

SIMULATING COUPLED CLOCKS

John Haine

INTRODUCTION

This article explains how computer simulations of clocks can be carried out to solve problems such as their sensitivity to lifts or shifts, as a companion paper to my HJ article on *Shifting and Lifting a Two-Pendulum Clock* [1]. As this article is somewhat specialised it is published on the BHI website rather than in print.

Philip Woodward [2] has also previously used simulation methods to test the validity of Airey's equations, while Richard Stephen [3] has used simulation to investigate the performance of the pendulum/crutch system. In the past it would have been necessary to write one's own computer program to carry out such a simulation. Today however simulation methods are commonly used to design extremely complex engineering systems using off-the-shelf software packages such as "*VisSim*" [4] to tackle even very complex problems relatively straightforwardly using standard-specification personal computers. Simulation allows us to evaluate the performance of very complex engineering systems (much more complex than most clocks) to a far higher level of detail and accuracy than was possible with pencil and paper, and it has therefore become indispensable in modern design. Stephen's use of *VisSim* reported in [4] stimulated me to look at this package, and it has proved to be highly effective. An evaluation version of *VisSim* with a limited-period licence can be downloaded free from the supplier's website.

SOLVING DIFFERENTIAL EQUATIONS BY SIMULATION

Philip Woodward elegantly described the simulation approach to solving physical systems and its difference from the normal "mathematical" approach in Reference 2, and I couldn't improve on his explanation. As long as we know the differential equations for a system, we can calculate its behaviour moment by moment to great precision, and set up "probes" to measure various aspects of its behaviour that would be quite inaccessible if we tried to perform experiments. *VisSim* is particularly easy to use if you come from an electronics background like myself because the system is described by wiring together functional blocks representing various terms in the equations.

For simplicity I will first describe how a single pendulum is simulated, and then indicate how the simulation can be extended to the full double pendulum system. To simulate a pendulum we need an equation which links its deflection angle " θ ", to the various torques that act upon it. The usual equation for a single pendulum is:

$$ml^2\theta'' + aI\theta' + mgl\sin(\theta) + e(t) = 0$$

In this equation, m is the bob mass; l the pendulum length; a the damping factor; g the acceleration due to gravity; and $e(t)$ the excitation torque. The notation θ' and θ'' indicate respectively the first and second derivatives of the angle with respect to time, that is, the angular velocity and angular acceleration of the pendulum.

The problem with trying to simulate this equation directly is that, numerically, differentiation is a very "noisy" process. Often we have to take the difference

between very small values, and in a computer where the precision of representing numbers is limited this can result in errors which accumulate from step to step. On the other hand, numerical integration is a much more precise process, because as we are adding numbers this provides a “smoothing” function which reduces errors. So it is always a good idea to find a way of simulating a set of differential equations by using integration rather than differentiation. Fortunately for equations like the one above there is a simple “trick” which makes this very easy. We turn the equation round and write it:

$$\theta'' = -\frac{1}{ml^2}(al\theta' + mgl \sin(\theta) + e(t))$$

Now, looking at [Figure 1](#)¹, if we knew θ'' , then we could compute θ' from it by a simple integration; and in turn compute θ from that by a further integration. In the diagram, each box labelled “1/s” provides an output on the right which is the time-integral of the input on the left². So, given an input θ'' (representing the angular acceleration of the pendulum) on the extreme left, the pair of integrators provides first θ' , the angular velocity; then θ , the angle.

APPROACH TO SIMULATING THE COMPLETE CLOCK

To complete the basic clock simulation we need two further functions: a “sin” function that computes an output which is the trigonometric sine of the input; and something which models the escapement to generate the excitation $e(t)$. *VisSim* provides a sine block as standard, labelled “sin”. An idealised escapement would generate a short impulse of force on the bob acting in the direction in which it is moving each time it passes through its neutral position, which is the position where $\theta = 0$. *VisSim* provides a block that does almost precisely this, labelled “*crossDetect*”. This generates a unit impulse of length equal to the time step used in the simulation and direction corresponding to the slope of the input – we just have to scale this to the magnitude of the actual required impulse.

