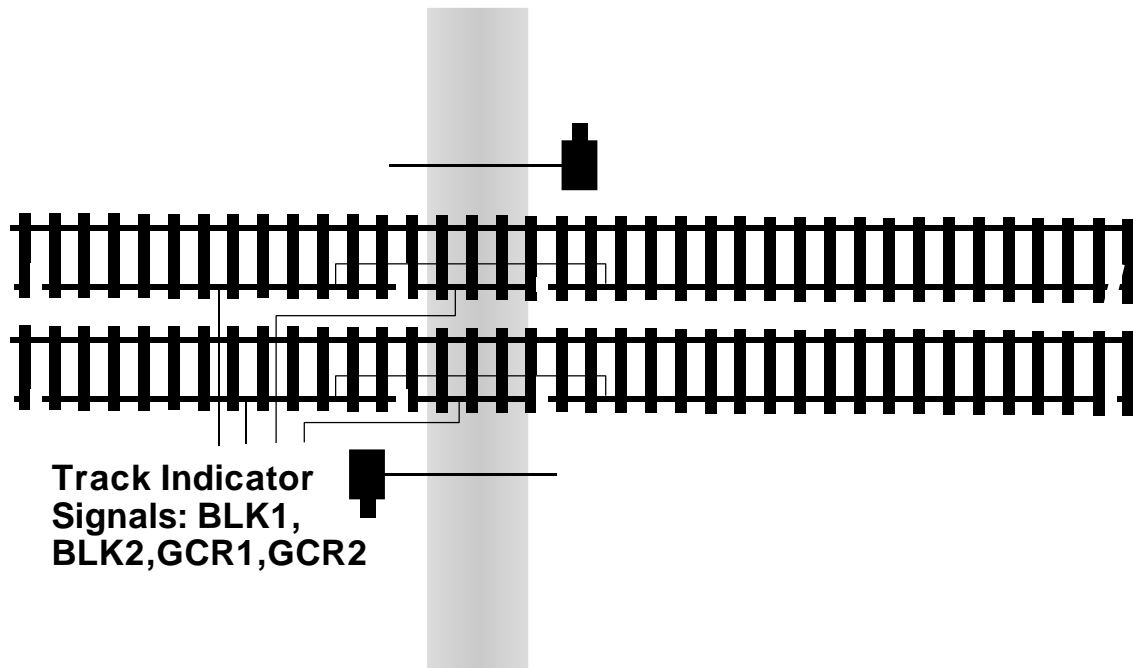


## DESIGN EXAMPLE

### THE DREADED RAILROAD GRADE CROSSING

The following problem was once given on a quiz. It was a disaster! It is not a hard problem - if you know how to approach it. The problem is shown in the diagram below.



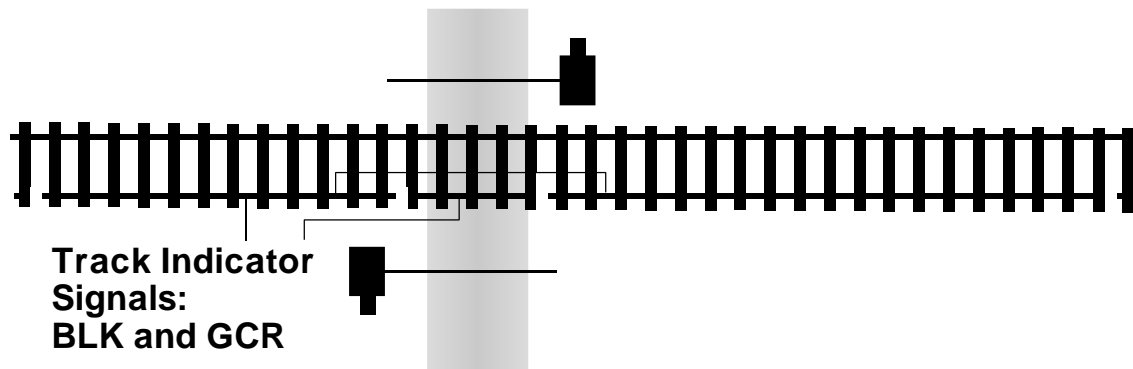
There are two tracks (one east-bound and one west-bound) and a road. They cross at a GRADE CROSSING. There are four indicator signals obtain directly from the tracks. The top rail on each track is grounded. The bottom rails have gaps in them at certain places: at BLOCK boundaries (anywhere from 100 yards to several miles in length) and at the boundaries of the grade crossing. The blocks are joined together on either side of the grade crossing. For our purposes it would be better if they were not joined but for railroad signaling purposes this must be. The BLK1 signal is grounded (asserted) whenever a train is on the WEST BOUND track (the upper one) and the BLK2 signal is grounded whenever a train is on the EAST BOUND track. GCR1 and GCR2 are grounded when a WEST BOUND or EAST BOUND train is in the grade crossing (or very near to it).

