X + Y = Y + X$
   b. **Postulate 3b (P3b):** $X \cdot Y = Y \cdot X$
4. Distributive Properties
   a. **Postulate 4a (P4a):** $X + (Y \cdot Z) = (X+Y).(X+Z)$
   b. **Postulate 4b (P4b):** $X \cdot (Y+Z) = X \cdot Y + X.Z$
5. Complement Properties
   a. **Postulate 5a (P5a):** $X + \overline{X} = 1$
   b. **Postulate 5b (P5b):** $X \cdot \overline{X} = 0$

Theorems

1. Involution Theorem
   Theorem 1 (T1): $\overline{\overline{X}} = X$
2. Identity Theorems
   a. **Theorem 2a (T2a):** $X + 1 = 1$
   b. **Theorem 2b (T2b):** $X \cdot 0 = 0$
3. Idempotency Theorems
   a. **Theorem 3a (T3a):** $X + X = X$
   b. **Theorem 3b (T3b):** $X \cdot X = X$
4. Associative Theorems
   a. **Theorem 4a (T4a):** $X + (Y + Z) = (X + Y) + Z$
   b. **Theorem 4b (T4b):** $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
5. DeMorgan's Theorems
   a. **Theorem 5a (T5a):** $\overline{X + Y} = \overline{X} \cdot \overline{Y}$
   b. **Theorem 5b (T5b):** $\overline{X \cdot Y} = \overline{X} + \overline{Y}$
6. Adjacency Theorems
   a. **Theorem 6a (T6a):** $X \cdot Y + X \cdot \overline{Y} = X$
   b. **Theorem 6b (T6b):** $(X + Y) \cdot (X + \overline{Y}) = X$
7. Absorption Theorems
   a. **Theorem 7a (T7a):** $X + X \cdot Y = X$
   b. **Theorem 7b (T7b):** $X \cdot (X+Y) = X$
8. Simplification Theorems
   a. **Theorem 8a (T8a):** $X + \overline{X}.Y = X + Y$
   b. **Theorem 8b (T8b):** $X \cdot (\overline{X} + Y) = X \cdot Y$
9. Consensus Theorems
   a. **Theorem 9a (T9a):** $X \cdot Y + \overline{X} \cdot Z + Y \cdot Z = X \cdot Y + \overline{X} \cdot Z$
   b. **Theorem 9b (T9b):** $(X+Y) \cdot (\overline{X}+Z) \cdot (Y+Z) = (X+Y) \cdot (\overline{X}+Z)$

VHDL Syntax ที่ต้องจำเป็นต้องใช้

1. Entity declaration

```
entity entity-name is
  port (sigal-names: mode signal-type;
        sigal-names: mode signal-type;
         ...
        sigal-names: mode signal-type);
end entity-name;
```

Example:

```
entity Inhibit is
    port (X,Y: in BIT;
             Z: out BIT);
end Inhibit;
```

2. Architecture Declaration

```
architecture architecture-name of entity-name is
  type declarations
  signal declarations
  constant declarations
  function definitions
  procedure definitions
  component declarations
begin
  concurrent statement;
        ...
  concurrent statement
end architecture-name;
```

3. Library declaration

```
library library_name;
use library_name.package_name_name.all;
```

Example:

```
library ieee;
use ieee.std_logic_1164.all;
```

4. Type and constant declarations

```
type type-name is (value-list);

subtype subtype-name is type-name start to end;
subtype subtype-name is type-name start downto end;
constant constant-name : type-name := value;
```

5. Signal declarations

```
signal signal_names: signal-type;
```

6. Component Declarations and Instantiations

Component Declarations

```
component component-name
  port (signal-names : mode signal-type;
        signal-names : mode signal-type;
        ...
        signal-names : mode signal-type);
end component;
```

Component Instantiations

```
label: component-name port map(signal1, signal2, ..., signaln);

label: component-name port map(port1=>signal1, port2=>signal2, ..., portn=>signaln);
```

7. Concurrent/Conditional Signal Assignments

```
signal-name <= expression;

signal-name <= expression when boolean-expression else
               expression when boolean-expression else
               ...
               expression when boolean-expression else
               expression;
```

8. Select Signal Assignments

```
with expression select
  signal-name <= signal-value when choices,
                 signal-value when choices,
                 ...
                 signal-value when choices;
```

9. Process Syntax

```
process (signal-name, signal-name, ..., signal-name)
  type declarations
  variable declarations
  constant declarations
  function definitions
  procedure definitions
begin
  sequential-statement
  ...
  sequential-statement
end process;
```

10. Sequential Statements

10.1 If statement syntax

```
if boolean-expression then sequential-statements
end if;

if boolean-expression then sequential-statements
else sequential-statements
end if;

if boolean-expression then sequential-statements
elsif boolean-expression then sequential-statements
...
elsif boolean-expression then sequential-statements
end if;

if boolean-expression then sequential-statements
elsif boolean-expression then sequential-statements
...
elsif boolean-expression then sequential-statements
else sequential-statements
end if;
```

10.2 Case statement syntax

```
case expression is
  when choices => sequential-statements
  ...
  when choices => sequential-statements
end case;
```