So the overall system diagram to simulate a single pendulum clock is as shown in [Figure 2](#). Given θ'' we compute θ' and θ by integration; then we can generate $\sin(\theta)$ and the impulse train when θ crosses zero using the *sin* and *crossDetect* functions. Once we have all these values, we can multiply them by appropriate constants and sum them to generate θ'' , which is where we started! To do this we have introduced *multiplier* (“x”) blocks and *sum* (“+”) blocks.

Now we have to set the initial conditions for the simulation. Just like a real clock, if we wind it up but leave it with the pendulum stationary and hanging straight down, nothing will happen! *VisSim* provides a facility to pre-set each integrator output to a starting value, or initial condition. In this case we could set the second integrator to a small positive value, corresponding to a small pendulum deflection, but the first to zero, corresponding to zero angular velocity. Then, when the simulation is started, the oscillation will build up steadily until the energy lost per swing equals the energy input from the impulses.

¹ When reading on-screen, underlined “[Figure x](#)” should hyperlink to the appropriate diagram.

² *VisSim* labels integrators “1/s” so I have used the same term – this is the Laplace Transform equivalent of time integration.

APPLYING LIFTS AND SHIFTS

The object of the simulation is to evaluate the affect of shifting or lifting the clock by applying horizontal or vertical accelerations. [Figure 3](#) shows how this can be done. First, I have simplified the working by assuming that l , the pendulum length, is one metre; and m , its mass, is 1 kg. This just means that there are two constants that we don't have to keep defining, and also the assumption that l is 1 metre means that for small displacements the horizontal acceleration and velocity of the bob equal its angular acceleration and velocity. This means that when we apply a horizontal acceleration to the clock we don't have to scale it by the pendulum length. So, in the block diagram, the horizontal acceleration S can just be added to the angular acceleration. Similarly, the vertical acceleration L can be added to the gravitational acceleration g . (As another simplification, I use a "g" value of 10 m/s/s rather than the actual value of ~9.81 m/s/s.)

How do we generate the shift and lift waveforms? I use a standard waveform where, at a certain time "T" into the run, the clock is accelerated at a constant rate "a" for a short period "t"; then decelerated at the same rate for the same period; so that it is then stationary having moved a small distance " at^2 ". [Figure 4](#) shows how vissim delay blocks can be wired together to achieve this. An input "a" defines the acceleration, which is applied to the input of the first block which delays it until after "T" seconds have passed. Then the value "a" is applied through the summer to the output; meanwhile it is delayed by "t" seconds before being added to the output with weight -2, which sets the output to $-a$. A further t seconds later, another "a" is added which drives the output to zero for the rest of time.

TIME MEASUREMENT

We are interested in finding the impact of lifts and shifts on the timekeeping and amplitudes of the pendulum. To measure the affect on timekeeping we need to look at how the times at which the zero-crossings of the pendulum waveforms change when its swing is perturbed. The *crossDetect* block lets us find out where the position crosses zero – now we have to find out the precise times. This is done with the arrangement of [Figure 5](#). We take the impulses from the *crossDetect* block, which are positive at each positive-going zero-crossing of the angular position θ and negative at each negative-going one. Then we use a "greater than" (" $>$ ") block, comparing the impulses with zero and selecting only the positive-going pulses (so we see just one zero-crossing per pendulum cycle). Now we have to find the exact position in time of these – to do this we use them to sample-and-hold a "ramp", a Vissim signal which increases linearly with time. We set this to have a gradient of unity so that any given time "t" the ramp value is also just "t" – if we sample-and-hold the ramp at this instant then the sampled value is just "t", the instant at which the zero-crossing occurred. We are really interested in what difference a lift or shift makes to the time relative to the clock had it not been disturbed. So to do this we create a complete duplicate of the whole system (just requiring a "copy and paste" operation on the computer, except that we do not apply any perturbing accelerations), and take the difference between the time kept by this "reference" clock and the time kept by the "experimental" one. Finally, we store the value of this error for every cycle in a data file which can be post-processed using a spreadsheet programme such as Excel. Also shown in [Figure 5](#), other measures such as the peak amplitude of a pendulum's swing can be recorded simultaneously.

