7
Mechatronics

7.1 Modelling of Mechatronic Systems

The aim of this chapter is to apply the foundations obtained in previous chapters to actual mechatronic systems. The interaction between the domains is particularly significant here because the interfaces contribute significantly to system behaviour. In particular, we are aiming too low if we only consider electronics or mechanics independently of each other. The problem of the joint simulation of electronics and mechanics must be solved, which again throws up a whole range of problems:

In the case of mechatronics, the time constants of mechanics and electronics often differ by orders of magnitude. For macromechanics we can expect oscillations of a few (tens of) hertz. In electronics the figure lies four to six orders of magnitude higher. So we could assume that the dynamic interaction between electronics and mechanics can be disregarded. The opposite is true. For example, a wide range of control algorithms are performed on embedded controllers. Their running time again lies in the millisecond range, so that dynamic feedback between electronics and mechanics very definitely plays a role. This requires the dynamic simulation of the entire system in order to be able to track cyclical dependencies, including those that cross domain boundaries. Another reason for the importance of this is the fact that domain boundaries often also represent the interfaces between design teams working in parallel.

For the field of mechanics, precise models that are compatible with an electronics simulator must be prepared. During the following chapter we will exclusively consider multibody mechanics, which is generally sufficient for system considerations in the field of macro mechatronics. Even with this limitation the vectorial nature of mechanics is not easy to represent on a circuit simulator.

An efficient conversion is of crucial importance for the field of software in particular. Millions of machine instructions are performed in a single second of real time. On the other hand, it is necessary to precisely determine the timing of the functions implemented using software, which requires a precise synchronisation between software and electronics. This is indispensable in order to correctly reflect the dynamics between software, electronics and mechanics.

Mechatronic Systems Georg Pelz
 ISBN: 0-470-84979-7

In addition to the provision of the models and the dynamic simulation of a system that goes beyond domain limits, the representation of the results can sometimes be a problem. Of course, we always obtain the values of system variables plotted against time, as is also normal for electronics simulation. In the case of mechanics, however, we would often prefer an animation, in order to be able to evaluate the system behaviour at a glance. As far as software is concerned, the typical outputs of an electronics simulator are virtually useless. We would like a debugger, like those used in pure software development, which illustrates the sequence of the software and furthermore permits control of the sequence, perhaps by breakpoints.

As will be shown in what follows, the introduction of hardware description languages into mechatronics lags behind that of microsystem technologies. A significant reason for this is that microsystem technologies developed from microelectronics, so hardware description languages, which were initially developed for microelectronics, were quickly implemented there too. By contrast, mechatronics developed from mechanical engineering, where electronics is often reduced to control technology and can thus be considered using comparatively simple equations. Recently, hardware description languages have also begun to be encountered in mechatronics, with the automotive industry taking the lead.

The use of hardware description languages for the design of mechatronic systems will be illustrated on the basis of four examples. These are a semi-active wheel suspension system, an internal combustion engine with drive train, a camera winder and a disk drive.

7.2 Demonstrator 1: Semi-Active Wheel Suspension

7.2.1 System description

In what follows a semi-active wheel suspension will be described, see Hennecke *et al*. [137] or [138], Duttlinger and Filsinger [89] or Roppenecker [352] for the technical fundamentals. The idea of semi-active wheel suspension is that the system adapts the parameters of shock absorbers or body springs to the current road conditions and the corresponding driving situation. This is achieved by an embedded processor, which means that electronics, mechanics and software have to be considered at the same time here. The system described in what follows reflects the concept of BMW's 'electronic damper control', see [137] and [138] and Figure 7.1. A similar system is also offered by Mercedes-Benz, see [89]. The difference between semi-active and active wheel suspension is that, in the latter case, forces can be applied by hydraulics, for example, in order to improve driving safety and comfort. Put simply, the vehicle lifts one or more wheels up in order to minimise the vertical movement of the body. This approach has, however, not yet become prevalent for reasons of cost and energy.

The motivation for the use of semi-active wheel suspension is that driving safety and comfort represent competing goals in the context of suspension systems. Driving safety is jeopardised because unevenness in the road triggers vibrations in the wheel or body. The former lie in the range of around ten hertz, the latter in the range of one hertz. In both cases it should be ensured that these resonances are sufficiently damped. Otherwise, strong variations in the wheel load will arise. In particular, the wheel load will fall significantly as a result of the upward movement in question. In the extreme case individual wheels will lift. The frictional connection between the wheels and the road, and consequently the cornering force of the wheels in question, then falls to zero, which can have severe consequences in a bend. These problems could be countered by basically setting the damping level high. However, this would have a negative effect upon the ride comfort, which would transmit every unevenness in the road directly to the body and thus to the passengers. This is particularly true for the vibrations in the range of four–eight hertz, which would be perceived as unpleasant or at least uncomfortable by the passengers.

Whilst an electronic control of the wheel suspension was not possible the primary issue was to find a reasonable compromise between safety and comfort. Semi-active wheel suspension is based upon the principle that the damping of road conditions can be switched over appropriately during the journey. Problematic driving situations requiring a high level of damping are recognised firstly by the consideration of the vertical acceleration of the body, from which the triggering of vibrations by the road can be deduced. In addition, driving manoeuvres that also require increased stability are recognised, for example, sharp braking, fast driving through bends or quick accelerator pedal movements in automatics. The resulting pitching and rolling movements of the vehicle should be limited by higher damping. After the identification of the road condition and the driving state, the next step is to determine the correct damping level and to set this at the shock absorber.

The implementation is based upon a digital controller that processes embedded software, see Figure 7.1. This carries out the actual control algorithm and takes on a whole range of additional functions such as, for example, the plausibility testing of the sensor values to be processed, the safety concept, or the provision of data to other components of the vehicle. In addition, there are electronics for the signal processing, such as D/A and A/D converters, which provide the connection between the digital and analogue worlds. The actual conversion between the physical quantities is taken care of by the acceleration sensor and the adjustable shock absorbers. Finally, the mechanics of the suspension also has to be taken into account.

