

## FIXING ESSENTIAL (AND OTHER HAZZARDS AND RACES

Consider a map of the state assignments for the asynchronous Digilock below

	00..	01..	11..	10..
..00	INIT	B0P2 : to 0000	ULK	ERR
..01	B0P1	B1R2 : to 1101	to 1001	to 1000
..11	B0R1 : to 1011	B1P2 : to 1111	to 1011	to 1000
..10	B1P1	B1R0 : to 1110	to 1010	to 1000

In order to eliminate critical races caused by the essential hazzards in this design, certain states, when a wrong input occurs, must be directed through unused states to the error state.

Take the case of state B1P2. It must be directed from 0111 to 1111 to 1011 to 1000. The last 2-bit transition does not result in a critical race - all 1xxx states lead to 1000, the error state. This requires modifications in the VHDL files like those below.

```

WHEN B1P2 = >
    UNLOCK< = '0';
    IF ( B0= '0' AND B1= '1' ) THEN
        next_sreg< = B1P2;
    END IF;
    IF ( B0= '1' ) THEN
        next_sreg< = "1111";--ERR;
    END IF;
    IF ( B0= '0' AND B1= '0' ) THEN
        next_sreg< = B1R2;
    END IF;

```

```

WHEN "1101" = >
    UNLOCK< = '0';
    next_sreg< = "1001";
WHEN "1111" = >
    UNLOCK< = '0';
    next_sreg< = "1011";
WHEN "1110" = >
    UNLOCK< = '0';
    next_sreg< = "1010";
WHEN OTHERS = > next_sreg< = ERR;

```

An example of the result of this is seen below.