COUPLED PENDULUMS

The equations for a pair of identical coupled pendulums are as follows.

$$ml^2\theta_1'' + al\theta_1' + bl(\theta_1' + \theta_2') + mgl\sin(\theta_1) + e_1(t) + kl(\theta_1 - \theta_2) = 0$$

and

$$ml^2\theta_2'' + al\theta_2' + bl(\theta_2' + \theta_1') + mgl\sin(\theta_2) + e_2(t) + kl(\theta_2 - \theta_1) = 0$$

These equations include two “coupling terms”. The most important ones are $kl(\theta_1 - \theta_2)$ and $kl(\theta_2 - \theta_1)$ which represent the extra torques exerted on each pendulum by the coupling spring which has spring factor k . The other term is $bl(\theta_1' + \theta_2')$ which is a damping torque proportional to the *even-mode* velocity of the pendulum system. These extra coupling terms can be included by adding cross-coupling paths between two separate but identical pendulum simulations, as shown in [Figure 6](#) (which for simplicity omits all but the essentials to show how this cross-coupling can be added).

AMPLITUDE MEASUREMENT

It will be apparent from my description of the results of simulating coupled pendulums in [1] that as well as measuring time-keeping we need to measure the amplitude of the “even mode oscillation” of a pair of pendulums, this being the in-phase mode which is excited by a sideways shift of the clock. The even mode amplitude is formed simply by adding the amplitudes θ_1 and θ_2 and dividing the result by 2. *VisSim* doesn’t provide a canned function for “rectification” so [Figure 7](#) shows how other blocks can be strung together to achieve this. The even-mode signal is essentially a sine wave, and we need to find its positive peak value. This is done by delaying the signal by a very small amount, and comparing the delayed signal with the undelayed using the comparator (“<”) block. As soon as the undelayed signal is slightly smaller than the undelayed, we know that we have just passed the peak value. The comparator output changes state from “0” to “1” at this instant, causing the “*crossDetect*” block to generate a short impulse, which in turn triggers the S&H block to sample-and-hold the peak value (which we carefully delayed anyway knowing that it would come in useful!). Thus the output of the S&H block changes to the most recent amplitude value after each peak has been reached, until the next one comes along. This amplitude value can also be fed into the data file shown in [Figure 5](#).

RUNNING SIMULATIONS

As well as stringing the necessary blocks together to represent the system to be simulated, a number of other aspects have to be taken care of to be successful.

The question of initial conditions has already been mentioned. If a clock is set up and set running, it will be found that at the start of the run its amplitude is probably either increasing or decreasing slowly, depending on how the initial conditions have been set. For example, if the initial conditions of the second integrators (representing the initial amplitudes) are set too low, then the system will take a long time (of the order of Q cycles) to reach dynamic equilibrium at its final amplitude of swing. Usually we want to apply test shifts or lifts only when it has reached equilibrium, so very long simulation runs have to be set up. The answer to this is to carry out an initial run just

to determine the steady-state amplitude, and then use this as the initial condition for subsequent runs. This approach is quite difficult for quadrature systems and I haven't found an alternative to simply running the simulation for 1000 seconds (simulation time) to settle down using approximate initial conditions, then applying the lift acceleration and looking at the next 1000 seconds.

VisSim provides choices for the time step used in simulation, and the underlying integration method. My standard approach is to use a time step of 0.1 milliseconds, which seems appropriate when assessing errors of the the order of 1 ms or less. I use what seems to be the most accurate integration option, "adaptive fifth-order Runge-Kutta". One could use an even smaller time step, the problem being simply one of the time taken to run, say, a 10,000 second (in simulation time) experiment at say 10 microseconds per step.