The central component of the system is the regulateable shock absorber, which will be considered in more detail in what follows. Fundamentally, shock absorbers use the Stokes' friction of a viscous liquid, e.g. hydraulic oil, in order to convert motion energy into heat. The oil is squeezed through a narrow valve in accordance with the movement. If we control the route of the oil in accordance with the direction of flow, then different valve diameters and thus different damper constants can be provided for compression and tension mode. Furthermore, several damper

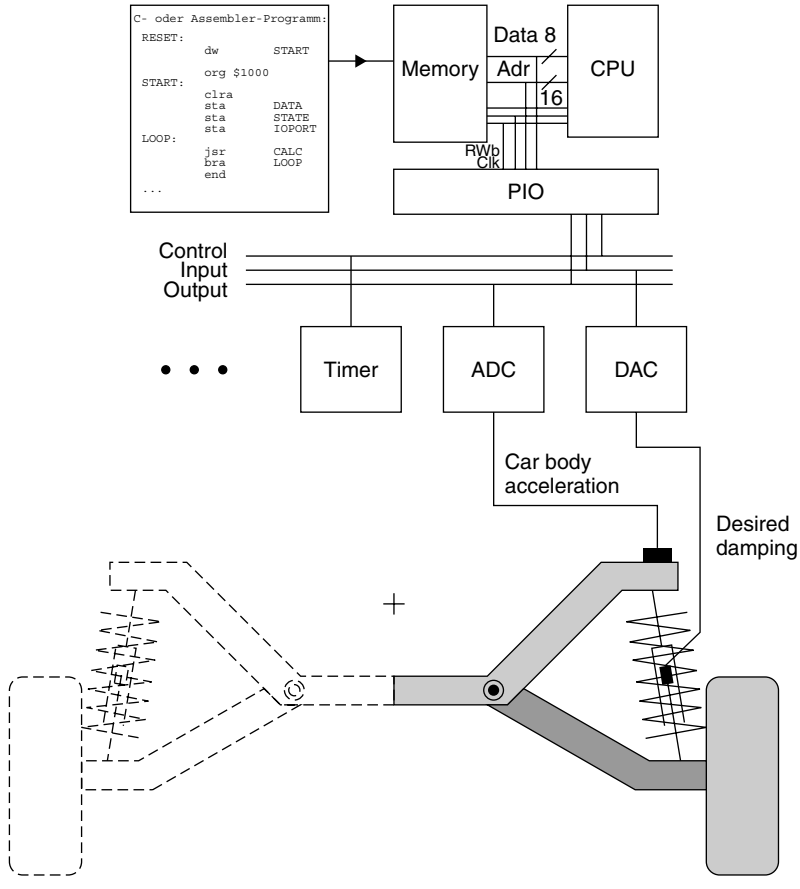

Figure 7.1 Semi-active suspension

characteristic lines can be planned for the adjustable shock absorber. By using a rotary disk valve the route of the hydraulic oil is determined such that the oil passes through various valves. The power consumption is thus limited because only the rotary disk valve has to be operated. Typical delay times for the adjustment of the characteristic lie in the range of around 20 ms.

7.2.2 Modelling of software

The software in this demonstrator runs on an embedded processor, which is compatible with the Motorola 68HC05. It was developed in C and then initially compiled into Assembler and from there compiled into machine code. The basis of the mechatronic HW/SW co-simulation used here is software modelling, which has already been described in detail in Chapter 5 and [326], [328]. For the sake of simplicity, only the dependency between the body acceleration and the damper characteristic

will be implemented here. More complex algorithms and further input data can be added in a straightforward manner.

7.2.3 Modelling of mechanics

The approach to the modelling of the mechanical components depends to a large degree upon the desired application of the models. Thus, we can provide complex multibody models for the development of the mechanics which can be used, for example, to determine the tilt of the wheels or changes to their toe-in in relation to the spring deflection of the car body, see for example Schmidt and Wolz [365]. However, these details are of less importance to the development of the electronics. In this context the overall behaviour of the system is much more interesting, and simpler models are sufficient for the consideration of this.

The model of the wheel suspension is based upon the fundamental model presented in Chapter 6. However, in this context some further boundary conditions have to be taken into account. Firstly, we should note the fact that real shock absorbers exhibit different characteristics for the compression and tension modes. Typically the damping is significantly higher in tension mode. This is because the dampers should transmit road unevenness to the passengers as little as possible. This means that comparatively low forces should arise in the compression mode. Consequently most of the consumption of the motion energy occurs in the tension mode. This situation generates a first nonlinearity. A second naturally arises as a result of the switching of the damper characteristics, which is precisely the main purpose of the system.

It has already been demonstrated in Chapter 6 how such a model is put together from the basic models for masses, springs, dampers, etc. for a quarter of a car. For this reason, a system-oriented approach to modelling will be followed here. The Lagrange equations shall form the basis for this. As shown in Chapter 6 these take the following form:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathrm{T}}{\partial \dot{\mathrm{q}}_\mathrm{i}}\right) - \frac{\partial \mathrm{T}}{\partial \mathrm{q}_\mathrm{i}} = \mathrm{Q}_\mathrm{i} \qquad (\mathrm{i} = 1, 2, \ldots, \mathrm{n}) \tag{7.1}$$

The degrees of freedom are again the y-positions. The kinetic energy of the system T is found from the kinetic energy of the two masses:

$$\mathrm{q}_1 = \mathrm{y}_\mathrm{a},\ \mathrm{q}_2 = \mathrm{y}_\mathrm{b}, \qquad \mathrm{T} = \tfrac{1}{2}\mathrm{m}_\mathrm{a}\dot{\mathrm{y}}_\mathrm{a}^2 + \tfrac{1}{2}\mathrm{m}_\mathrm{b}\dot{\mathrm{y}}_\mathrm{b}^2 \tag{7.2}$$

The generalised forces are the spring, damper and weight forces:

$$\begin{aligned}
\mathrm{Q}_{\mathrm{a}1} &= \mathrm{k}_\mathrm{w}(\mathrm{y}_\mathrm{s} - \mathrm{y}_\mathrm{a} + \mathrm{l}_{0\mathrm{a}}),\\
\mathrm{Q}_{\mathrm{a}2} &= -\mathrm{k}_\mathrm{s}(\mathrm{y}_\mathrm{a} - \mathrm{y}_\mathrm{b} + \mathrm{l}_{0\mathrm{b}}),\\
\mathrm{Q}_{\mathrm{a}3} &= -\mathrm{b}(\dot{\mathrm{y}}_\mathrm{a} - \dot{\mathrm{y}}_\mathrm{b}),
\end{aligned}$$

$$
\begin{aligned}
Q_{a4} &= -m_a g, \\
Q_{b1} &= k_s(y_a - y_b + l_{0b}), \\
Q_{b2} &= b(\dot{y}_a - \dot{y}_b), \\
Q_{b3} &= -m_b g
\end{aligned} \tag{7.3}
$$

where k_w and k_s denote the constants of wheel and body springs, Q_{ai} and Q_{bi} the components of the generalised forces Q_i on the bodies A and B, and b the coefficient of damping. Furthermore, the convention is used for the springs that a positive force increases the positions in question. The constants l_{0a} and l_{0b} correspond with the y-positions of the two related bodies in a relaxed state. At the damper, positive forces bring about an increase in the coordinate difference. Thus the damping force tends to resist a positive relative velocity. The weights Q_{a4} and Q_{b3} finally effect a reduction in the positions and are thus counted negatively. Substitution into the Lagrange formula gives the following equation system:

$$
\begin{aligned}
m_a\ddot{y}_a &= k_r(y_s - y_a + l_{0r}) - k_f(y_a - y_b + l_{0f}) - b(\dot{y}_a - \dot{y}_b) - m_a g \\
m_b\ddot{y}_b &= k_f(y_a - y_b + l_{0f}) + b(\dot{y}_a - \dot{y}_b) - m_b g
\end{aligned} \tag{7.4}
$$

This can be directly formulated in an analogue hardware description language, whereby the damper constant b can be set to the value in question using an if-then-else construct.

