1 Boolean Functions (30 points)

Shannon's decomposition provides a technique to re-write a Boolean function of n variables in terms of functions of fewer variables. The general form of Shannon's decomposition can be written as follows

$$f = x.f_x + \bar{x}.f_{\bar{x}} \tag{1}$$

Where f is a Boolean function, x is one of its input variables, and $\bar{x}$ refers to the complement of x. f_x is the co-factor of f with respect to variable x, and is derived by setting $x = 1$ in f. Similarly, $f_{\bar{x}}$ is the co-factor of f with respect to $\bar{x}$, and is derived by setting $x = 0$ in f.

1.1 Shannon's Decomposition (15 points)

Apply Shannon's decomposition to the following function on variable x. Then, apply Shannon's decomposition to f_x and $f_{\bar{x}}$ on variable y, *i.e.*, find f_{xy}, $f_{x\bar{y}}$, $f_{\bar{x}y}$, and $f_{\bar{x}\bar{y}}$. Draw the resulting circuit as a network consisting of AND gates, OR gates, inverters, and 2-to-1 multiplexers. Your answer must include at least one multiplexer.

$$f = \bar{w}\bar{y}z + xz + x\bar{y}\bar{z} + w\bar{x}y\bar{z} \tag{2}$$

1.2 Application of Shannon's Decomposition (15 points)

Explain how any Boolean function can be realized as a tree of multiplexers through the application of Shannon's decomposition.

Write in Exam Book Only

2 Combinational Logic (30 points)

This question deals with minimizing combinational logic functions that are specified as a Sum-of-Products expression (AND-OR circuit).

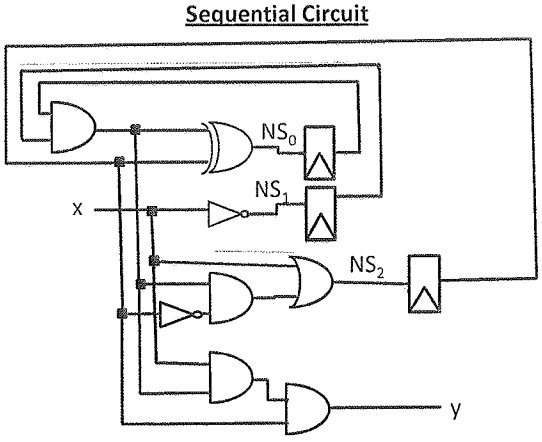
Consider the function $f = x_1x_2x_3 + x_4x_5x_6 + \ldots + x_{3n-2}x_{3n-1}x_{3n}$. The sum-of-products expression provided for the function has n product terms. Answer the following questions.

1. **15 points.** How many product terms are contained in the minimum sum-of-products implementation of f?
2. **15 points.** How many product terms are contained in the minimum sum-of-products implementation of $\bar{f}$ (*i.e.*, the complement of f)?

Write in Exam Book Only

3 Sequential Logic (40 points)

Finite State Machine (FSM) minimization and retiming are two popular approaches to optimizing sequential circuits. This question examines these techniques and the relationship between them.



Logic Equations for Next State and Output

$NS_0 = (PS_0\ PS_1) \oplus PS_2$
$NS_1 = x'$
$NS_2 = PS_0\ PS_1\ PS_2' + x$
$y = PS_0\ PS_1\ PS_2\ x$

State Table

State	$PS_0PS_1PS_2$	$NS_0NS_1NS_2$ x=0	$NS_0NS_1NS_2$ x=1	y x=0	y x=1
S0	000	010	001	0	0
S1	001	110	101	0	0
S2	010	010	001	0	0
S3	011	110	101	0	0
S4	100	010	001	0	0
S5	101	110	101	0	0
S6	110	111	101	0	0
S7	111	010	001	0	1

Figure 1: A sequential circuit and its state table

Consider the sequential circuit shown in Figure 1. The FSM representation of the circuit is shown in the form of a state table. The state table is derived by writing down the logic equations for each next state bit and the output, as shown in the figure. The state table is essentially a combined truth table for these logic equations. In the state table, PS_i is the i-th bit of the present state, and NS_i is the i-th bit of the next state.

Write in Exam Book Only

3.1 FSM minimization (15 points)

It is often desirable to design a state machine with the minimum number of states. FSM minimization is performed by repeatedly merging equivalent states, resulting in a reduced FSM where there are no equivalent states.

Definition: Two states are equivalent if and only if, for any input, the two states have identical outputs and the corresponding next states are equivalent. State equivalence is symmetric, reflexive, and transitive.

Question: Minimize the FSM state table shown in Figure 1 by identifying equivalent states and replacing equivalent states by a single state. Specify the minimized FSM as a state table. You may use symbolic names (*e.g.*, A, B, C, ...) for states in the minimized state table.

3.2 Re-timing (10 points)

Retiming is a structural technique used to optimize sequential circuits without converting them into a finite state machine representation. Retiming consists of moving sequential elements (latches or flip-flops) across gates or fanout points in order to position them at desirable locations so as to minimize a desired metric such as the number of flip-flops or the circuit delay.

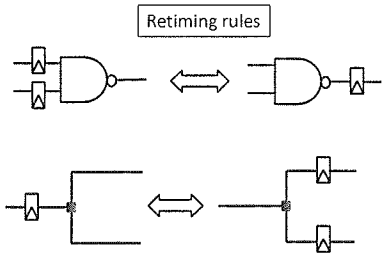


Figure 2: Illustration of retiming

Any legal retiming can be expressed by applying a sequence of transformations to the circuit based on the rules shown in Figure 2 (the NAND gate can be replaced by any other gate).

Question: Retime the circuit shown in Figure 1 such that the number of flip-flops is minimized. Derive the state table for the retimed circuit.

Write in Exam Book Only

3.3 Retiming *vs.* FSM minimization (15 points)

- **5 points.** Compare the state table derived for the retimed circuit with the state table resulting from applying FSM minimization on the original circuit. Do retiming and FSM minimization have the same impact on the number of states for this example?
- **10 points.** In general, can retiming always achieve the same effect as FSM minimization? In other words, given any sequential circuit, can retiming the circuit always result in a circuit with number of states that is the same as what you get by deriving the state table for the original circuit and minimizing it? Justify your answer.