It will be seen that I use the data file construct to store the results, even though *VisSim* does provide facilities for plotting results. I do this because the file can be post-processed using the *Excel* spreadsheet programme to perform additional computation and also for its superb plotting facilities.

FINAL REMARKS

In this article I have given a very brief description of how a pendulum system and a clock can be simulated using the *VisSim* computer simulation package. Though I haven't gone into complete detail, I think that anyone interested with some computer skills could puzzle out how to assemble a clock simulation given a copy of the package itself (with its excellent help facilities). I am happy to provide electronic copies of my *VisSim* files as a starting point to anyone interested (but I am afraid that I could provide only very limited support).

There are some obvious extensions of this work which would be interesting and which I may tackle given time (or others may wish to try). First, I have assumed a very simple and idealised escapement – more complex escapements could be modelled by including comparator blocks with hysteresis and also offsetting their thresholds to examine the affects of more realistic mechanisms. So, for example, my simulation cannot reveal the "beating death" of two coupled clocks that was noted by Huygens, which depends on two out-of-phase pendulums beating together so that the amplitude of one falls below the level needed to unlock its escapement.

Given the current work on Project 150, a 3-pendulum clock would also be an interesting simulation project. A system with three pendulums has one "even" mode but three "odd" modes, and its motions would be of great interest. One assumes that the simulation will confirm that the long-term timekeeping is unaffected by a shift along any horizontal axis, except that due to the induced circular error as is the case for a two-pendulum clock.

Finally, most practical clock pendulums can actually swing in two planes, being provided with an additional pivot perpendicular to the main one to enable it to hang exactly vertically in the "for and aft" plane. This means that the pendulum has a second resonance close to its main period (though usually highly damped). The pendulum would therefore probably also be sensitive to shocks at right-angles to its normal plane of swing [5].

REFERENCES

1. John Haine; *Shifting and Lifting a Two – Pendulum Clock*; Horological Journal, July and August 2007.
2. Philip Woodward; *Experiments with a Simulated Clock*; Horological Journal, January 1991. Reprinted in *Woodward on Time*, pp 108 – 114.
3. Richard Stephen; *Modelling the Pendulum*; Horological Journal, August 2004.
4. See www.vissim.com; in the UK *VisSim* is available from www.adeptscience.co.uk.
5. Robert J Matthys; *Accurate Clock Pendulums*; OUP 2004; pp106 – 108; and private correspondence with the author.