7.2.4 Simulation

The test case for which the simulation should be performed is, as in Chapter 6, driving over a step of 5 cm height. As described in Chapter 6, the model of a quarter of a car can be used again at this point. The first result of the simulation is shown in Figure 7.2 and represents the relative movement of wheel and vehicle body after driving over the bump. Initially the wheel starts to move and compresses the body spring. This movement produces an overshoot. Overall, we recognise that the natural frequency of the wheel actually lies in the range of around ten hertz. The oscillation, however, decays very rapidly as a result of the damping, which is increased at the change-over point. The car body follows the vertical movement at a significantly slower rate than the wheel. The forces in question are applied by the compressed body spring. Here too the natural oscillation is, as expected, set to a value of around one hertz. The change-over of the damping from the 'soft' to the 'hard' characteristic is triggered by the digital signal shown at the top and takes place around 50 ms after the step is reached.

The difference in driving behaviour resulting from the change-over is particularly well illustrated by the consideration of the body spring compression, which is equated with the compression of the shock absorber. This is shown in Figure 7.3 for the same case of driving over a step of 5 cm height. In the first 50 ms there

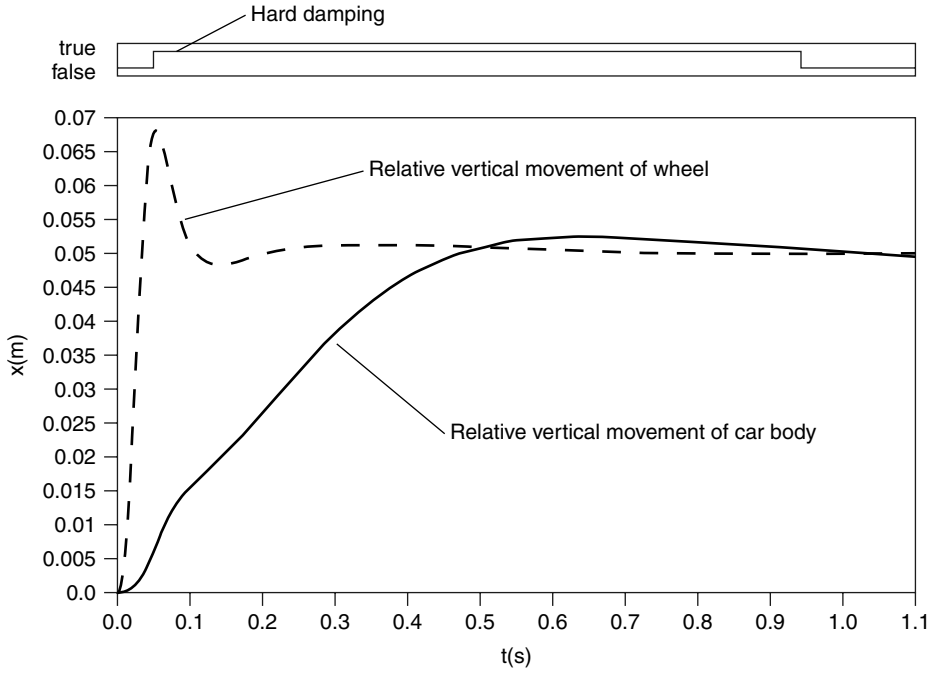

Figure 7.2 Relative vertical movement of wheel and body

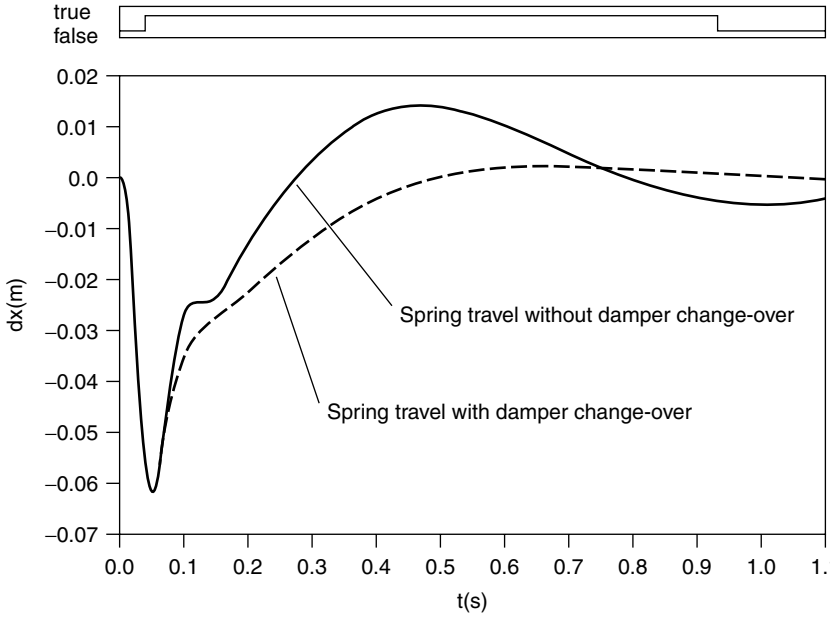

Figure 7.3 Spring compression with and without the changeover of the damper characteristic

is still no difference, because the change-over to hard damping has not yet taken place. Then we see clear differences. If the change-over does not take place, then the spring compression goes positive, which means that the car rises off the springs and thus the wheel loading falls. This leads to the safety problems discussed above. When the damper characteristic is changed over, the spring compression exhibits a significantly more desirable behaviour.

As discussed above, it is also necessary to concern ourselves with the visualisation of the mechanics and the software. For the mechanics this quickly becomes an ad hoc solution, unless CAD data is available from the development of the mechanics, which can be drawn upon for a suitable representation. The solution shown here shows a graphic representation of a quarter of a car, which is 'driven' by the simulation data, see Figure 7.4. This functions both as direct output during simulation and also for a subsequent consideration of the extracted data.

