Problem 1: DAG Covering

Using the following library:

<table>
<thead>
<tr>
<th>Gate</th>
<th>Area</th>
<th>Delay</th>
<th>Symbol</th>
<th>Primitive DAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVERTER</td>
<td>2</td>
<td>1</td>
<td>![INVERTER Symbol]</td>
<td>![INVERTER DAG]</td>
</tr>
<tr>
<td>NAND2</td>
<td>4</td>
<td>2</td>
<td>![NAND2 Symbol]</td>
<td>![NAND2 DAG]</td>
</tr>
<tr>
<td>NAND3</td>
<td>6</td>
<td>3</td>
<td>![NAND3 Symbol]</td>
<td>![NAND3 DAG]</td>
</tr>
<tr>
<td>NAND4</td>
<td>8</td>
<td>4</td>
<td>![NAND4 Symbol]</td>
<td>![NAND4 DAG]</td>
</tr>
<tr>
<td>AO121</td>
<td>6</td>
<td>6</td>
<td>![AO121 Symbol]</td>
<td>![AO121 DAG]</td>
</tr>
</tbody>
</table>
Consider the following circuit:

- a) Show all matches at each node
- b) Show least area cost optimal match
- c) Show least delay cost optimal match
Problem 2: Kernel Extraction

Given the Boolean function:
\[ f = ab + acd' + aefg + aeh + abcj + abckm + abcnegp + abcneq' \]

a) Give all the kernels of the Boolean function \( f \) manually.

Problem 3: Retiming

Given the following circuit:

The numbers represent delay.

a) Draw the associated retiming graph.
b) Calculate W and D matrices.
c) Is there a retiming such that clock cycle is smaller or equal to 2? If yes, give the retiming, if no, then give the path that cannot be retimed to fit within the cycle.

Problem 4: ROBDD

a) Construct ROBDD for the signal order \( g,h,i,f \). BDD variable order is given as: \( a \rightarrow b \rightarrow c \rightarrow d \)
b) Run the BDD algorithm to create a ROBDD for each node \( g, h, \) and \( i \).
   b1) First show the unreduced BDD that results for each node, \( g, h, \) and \( i \).
   b2) Then reduce to create a ROBDD for each of those nodes.
Problem 5: Combinational Test

(1) Consider the following schematic diagram:

(a) List all of the stuck-at faults in the circuit (e.g. SA1: A1.1, A1.2, A1.out, O1.1, …)
(b) List all of the essential stuck-at faults in the network.
(c) Derive a test (cube) for each essential fault, and list them.
(d) Combine your test cubes into a minimum set of test vectors (minterms) for the circuit.

Problem 6: Finite-State Machines

Consider the following state transition graph:

(a) Does the machine have any equivalent states? Use the algorithm described in class to identify them. Show all working.
(b) Draw a state transition graph for the reduced machine.
(c) Using your reduced graph, perform an optimal state assignment. List all embedding constraints (following the Mustang model, as presented in class), but not the output constraints (just the fanin and fanout oriented constraints).
(d) Show the space (n-cube) for the state codes, the assignment, and how your constraints map onto the space. List any constraints not satisfied by your assignment.