FIGURES

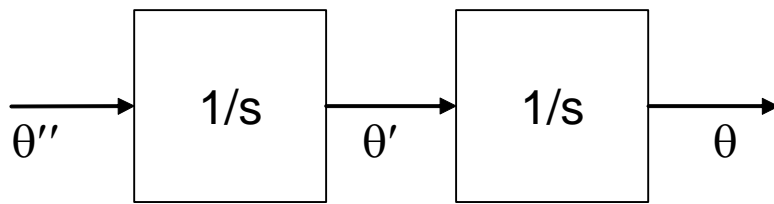


Figure 1: Integrating θ'' twice to get θ' and θ

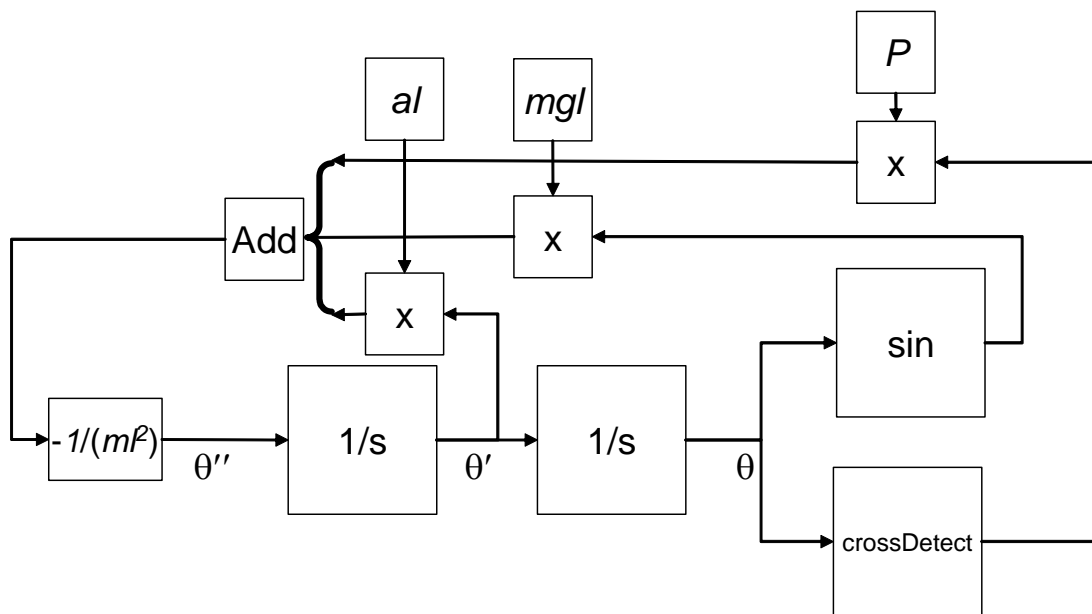