By contrast, a general solution can be found for the software. As already indicated it is a question of considering the software using a type of debugger and controlling its processing, see Figure 5.7. In contrast to the tools normally used, this is a debugger that considers the processing of software on virtual hardware. Naturally, if this is to run properly an exact synchronisation between hardware and software is crucial. This is guaranteed by the approach to mechatronic hardware/software cosimulation described in Chapter 5. This synchronisation ensures that when a break-point is reached, the simulation of mechanics and electronics

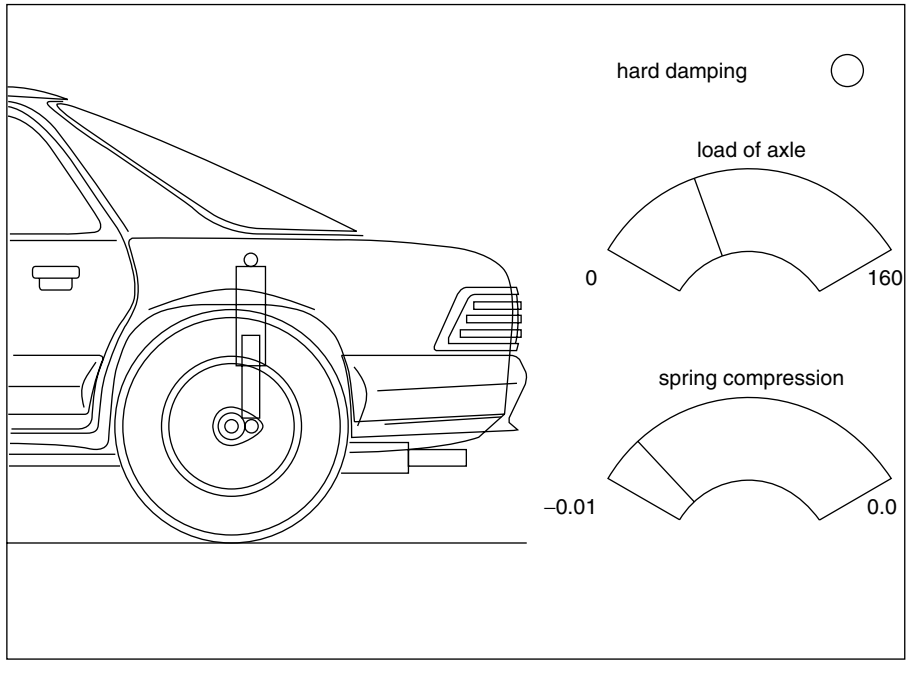

Figure 7.4 Visualisation of the mechanics

do not proceed further. Such a debugger has been implemented and used. It can represent the underlying software both on assembler and high-level language level. The user interface of the debugger is shown in Figure 5.7.

The models were formulated in the analogue hardware description language MAST and simulated using the Saber electronics simulator. On a SUN Sparc 20 workstation the simulation requires 22 CPU seconds. This includes the simulated processing of 3.8 million machine instructions of the embedded processor.

7.3 Demonstrator 2: Internal Combustion Engine with Drive Train

7.3.1 System description

This example relates to the modelling of the propulsion of a motor vehicle see also [332]. If all components that are relevant in this context are to be considered, then the modelling would extend from the accelerator pedal, via the actual engine and the drive train, to the road, which has a certain gradient. If we impose narrower limits on the model, then we can at best investigate specific parts of the system. In what follows, the overall system will be investigated, whereby the necessary foundations can be found for example in Roduner and Geering [349], Hockel [146], Tiller *et al*. [400] or in the publication by Bosch [39], [40]. As in the previous example, generic models are used where possible, which can be adapted to the application case in question by suitable parameterisation. Again, the model is based upon the assembly of the overall model from basic components.

The system under investigation is shown in Figure 7.5. The partitioning of the system for the modelling mainly follows the function blocks shown in the illustration. At the top level, the system model consists of an electronic circuit diagram. The components of this have been modelled using an analogue hardware description language, and so the whole of the overall system can be simulated without further problems. Particular mention should be made of the fact that two types of modelling have been used here.

The drive train primarily relates to the rotational and translational movements, whereby here a line represents the duality of torque / angular velocity or force / velocity. Kirchhoff's laws in particular apply here, i.e. the forces and torques at a node and the velocities and angular velocities in a closed loop add up to zero. The lines in question are printed in bold.

All other lines are directional. They form a network that is comparable to the block circuit diagrams of control technology. In particular, Kirchhoff's laws do not apply here. The main task of this partial network is to supply the torque generation block and the automatic gearbox with the necessary data. This includes the control of the combustion engine.

In the system description we start with the accelerator pedal, which in our case represents the driver's primary input. Its angle a_{pe} sets the kickdown mode in train

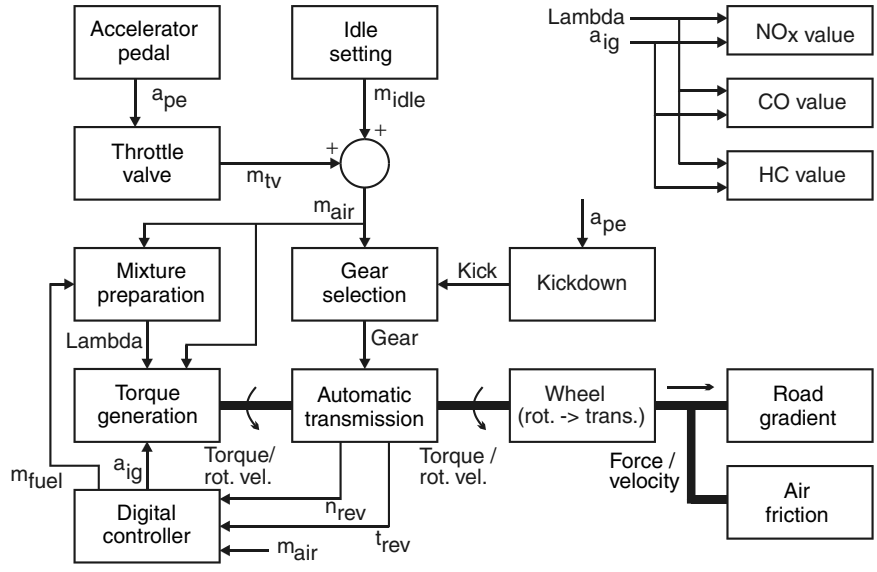

Figure 7.5 System structure of the internal combustion engine with drive train

when the driver stamps upon the accelerator pedal, which may cause the automatic gearbox to change down so that a higher acceleration can be achieved. Furthermore, a_{pe} determines the position of the throttle valve, which again provides a suitable quantity of air m_{tv} for combustion. In addition, there is a further proportion of air m_{idle} which corresponds with the idle setting and is permanently available. The sum of the two air masses m_{air} is firstly a measure for the desired acceleration and is therefore fed into the gear selection and the digital computer. Secondly, m_{air} is naturally available to the mix preparation, which calculates the fuel mixture ratio λ from the air and fuel mass m_{fuel}. The digital micro-controller determines the fuel quantity and also the ignition advance angle a_{ig}. Its inputs are the engine speed n_{cs}, a trigger signal of the crankshaft t_{cs} and the air mass m_{air}. The regulator can be described as a software model, see Chapter 5, or as a simple behavioural model.