10.3 Loop, for loop, and while loop

```
loop
  sequential-statement
  ...
  sequential-statement
end loop;
```

```
for identifier in range loop
  sequential-statement
  ...
  sequential-statement
end loop;
```

```
while boolean-expression loop
  sequential-statement
  ...
  sequential-statement
end loop;
```

11. Example Codes

11.1 Structural Style

```
library IEEE;
use IEEE.std_logic_1164.all;
library unisim;
use unisim.vcomponents.all;

entity prime is
    port ( N: in STD_LOGIC_VECTOR (3 downto 0);
           F: out STD_LOGIC );
end prime;

architecture prime1_arch of prime is
signal N3_L, N2_L, N1_L: STD_LOGIC;
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;
component INV port (I: in STD_LOGIC; O: out STD_LOGIC); end component;
component AND2 port (I0,I1: in STD_LOGIC; O: out STD_LOGIC); end component;
component AND3 port (I0,I1,I2: in STD_LOGIC; O: out STD_LOGIC); end component;
component OR4 port (I0,I1,I2,I3: in STD_LOGIC; O: out STD_LOGIC); end component;
begin
  U1: INV port map (N(3), N3_L);
  U2: INV port map (N(2), N2_L);
  U3: INV port map (N(1), N1_L);
  U4: AND2 port map (N3_L, N(0), N3L_N0);
  U5: AND3 port map (N3_L, N2_L, N(1), N3L_N2L_N1);
  U6: AND3 port map (N2_L, N(1), N(0), N2L_N1_N0);
  U7: AND3 port map (N(2), N1_L, N(0), N2_N1L_N0);
  U8: OR4 port map (N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0, F);
end prime1_arch;
```

11.2 Dataflow Style 1 Using Conditional Signal Assignment

ใช้ entity เดียวกับ 11.1

```
architecture prime3_arch of prime is
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;
begin
  N3L_N0     <= '1' when N(3)='0' and N(0)='1' else '0';
  N3L_N2L_N1 <= '1' when N(3)='0' and N(2)='0' and N(1)='1' else '0';
  N2L_N1_N0  <= '1' when N(2)='0' and N(1)='1' and N(0)='1' else '0';
  N2_N1L_N0  <= '1' when N(2)='1' and N(1)='0' and N(0)='1' else '0';
  F <= N3L_N0 or N3L_N2L_N1 or N2L_N1_N0 or N2_N1L_N0;
end prime3_arch;
```

Dataflow Style 2 using "with select"

ใช้ entity เดียวกับ 11.1

```
architecture prime4_arch of prime is
begin
  with N select
    F <= '1' when "0001",
         '1' when "0010",
         '1' when "0011" | "0101" | "0111",
         '1' when "1011" | "1101",
         '0' when others;
end prime4_arch;
```

11.4 Behavioral Style

```
architecture prime7_arch of prime is
begin
  process(N)
    variable NI: INTEGER;
  begin
    NI := CONV_INTEGER(N);
    if NI=1 or NI=2 then F <= '1';
    elsif NI=3 or NI=5 or NI=7 or NI=11 or NI=13 then F <= '1';
    else F <= '0';
    end if;
  end process;
end prime7_arch;
```

```
architecture prime8_arch of prime is
begin
  process(N)
  begin
    case CONV_INTEGER(N) is
      when 1 => F <= '1';
      when 2 => F <= '1';
      when 3 | 5 | 7 | 11 | 13 => F <= '1';
      when others => F <= '0';
    end case;
  end process;
end prime8_arch;
```

```
entity prime9 is
    port ( N: in STD_LOGIC_VECTOR (15 downto 0);
           F: out STD_LOGIC );
end prime9;

architecture prime9_arch of prime9 is
begin
  process(N)
  variable NI: INTEGER;
  variable prime: boolean;
  begin
    NI := CONV_INTEGER(N);
    prime := true;
    if NI=1 or NI=2 then null; -- take care of boundary cases
    else for i in 2 to 253 loop
           if (NI mod i = 0) and (NI /= i) then
             prime := false; exit;
           end if;
         end loop;
    end if;
    if prime then F <= '1'; else F <= '0'; end if;
  end process;
end prime9_arch;
```

Flip-Flop Characteristic Equations

Device Type	*Characteristic Equation*
S-R latch	$Q^* = S + R' \cdot Q$
D latch	$Q^* = D$
Edge-triggered D flip-flop	$Q^* = D$
D flip-flop with enable	$Q^* = EN \cdot D + EN' \cdot Q$
Master/slave S-R flip-flop	$Q^* = S + R' \cdot Q$
Master/slave J-K flip-flop	$Q^* = J \cdot Q' + K' \cdot Q$
Edge-triggered J-K flip-flop	$Q^* = J \cdot Q' + K' \cdot Q$
T flip-flop	$Q^* = Q'$
T flip-flop with enable	$Q^* = EN \cdot Q' + EN' \cdot Q$

From *Digital Design: Principles and Practices*, Fourth Edition, John F. Wakerly, ISBN 0-13-186389-4.
©2006, Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.