Figure 2: Complete simulation block diagram for simple clock

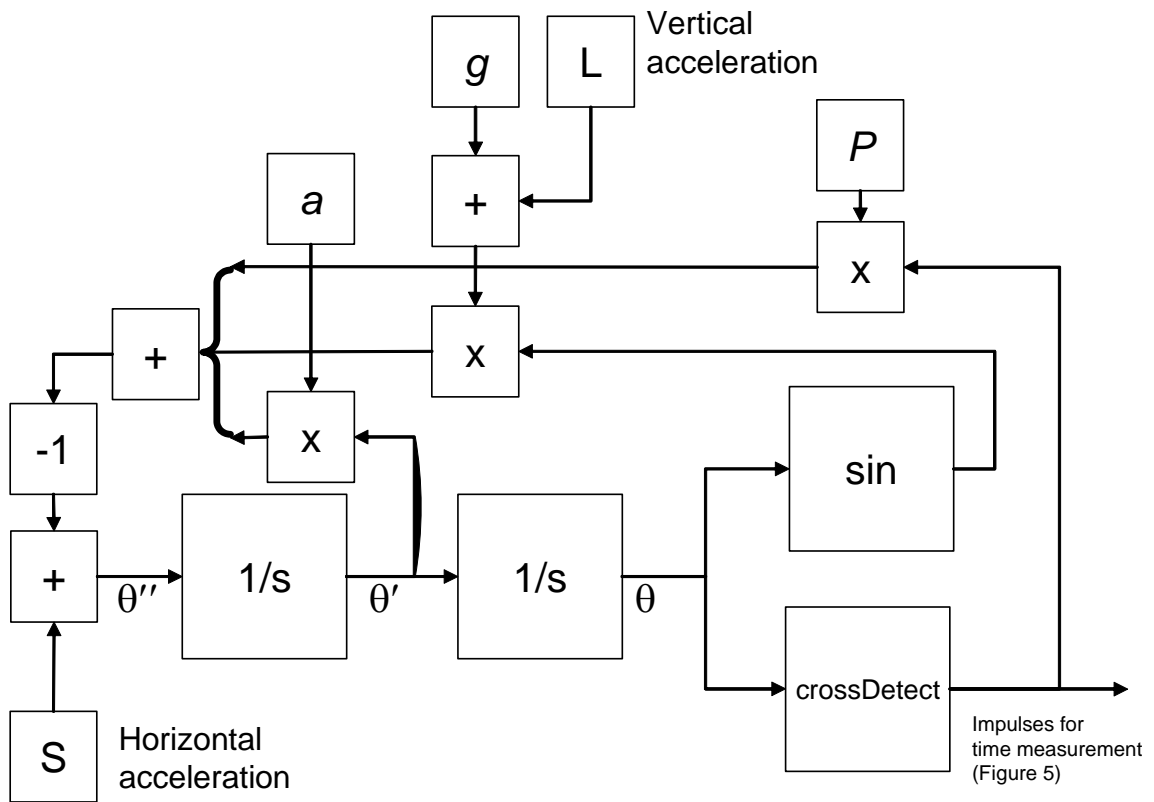


Figure 3: Adding shifts and lifts

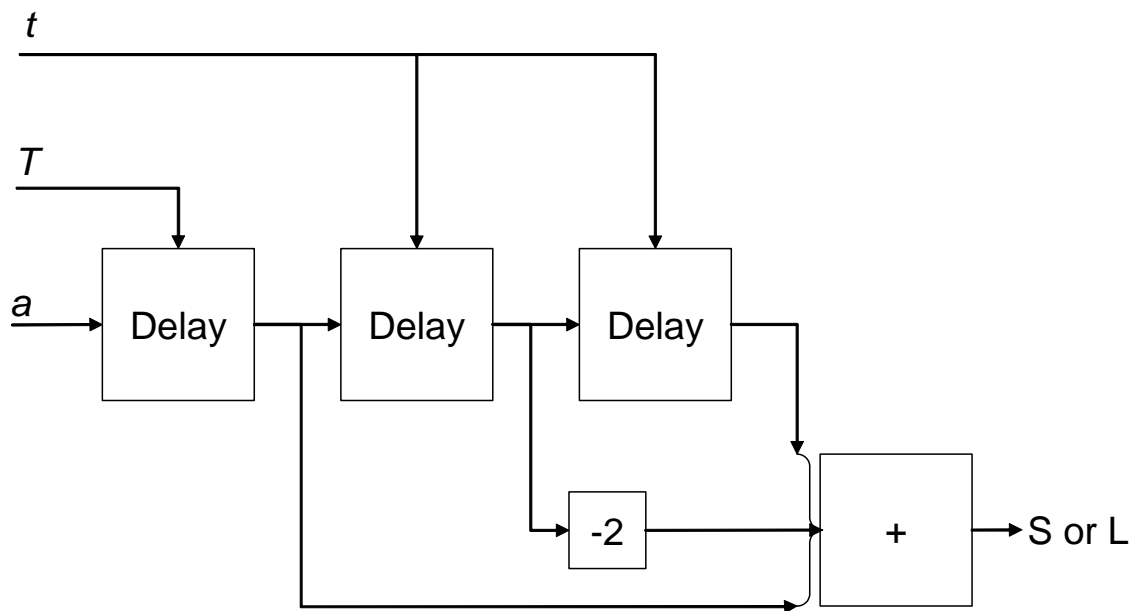