The actual engine is represented by the torque generation block, which converts the fuel mixture ratio λ, the air mass m_{air} and the ignition advance angle a_{ig} into the mechanical duality of torque and angular velocity. The first step in the modelling process is to estimate the energy of the fuel mass and to multiply this by the associated efficiency. Furthermore, this raw torque is further reduced if the engine is not run in the optimal range of λ, m_{air} and a_{ig}. Then the rotary motion is transmitted to the gearbox and there suitably converted, which leads to a different defined relationship of torque and angular velocity. The model of the wheel converts the rotary motion into a translational motion, which is characterised by the duality of force and velocity. Counter-forces come into play here, which represent the gradient of the road and the drag due to air resistance. Finally, the

exhaust blocks supply a broad overview of the current NO_x, CO and HC values of the engine, which are obtained from λ and a_{ig}.

7.3.2 Modelling

At this point some of the above-mentioned blocks will be described. We begin with the generation of torque by an internal combustion engine. The underlying process represents the conversion of chemical into mechanical energy. The input quantities are the air mass m_{air}, the fuel mixture ratio λ, the ignition advance angle a_{ig}, and the engine speed at the crankshaft n_{cs}. We could start by deriving the resulting torque M from the average pressure in the cylinder p_m and the piston-swept volume V_D, see [39]:

$$M = \frac{V_D \cdot p_m}{2\pi} \tag{7.5}$$

However, the average pressure in the cylinder cannot easily be determined from the input quantities. Rather, p_m depends upon the combustion process, which takes place in three spatial dimensions. Also problematic is the irregular shape of the combustion space, which furthermore continuously changes shape. Finally, turbulence effects in the mixture of air and fuel often cannot be disregarded. Thus the physical modelling is separated from the consideration of the system as a whole. Since in this case there is no good foundation for a structural modelling, we now turn our attention to experimental modelling. Figure 7.6 shows a typical characteristic of torque against engine speed for a predetermined air mass, see [39]. Such a characteristic can be described by an inverted parabola, whereby its parameters are to be adjusted for each operating case of the engine under consideration. In this context a range of values for different air masses should be recorded.

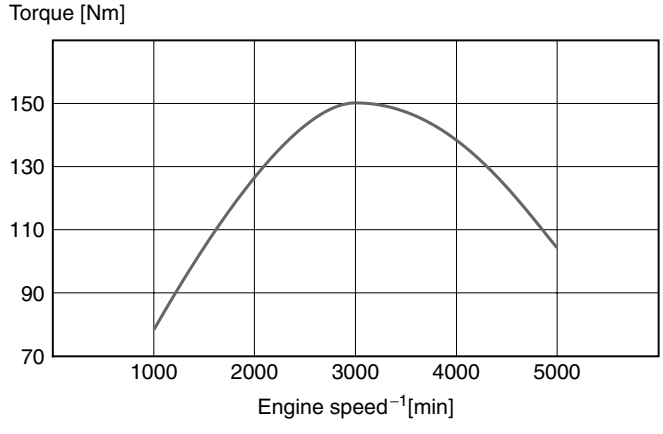

Figure 7.6 Typical characteristic of torque against engine speed for a given air mass

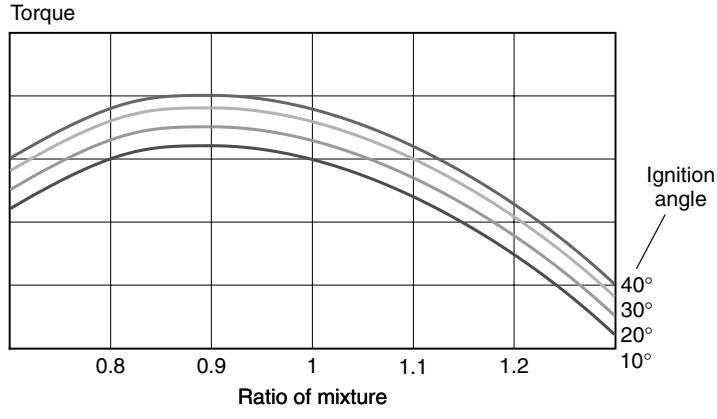

Figure 7.7 Typical dependency of the torque upon the fuel mixture ratio and the ignition advance angle

In this manner a 'raw torque' figure is obtained that can be adjusted by multiplying it by a weighting factor to take account of the other input quantities λ and a_{ig}. We thus do not have to store the characteristics for all sensible combinations of m_{air}, n_{cs}, λ, and a_{ig}. The factor for the calculation of the actual torque is found from Figure 7.7, which shows a typical dependency of the torque upon the fuel mixture ratio and the ignition advance angle, see [39].

The generated torque is opposed by a corresponding counter-moment, which is found from the inertial force of the vehicle mass and the forces due to air friction and road gradient[1]. The rolling friction is disregarded in this context. The inertial force F_i can be replaced by:

$$F_i = -m\dot{v} \tag{7.6}$$

where m represents the vehicle mass and v the vehicle velocity. For the drag F_l:

$$F_l = -\tfrac{1}{2}c_W A \rho v^2 \tag{7.7}$$

where c_W is the drag coefficient of the car body, A its frontal area and ρ the density of the air. The force from upward and downward gradients, F_g, is formulated:

$$F_g = -mg\sin(\alpha) \tag{7.8}$$

where g is gravity and α the angle of the road gradient. A single hardware description is used for each of these three force components. This is connected such that the counter-forces are summed. The wheel model converts the counter-force into a counter-moment which again is linked to the generated moment via the gearbox.

[1] Rise or fall.

7.3.3 Simulation

The test case that underlies the simulation is a virtual ride of 35 seconds real time, see Figure 7.8. From top to bottom the following values are represented: the operation of the accelerator pedal as a percentage of the maximum angle; the automatically selected gear; the engine speed and the road speed. Regarding the road profile, the vehicle comes upon a gradient of 10% after 10 seconds, which returns to a level section after 25 seconds.