The crossing is controlled by BELLS and crossing ARMS. The bells and arms on either side of the tracks are connected together - obviously. The BELLS are to begin to ring as soon as (within 5 seconds) of the time a train first enters the block. Then, 5 seconds later, the arms are to come down. The additional five seconds is so you won't get the roof of your car smashed in by the arm! The bells are to stop ringing as soon as the train clears the interestion. The arms are also to rise (the exact timing of the arms with respect to the bells here is not

critical). We are to design a digital controller which will use these four indicator signals to control the bells and the arms.

Where should we start? We might design a machine with four inputs (and a clock) and two outputs to do the job. We ought to use a 5 second clock to meet the specifications above. We might do this but it would be the WRONG approach. Most students on the quiz tried to do it this way and never finished the problem! Instead, you ought to look for other solutions.

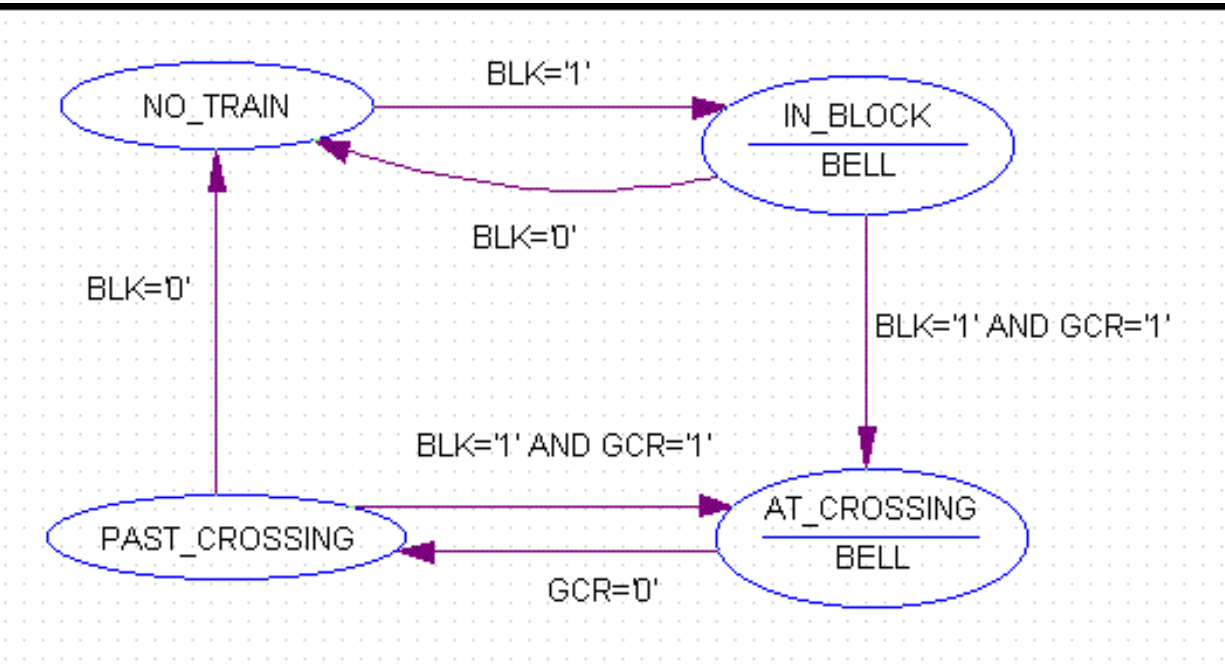
In this case, a good starting point would be to consider the following problem.



Here is a less complicated problem. It is not the problem we are to solve but solving it may give us an idea of how to do the more difficult problem. Here we have only one track to deal with. We ought to be able to make up a state diagram for this simple controller.