Figure 4: Generating shift and lift waveforms

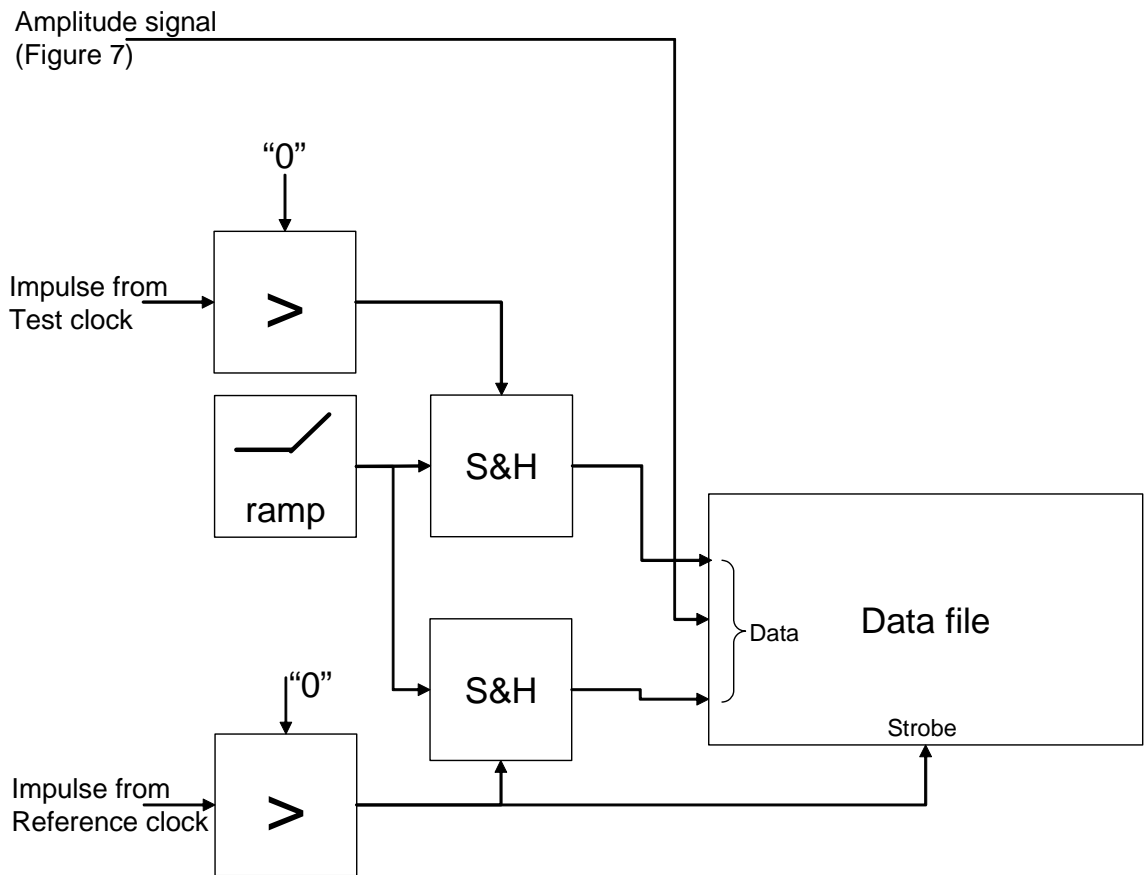


Figure 5: Time measurement and data storage