At the beginning of the simulation, the driver operates the accelerator pedal at 20%, so that the vehicle slowly accelerates away from stationary. Upward gear changes are made until the vehicle is in third gear. The engine speed lies at just under 2000 rpm. The road speed rises to around 50 km/h. After ten seconds the vehicle reaches the incline, which after a short time results in a slight decrease in road speed. The driver reacts to this by pushing the accelerator pedal down to 90%. The gearbox, which in the meantime has changed up into fourth gear, reacts and changes through third gear into second gear. The engine speed rises accordingly and the vehicle accelerates up the hill. With the higher speed the gearbox can change back up into third gear after around 18 seconds, which again reduces the engine speed slightly. After 21 seconds the driver lifts off the accelerator a little, so that the speed falls from approximately 80 km/h to 70 km/h. At 25 seconds the incline is ended, so that the vehicle easily accelerates at 60% throttle in fourth gear and at 35 seconds a road speed of around 90 km/h is achieved.

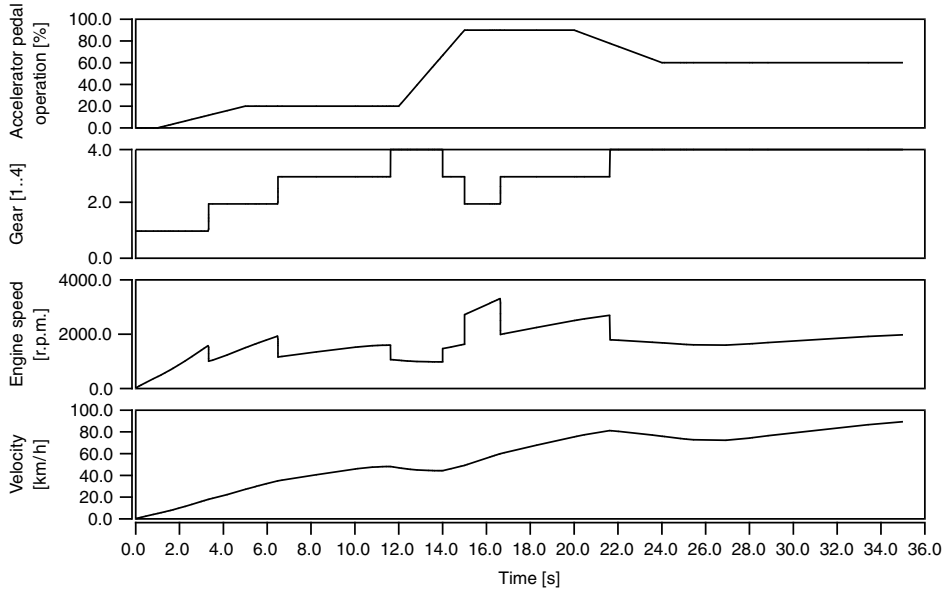

Figure 7.8 A virtual ride by car. The curves represent from top to bottom: accelerator pedal as a percentage of full acceleration; automatically selected gear; engine speed and road speed

7.4 Demonstrator 3: Camera Winder

7.4.1 Introduction

The third example originates not from automotive engineering. It relates to a photographic camera with an external motor winder, which looks after the automatic film movement. This system differs from the two other mechatronic examples because, in this case, the electronics provides the drive and not only the control of the system. Furthermore, at the time of the investigation not all components of the system were available. Nevertheless, the system could still be investigated on this basis, and this investigation yielded a significant foundation for the design decisions under consideration. Thus the described approach is capable of supporting the top-down design of mechatronic systems. The subject is also dealt with in Pelz *et al*. [331] and Landt [213].

The model was set up for this system to illustrate a real system, the Leica R8, together with its motor winder. At the time of modelling the winder was still at the development stage. The main task was to maximise the number of pictures that could be taken by a battery or rechargeable battery. Several motors were investigated to this end; one of the candidates was not even in existence at the time. All that was available for this motor was a set of parameters that had been calculated and estimated in advance. One of the main questions was whether it would be worthwhile to actually have this virtual motor developed. The gearbox had to be set up for each motor in order to minimise the power consumption whilst achieving the specified time for taking a picture.

7.4.2 System description

The system includes camera and motor winder and has an electronic and a mechanical circuit. The electronics consists of a battery pack, the power transistors, the associated control circuit and the electric motor, which is naturally also part of the mechanics. This subsystem also comprises of a gearbox, a mechanical load and a mechanical stop. The mechanical load represents the effect of friction when the film is pulled out of its cartridge.

Figure 7.9 shows the (simplified) system structure, placing emphasis on the mechanics and the interface between electronics and mechanics. Again, the simulation is performed using an electronics simulator, which we can expect to correctly process the electronics.

7.4.3 Modelling

Batteries and rechargeable batteries

The model of the battery or rechargeable battery pack is based upon measurements of various brands and types. First a load cycle is defined that imitates the typical

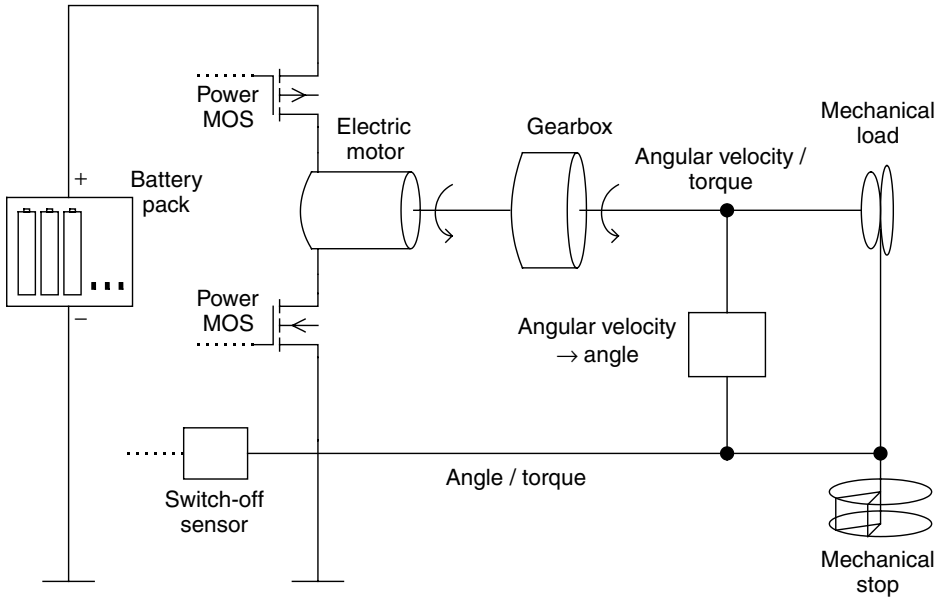

Figure 7.9 System structure of the camera winder

use of the camera. Then a large number of cycles are performed under defined temperature conditions, e.g. $+20^{\circ}$C and -20°C. The voltage and internal resistance are measured regularly and related to the charge consumed. This information parameterises the model of the battery/rechargeable battery pack, which consists primarily of a voltage source with internal resistance. This can be used to evaluate the variations in the number of pictures that are produced by the temperature and the brand and type of the batteries and rechargeable batteries.

