Dominic Moylett - dm1905@my.bristol.ac.uk

Fault Tolerant Computing and VLSI Testing

Assignment 2

1. (a) $R_{BC}(t) = 1 - (1 - R(t))^2 = 2R(t) - R(t)^2$
   $R_{ABC}(t) = R_A(t)R_{BC}(t) = 2R(t)^2 - R(t)^3$
   $R_{ABCD}(t) = 1 - (1 - R_{ABC}(t))(1 - R_D(t))$
   $= 1 - (R(t)^3 - 2R(t)^2 + 1)(1 - R(t))$
   $= R(t)^4 - 3R(t)^3 + 2R(t)^2 + R(t)$
   $R_{system}(t) = R_{ABCD}(t)R_E(t) = R(t)^5 - 3R(t)^4 + 2R(t)^3 + R(t)^2$

   (b) If the life time of each of the five modules is exponentially distributed with parameter λ, then the reliability of each module is $R(t) = e^{-\lambda t}$:
   $R_{system}(t) = e^{-5\lambda t} - 3e^{-4\lambda t} + 2e^{-3\lambda t} + e^{-2\lambda t}$

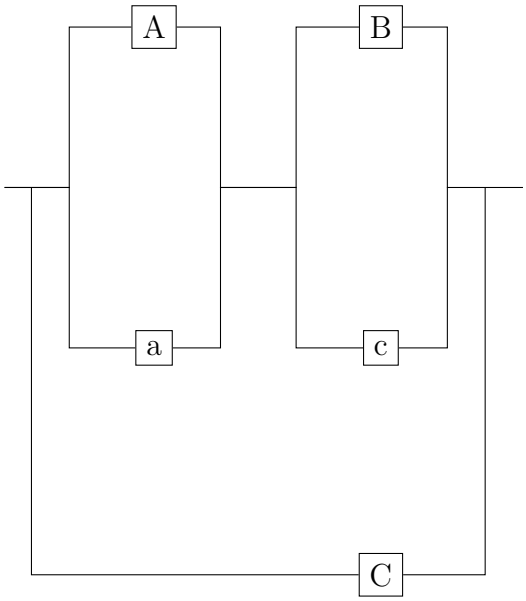
2. (a) NMR with spare means that the system will always act as an NMR system, but if any of the modules develops a fault that is detected, that module will be swapped out for a spare.

   In the case of our example will five modules, three of the modules will be in use and the other two will act as spares. If one of the three active modules develops a fault, this is swapped out for a spare and there will be three active modules again, but only one spare left. When a second module develops a fault, this is again swapped out for the remaining spare. We are now out of spares, so the system continues as a traditional NMR (in this case TMR).

   Self-purging redundancy has every module that hasn't yet developed a fault active at the same time. If a module develops a fault, that module is removed and the system continues with the remaining modules.

   In the case of our five module system, the system starts off in a 5MR setup. When a module develops a fault, this module is removed and the system switches to a setup with four voting modules. When another module develops a fault, that module is also removed, leaving three voters. The system now continues as a traditional TMR.

   (b) Because module b never fails, we do not need to consider it in the reliability diagram. The result is the following:



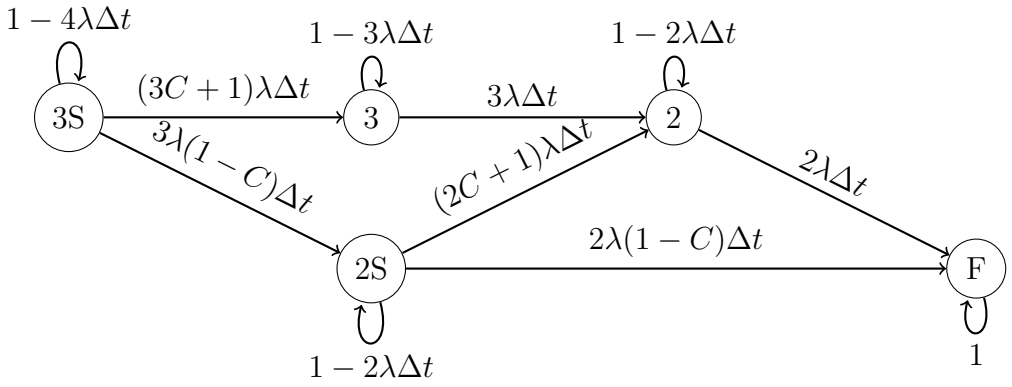
From this simplified reliability block diagram, we can derive an equation for the overall reliability as follows:

$R_{Aa} = R_{Bc} = 1 - (1 - R)^2$

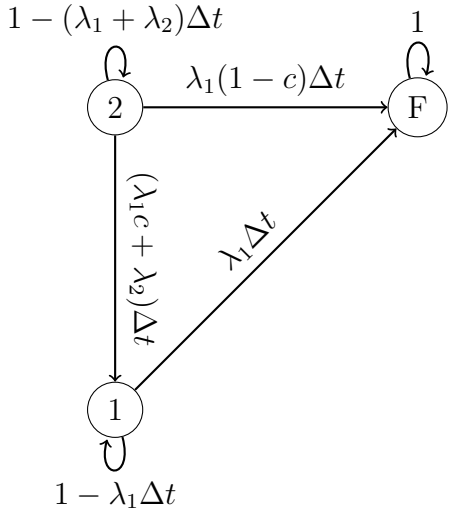
$R_{ABac} = R_{Aa} * R_{Bc} = (1 - (1 - R)^2)^2$

$R_{system} = 1 - (1 - R_{ABac})(1 - R_C) = 1 - (1 - (1 - (1 - R)^2)^2)(1 - R)$

3.
- 3S is all three modules working and the spare not in use.
- 3 is three modules working and the spare has either been put in for another module or has failed.
- 2 is two modules working and the spare has either been put in for another module or has failed.
- 2S is where one of the modules has failed, but the failure hasn't been detected.
- F is the failure state; either two modules have failed but neither were successfully detected and swapped for the spare, or three modules have failed and one was replaced with the spare.



4. **(1)**
   - 2 is where both modules are working.
   - 1 is where one module has discovered a fault: Either a fault has occured in the active module and it has been discovered and swapped successfully, or a fault has occured in the spare.
   - F is the failure state: Either a fault has occured in both modules or a fault has occured in the active module but it hasn't been detected yet.



(2) $P(t) = \begin{bmatrix} P_2(t) \\ P_1(t) \\ P_F(t) \end{bmatrix}, P_2(t) + P_1(t) + P_F(t) = 1, P(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$$A = \begin{bmatrix} 1-(\lambda_1+\lambda_2)\Delta t & 0 & 0 \\ (\lambda_1 c+\lambda_2)\Delta t & 1-\lambda_1\Delta t & 0 \\ \lambda_1(1-c)\Delta t & \lambda_1\Delta t & 1 \end{bmatrix}$$

$$P(t+\Delta t) = AP(t) = \begin{bmatrix} P_2(t) - P_2(t)(\lambda_1+\lambda_2)\Delta t \\ P_2(t)(\lambda_1 c + \lambda_2)\Delta t + P_1(t) - P_1(t)\lambda_1\Delta t \\ P_2(t)\lambda_1(1-c)\Delta t + P_1(t)\lambda_1\Delta t + P_F(t) \end{bmatrix}$$

$\frac{P_2(t+\Delta t)-P_2(t)}{\Delta t} = -P_2(t)(\lambda_1+\lambda_2)$

$\frac{P_1(t+\Delta t)-P_1(t)}{\Delta t} = P_2(t)(\lambda_1 c+\lambda_2) - P_1(t)\lambda_1$

$\frac{P_F(t+\Delta t)-P_F(t)}{\Delta t} = P_2(t)\lambda_1(1-c) + P_1(t)\lambda_1$

As $\Delta_t \to 0$, we get the differential equations for the Markov model:

$\frac{dP_2(t)}{dt} = -P_2(t)(\lambda_1+\lambda_2)$

$\frac{dP_1(t)}{dt} = P_2(t)(\lambda_1 c+\lambda_2) - P_1(t)\lambda_1$

$\frac{dP_F(t)}{dt} = P_2(t)\lambda_1(1-c) + P_1(t)\lambda_1$

We can integrate these using the Laplace Transform:

$sP_2(s) - P_2(0) = sP_2(s) - 1 = -P_2(s)(\lambda_1+\lambda_2)$

$P_2(s+\lambda_1+\lambda_2) = 1 \implies P_2(s) = \frac{1}{s+\lambda_1+\lambda_2} \implies P_2(t) = e^{-(\lambda_1+\lambda_2)t}$

$sP_1(s) - P_1(0) = sP_1(s) = P_2(s)(\lambda_1 c+\lambda_2) - P_1(s)\lambda_1$

$P_1(s)(s+\lambda_1) = P_2(s)(\lambda_1 c+\lambda_2) = \frac{\lambda_1 c+\lambda_2}{s+\lambda_1+\lambda_2}$

$P_1(s) = \frac{\lambda_1 c+\lambda_2}{(s+\lambda_1+\lambda_2)(s+\lambda_1)} = \frac{A}{s+\lambda_1} + \frac{B}{s+\lambda_1+\lambda_2}$

We can find the values of A and B via partial fractions:

$\lambda_1 c+\lambda_2 = A(s+\lambda_1+\lambda_2) + B(s+\lambda_1)$

If we set $s = -\lambda_1$ then:

$\lambda_1 c+\lambda_2 = A\lambda_2 \implies A = \frac{\lambda_1 c+\lambda_2}{\lambda_2}$

If we set $s = -\lambda_1-\lambda_2$ then:

$\lambda_1 c+\lambda_2 = -B\lambda_2 \implies B = -\frac{\lambda_1 c+\lambda_2}{\lambda_2}$

Substituting this back in, we find:

$P_1(s) = \frac{\lambda_1 c+\lambda_2}{\lambda_2(s+\lambda_1)} - \frac{\lambda_1 c+\lambda_2}{\lambda_2(s+\lambda_1+\lambda_2)} = \frac{\lambda_1 c+\lambda_2}{\lambda_2}(\frac{1}{s+\lambda_1} - \frac{1}{s+\lambda_1+\lambda_2})$

Finally, we find:

$P_1(t) = \frac{\lambda_1 c+\lambda_2}{\lambda_2}(e^{-\lambda_1 t} - e^{-(\lambda_1+\lambda_2)t})$

$R_{system}(t) = P_2(t) + P_1(t) = e^{-(\lambda_1+\lambda_2)t} + \frac{\lambda_1 c+\lambda_2}{\lambda_2}(e^{-\lambda_1 t} - e^{-(\lambda_1+\lambda_2)t})$

5. (a) Software Implemented Fault Tolerance (SIFT) is where we use software to make systems tolerant against hardware faults. Common examples of this include implementing the voter for an NMR system in software, or using software to implement system checkpoints.

   Software Fault Tolerance (SFT) is using fault tolerance techniques to defend against programming errors. Examples of this include acceptance tests and N-version programming.

(b) **(a)** The probability of a routine having no faults is $1-0.01 = 0.99$
Therefore the probrability of there being no faults in the whole program is $0.99^{20} = 0.8179$

(b) If the probability of the routine being faulty is reduced to 10% of the original level, then the new probability is:
$0.01 * 0.1 = 0.001$
Thus the probability of a program consisting of only this routine being free of faults is $1 - 0.001 = 0.999$

(c) The probability of version having no faults is $0.99^{20} = 0.8179$, as stated in part **(a)**.
The probability of a version being faulty is the opposite of this probability: $1 - 0.99^{20} = 0.1821$
The probability of the whole program being free of faults when three-version programming being used is therefore:
$(0.99^{20})^3 + 3 * (0.99^{20})^2 * (1 - 0.99^{20}) * 0.85$
$= 0.5472 + 0.3406 = 0.8578$