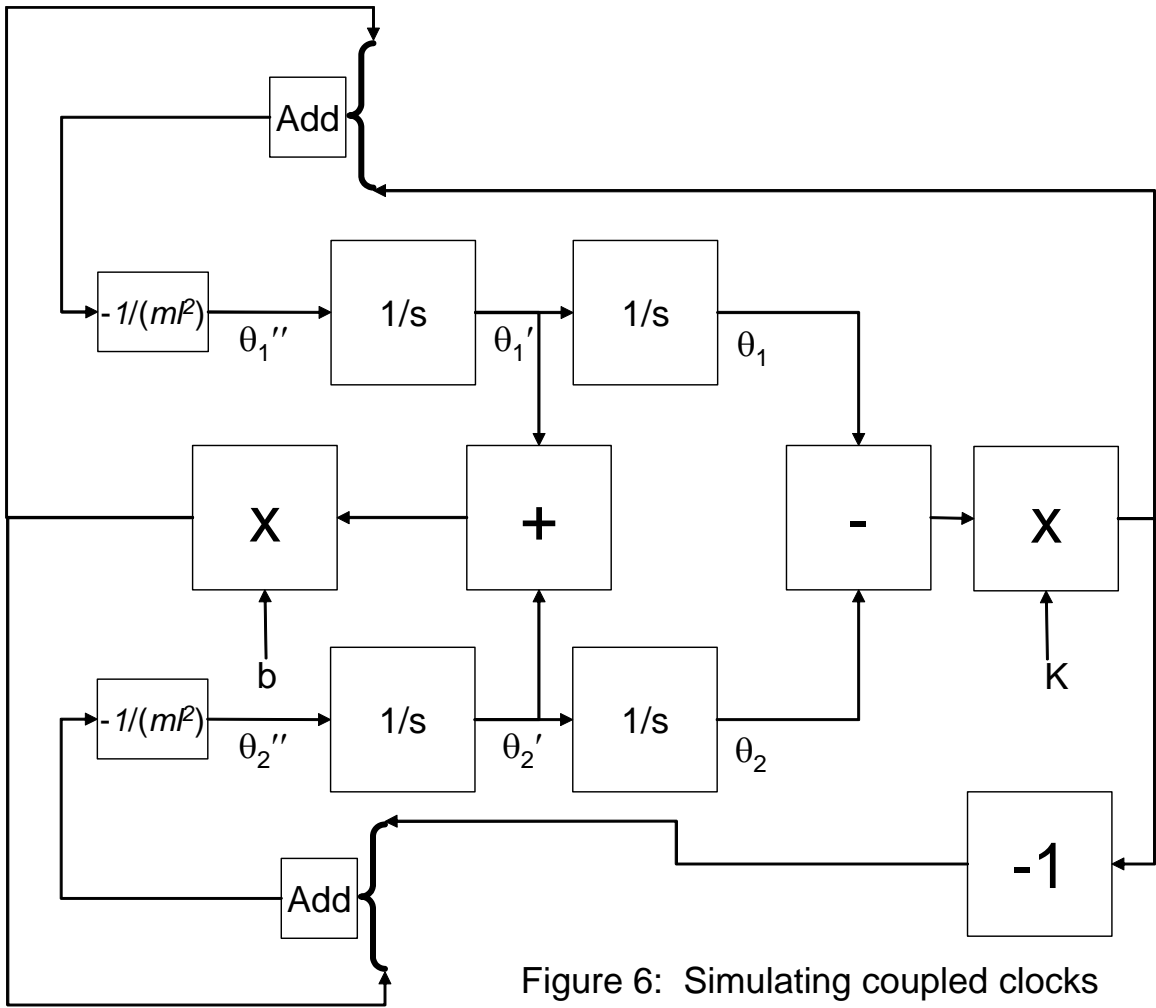


Figure 6: Simulating coupled clocks

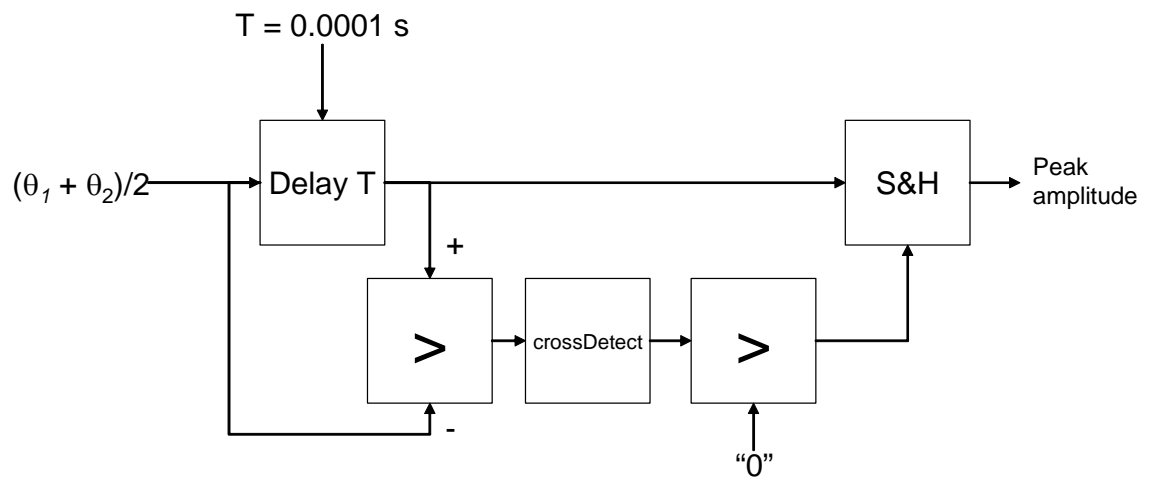


Figure 7: Measuring peak amplitude