Electric motor

Only direct current motors with permanent magnets will be considered in this context. The modelling, see Hardware description 7.1, is initially based upon equating the sum of the internal voltages with the externally applied voltage. This corresponds with a series connection of armature resistance, armature inductance and generator voltage. This yields the following voltage balance:

$$v_{in} = R_a i_a + L_a i_a + b\omega \tag{7.9}$$

where v_{in} is the input voltage, R_a the resistance of the armature winding, L_a the inductance of the armature winding, i_a the armature current, b the generator voltage constant (back-emf) and ω the angular velocity of the shaft. The resulting torque is found from the generated torque minus the moment of inertia of the armature and the frictional moment.

```
template my_dcpm A1 A2 WRM WRME= kt, b, la, ra, j, dft
 electrical    A1, A2
 rotational_vel WRM, WRME
 number         kt  = 1.0,                    # Torque constant
                b   = 1.0,          # Generator voltage constant
                la  = 1e-6,                # Armature inductance
                ra  = 1.0,                 # Armature resistance
                j   = 1.0,                   # Moment of inertia
                dft = 0.0           # Frictional losses, dynamic
{
 branch iin = i(A1->A2),                  # Input current, motor
        vin = v(A1,A2)                 # Terminal voltage, motor
 branch at = tq_Nm(WRM->WRME),                  # Torque at shaft
        av = w_radps(WRM,WRME)         # Angular velocity at the
                                            shaft
 val   tq_Nm frict                           # Frictional moment
 var   nu    av_t, iin_t              # Derivative of av and iin
                                          # with respect to time
    values {
     if (av > 0) {          # Frictional moment acts against the
      frict = dft;                          # rotation direction
     }
     else {
       frict = -dft;
       }
     }
equations {
  iin_t = d_by_dt(iin)           # Calculation of the derivative
                                          # with respect to time
  av_t = d_by_dt(av)             # Calculation of the derivative
                                          # with respect to time
  vin =
    ra*iin + la*iin_t + b*av                 # Armature circuit
    at = kt*iin - j*av_t - frict # Calculation of output torque
 }
}
```

Hardware description 7.1 MAST model of the direct current motor

Gearbox

Like the electric motor, the gearbox represents a basic model that can be used very diversely. Its function is to set the relationship between rotational velocity and torque between the two mechanical terminals according to the transmission ratio. The following equations apply:

$$\omega_A = \alpha\omega_B$$
$$M_B = \eta\alpha M_A \tag{7.10}$$

where the angular velocities ω_A and ω_B are in a fixed ratio of α, the moments M_A and M_B are in a fixed ratio of $1/\alpha$ and furthermore the efficiency η is taken into account. Thus there is nothing further standing in the way of a direct implementation of the model in VHDL-AMS or another hardware description language.

Mechanical load

A further problem in the modelling is the mechanical load, which arises from the film being pulled out of the cartridge. Here too — as in the case of batteries and rechargeable batteries — it makes little sense to model the underlying physical relationships. Firstly, these processes are complex, such as for example the transition from static to sliding friction. Secondly, the necessary material properties cannot easily be determined. In addition, a physical model would involve significant calculation time, so that it would barely be suitable to investigate the system as a whole. Here too experimental modelling is used, with measurements of the load moment being incorporated into a model. Such a table model breaks down the winding into 18 increments each of 10°, for example. A complete winding movement thus covers 180°. Every section is assigned a load moment that opposes the motor moment. This yields an incremental or at best a piece-wise linear path of the load moment characteristic, see Figure 7.10. The characteristic thus has irregularities or at least kinks. Both make the numerical calculation of the underlying differential equation system considerably more difficult. To remedy this, the characteristic was approximated in the form of a Chebyshev polynomial, although spline or Bézier interpolations would also have been feasible.

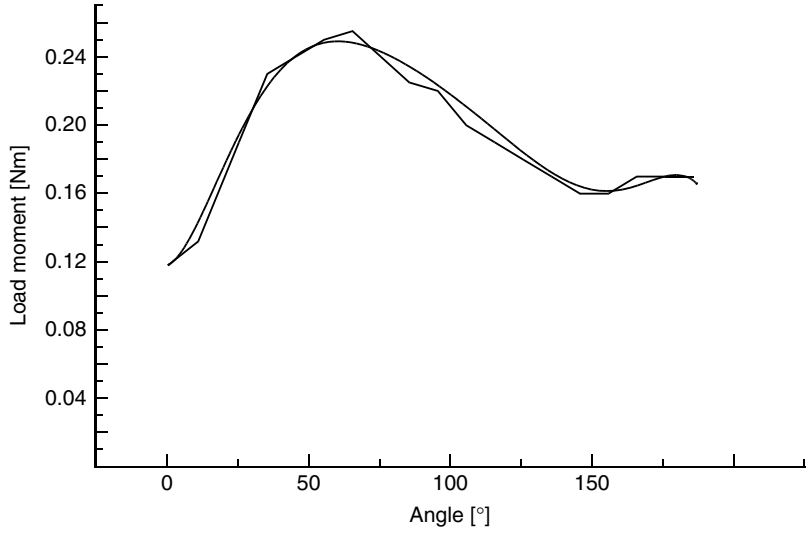

Figure 7.10 Measured and approximated load torque during film movement

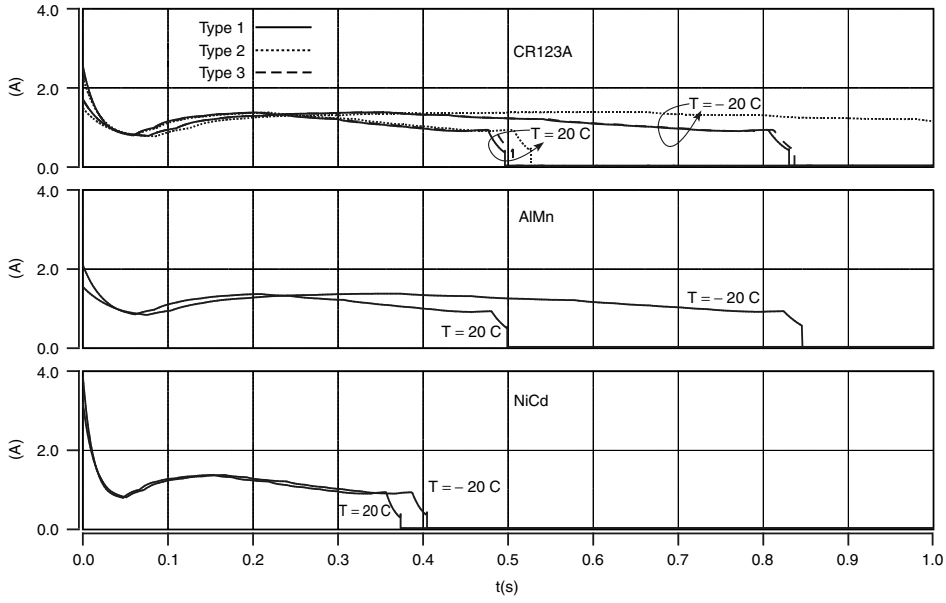

Figure 7.11 Armature currents for various battery/rechargable battery types and temperatures