#### THE SINGLE TRACK CONTROLLER

The first step in the design of the single track controller is to devise the state diagram for it. Of course, we will use StateCAD for this. The diagram is below.



Before describing the entire diagram, there is an important StateCAD property to note. The **ELSE THEN** option was used. This option says that if none of the conditions are met on the transitions leading from a state, the machine remains in that state. In other words, it is not necessary to show arrows looping on one state.

The most logical place to begin is to assume there are no trains around and so neither the BLK nor the GCR detector is asserted. Note that it is highly unlikely that the GCR detector could be asserted without the BLK detector asserted - it would require a VERY short train!

The machine thus begins in the NO\_TRAIN state. It remains there until a train enters the block. Then it goes to the IN\_BLOCK state where the bell rings and the gate comes down. We have shown a transition going back to the NO\_TRAIN state. This is not likely to occur unless there is a side track in the preceding block and the train enters the block and then back out of it on to the siding. Better plan for this anyway.

The next logical event is for the train to enter the grade crossing. When this happens the controller advances to the AT\_CROSSING state where the bell continues to ring. Except for an unreasonably short train, the next event occurs when the train clears the crossing. Here the controller enters the PAST\_CROSSING state where the bell no longer rings.

Finally, the train leaves the block (BLK= 0) and the controller returns to the NO\_TRAIN state. There is, however, one additional transition shown. There is a transition from the PAST\_CROSSING state back to the AT\_CROSSING state. What happens if a train enters the crossing, backs up, and then enters the crossing again? We hope that doesn't happen but have done as much as we can do under the circumstances.

Before we go any further with this design, we need to compile it and then simulate it. (We could have simulated in StateCAD but I can't paste that into a document! The VHDL module is shown below (comments has been deleted from the file).

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY UNTITLED IS
    PORT (CLK, BLK, GCR: IN std_logic;
          BELL : OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF UNTITLED IS
    -- State variables for machine sreg
    SIGNAL AT_CROSSING, next_AT_CROSSING, IN_BLOCK, next_IN_BLOCK, NO_TRAIN,
    next_NO_TRAIN, PAST_CROSSING, next_PAST_CROSSING : std_logic;
BEGIN
    PROCESS (CLK, next_AT_CROSSING, next_IN_BLOCK, next_NO_TRAIN,
             next_PAST_CROSSING)
    BEGIN
        IF CLK='1' AND CLK'event THEN
            AT_CROSSING <= next_AT_CROSSING;
            IN_BLOCK <= next_IN_BLOCK;
            NO_TRAIN <= next_NO_TRAIN;
            PAST_CROSSING <= next_PAST_CROSSING;
        END IF;
    END PROCESS;

    PROCESS (AT_CROSSING, BLK, GCR, IN_BLOCK, NO_TRAIN, PAST_CROSSING)
    BEGIN
        IF (( GCR='1' AND (AT_CROSSING='1')) OR ( BLK='1' AND GCR='1' AND (
            IN_BLOCK='1')) OR ( BLK='1' AND GCR='1' AND (PAST_CROSSING='1')))) THEN
            next_AT_CROSSING<='1';
        ELSE next_AT_CROSSING<='0';
        END IF;

        IF (( GCR='0' AND BLK='1' AND (IN_BLOCK='1')) OR ( BLK='1' AND (
            NO_TRAIN='1'))))
            THEN next_IN_BLOCK<='1';
        ELSE next_IN_BLOCK<='0';
        END IF;

        IF (( BLK='0' AND (IN_BLOCK='1')) OR ( BLK='0' AND (NO_TRAIN='1')) OR (
            BLK='0' AND (PAST_CROSSING='1'))))
            THEN next_NO_TRAIN<='1';
        ELSE next_NO_TRAIN<='0';
        END IF;

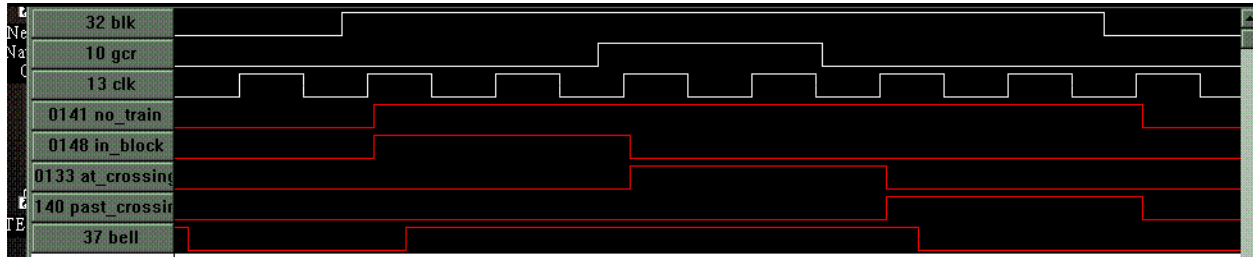
        IF (( GCR='0' AND (AT_CROSSING='1')) OR ( BLK='1' AND GCR='0' AND (
            PAST_CROSSING='1'))))
            THEN next_PAST_CROSSING<='1';
        ELSE next_PAST_CROSSING<='0';
        END IF;

        IF (( (AT_CROSSING='1')) OR ( (IN_BLOCK='1'))))
            THEN BELL<='1';
        ELSE BELL<='0';
        END IF;
    END PROCESS;
END BEHAVIOR;

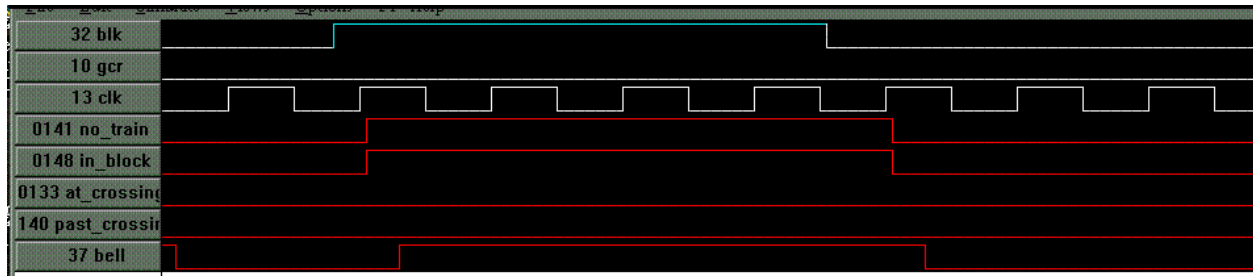
```

We will not show the compiled results here. They were compiled into a CPLD in order to generate a JEDEC file to use with the NOVA simulator.

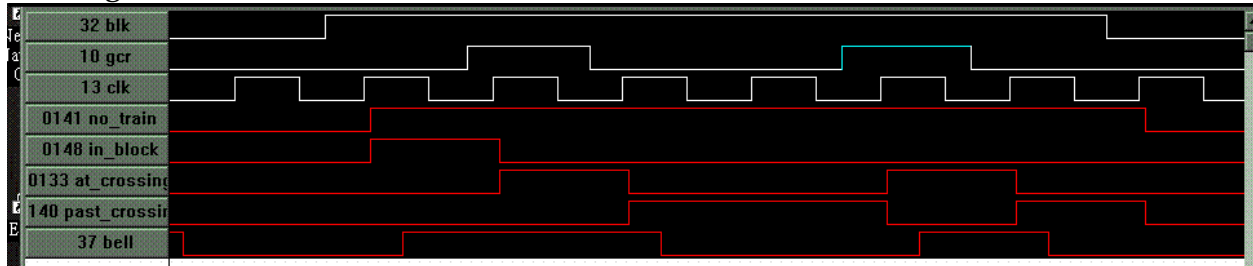
The first simulation is below.



In this simulation, the BLK signal is asserted for some time. The GCR signal is asserted for a short time in the middle. The machine functions normally with the Bell ringing when the train enters the block and stopping when the train leaves the crossing.



In this second simulation, the train enters the block and then, apparently, backs out of it. The Bell rings while it is in the block - as it should.



In this final simulation, the train enters the block, enters the crossing and then either passes through the crossing or backs out of the crossing. Then it enters the crossing a second time. Finally, it leaves in a normal manner. This is not a very desirable case since the bell stops ringing the first time it clears the crossing (OK) but does not start to ring again until the train actually enters the crossing for the second time. The least we can do is to get the NTSB to issue a decree that such actions must be done by engineers with the train moving VERY slowly!

We'll let it go at that.

Now, we have designed a controller for a grade crossing for a single track. Our problem has

two tracks. Where do we go from here? In some cases, the next step is to take what you learned in doing this design and attempt to expand it to the larger design. Before you do this, you really ought to give the problem some thought.

Suppose you had two single track crossings parallel with each other and only a few yards apart. We already know how to handle that. It wouldn't be practical to have two sets of bells and gates for tracks that are so close together. One set of bells and gates should be sufficient. They would operate when either track required it.

What have we concluded then? We don't have to do another controller. We only need to replicate the above controller and OR the two bell signals together. This ought to be easy (it wasn't on the quiz!).

The first step is to recompile the single track controller, this time placing it in a work library rather than a CPLD. Then we need to make a new VHDL module to combine these together.

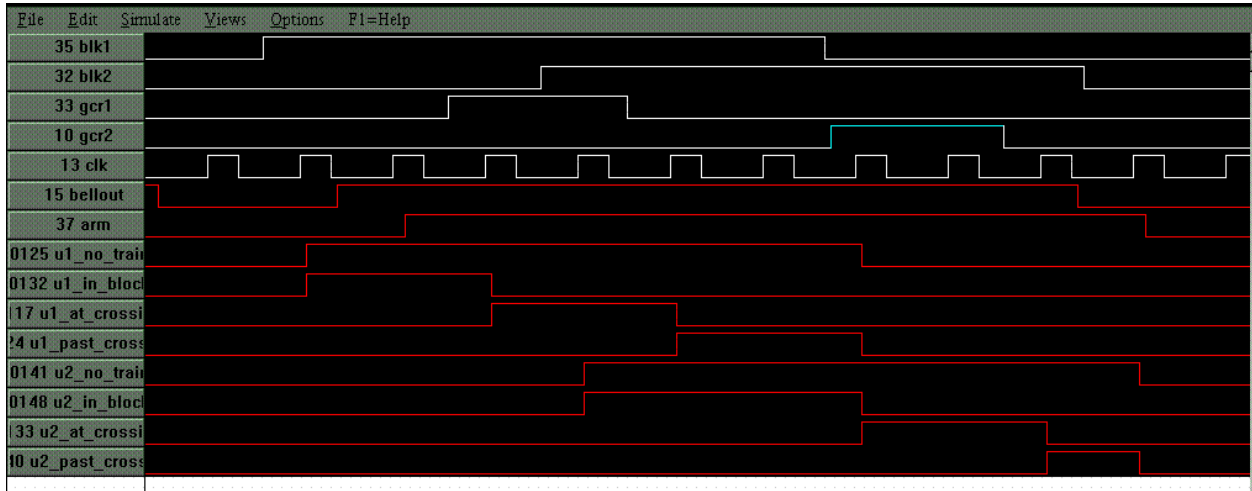
```
library ieee;
USE ieee.std_logic_1164.all;
USE work.all;
ENTITY two_track is
    PORT(
        blk1, blk2, gcr1, gcr2, clk: in std_logic;
        bellout, arm: out std_logic
    );
end two_track;
ARCHITECTURE main of two_track is
    signal bell1, bell2: std_logic;
begin
    u1: track_1 port map (clk, blk1, gcr1, bell1);
    u2: track_1 port map (clk, blk2, gcr2, bell2);
    bellout<=bell1 or bell2;
    process(clk)
    begin
        if (clk'event and clk='1') then
            if (bell1='1' or bell2='1') then arm<='1'; else arm<='0';
            end if;
        end if;
    end process;
end;
```

The above VHDL should be fairly readable by now. We will comment on the significant parts. Note that the WORK library is used; this is where the single track design is stored. We have added two internal signals, bell1 and bell2, which will be the outputs of the two single track controllers.

Next, u1 and u2 are the two controllers which are mapped to their respective signals. The bell signal, bellout, is obtained by ORing the two bell signals together.

Finally, the delay between the time the bells starts ringing and the time the arm comes down is accomplished by delaying the bell signal for one clock period (5 seconds).

Finally, we need to simulate the entire controller. In later notes we will talk about the correct way to test. We shall perform a simple test here.



Here we have a WEST BOUND train entering the block, passing through the crossing and, finally, leaving the block. As this train is in the crossing, and EAST BOUND train enters the block. At passes through the crossing a bit later and eventually leaves the block. Note that the BELL starts ringing as soon as the first train enters the intersection (within one 5 second clock period) and stops ringing when the last train clears the crossing - SUCCESS!