7.4.4 Simulation

For the simulation shown in the following, see Figure 7.11, a motor was selected and operated using extremely varied batteries and accumulators at $+20^{\circ}$C and -20°C. All the curves show the armature current, the integral of which is a measure for the charge consumption. When the armature current is zero, the winding operation is complete. Thus we can also use these diagrams to check adherence to the time specification for the winding operation. Individually, lithium batteries (CR123A), AlMn batteries and NiCd rechargeable batteries were used. It is striking that the NiCd accumulators are considerably less problematic, which is due to their overall lower internal resistance. Furthermore, we see that in some cases the lithium batteries are no longer capable of adhering to the specification of 850 ms for a winding operation at low temperatures.

7.5 Demonstrator 4: Disk Drive

7.5.1 Introduction

The section gives an overview of electronics and firmware development using The Virtual Disk Drive, which is a generic model of a disk drive, formulated in hardware description language, see also [78], [320], [321]. The Virtual Disk Drive covers digital, analogue and power electronics, firmware and mechanics. Thus it

even allows us to assess functionality spread over all these domains. One example of this is track following and track seeking for the read/write-head.

Moreover, this section details the resulting electronics design methodology. For instance, it will be shown how key system properties, e.g. seek time, can be determined by means of the mixed simulation of mechanics, electronics and firmware. In addition, the same simulation environment is used to realistically verify analogue and digital circuitry as well as firmware.

7.5.2 The disk drive

The following section details some typical configurations of a disk drive and discusses how these translate into requirements for the associated electronics, see also Figure 7.12. A drive normally contains up to five rotating disks. This disk (stack of disks) is driven by the spindle motor, which is a brushless DC motor. The rotational velocity varies between 4200 and 15 000 revolutions per minute.[2]

The RW-head flies on an air cushion 10–50 nm above the disk surface. It is supported by the load-beam, which can be moved about its pivot by the so-called voice coil motor. This consists of a coil that lies in a fixed magnetic field provided by permanent magnets, see Figure 7.13. Any current through the coil results in a torque on the load-beam and thus a circular motion of the RW-head.

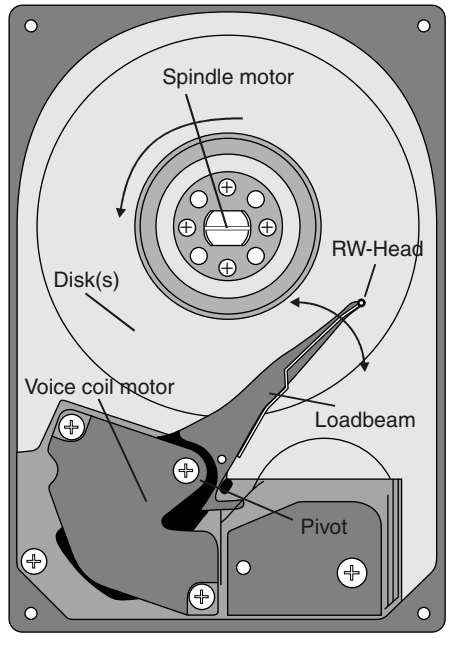

Figure 7.12 Hard disk drive overview

[2] Higher as well as lower rotational velocities are chosen at times, e.g. to cope with low latency or low noise requirements.

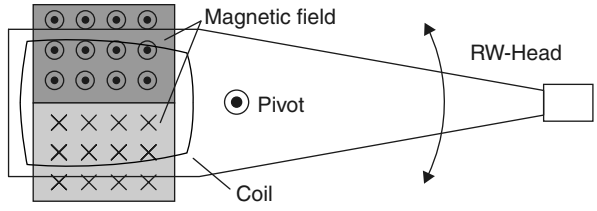

Figure 7.13 Voice coil motor

The data on the disk is organized in circular tracks. The outermost track may be located at a radius of 1.8 inch for a typical 3.5 inch form factor drive.[3] This means that the maximum length of a track is 11.3 inches, which gives rise to a linear speed of about 1880 inch/s for a 10 000 rev/min drive. This, together with a bit density within a track of 400 Kbit/inch, for example, in turn leads to a peak data rate of 750 Mbit/s. Interestingly enough, 1880 inch/s — in terms of the velocity of a car — is far beyond any speed limit, while bits of the length of about 60 nm are reliably read.

The track density of drives currently under design is in the range of 30 000–100 000 tracks per inch. Together with bit densities of 300–500 Kbit/inch this leads to surface densities of 9–50 Gbit/inch^2. Given the current track densities, the track pitch is in the range of 250 nm (100 000 tracks/inch) to 850 nm (30 000 tracks/inch). Controlling the head position to a precision of 10% of the track width, to make sure that most of the track width adds to the signal and to keep noise at bay, results in a required precision of 25 nm–85 nm when controlling the track following.

7.5.3 Circuit development for disk drives

Designing circuits for disk drives is currently an ASIC[4] business with a low number of customers, i.e. the disk drive companies, and high volumes. For instance, 223 million disk drives were shipped in the year 2000. This is still the case even though several attempts to create ASSPs[5] have been made in the past and are underway at the moment. The electronics of a disk drive — together with the related firmware — can be partitioned into a few functions:

- **Servo control**
  The servo control detects the current position of the head using so-called servo marks which are embedded into the tracks. Several hundreds of these marks are available throughout a full rotation of the disk. On the basis of this information the voice coil motor is controlled to allow for track following and seeking.

[3] Note that none of the dimensions of a 3.5 inch drive is actually 3.5 inch. The disk's form factor is just the same as for a 3.5 inch floppy drive.
[4] Application specific integrated circuit.
[5] Application specific standard product.

- **Spindle control**
  The spindle control detects the current rotational velocity, e.g. by means of back-emf[6] evaluation of the undriven phase of the brushless DC spindle motor. This information is taken into account in the control of the rotational velocity of spindle and disk stack. Spindle control also includes the (electronically controlled) commutation of the motor's phases and the implementation of sophisticated start-up algorithms. The problem with the start-up is that the brushless DC motor has no preferred rotation direction. However, if the disk rotates in the wrong direction the read/write-head might easily damage the disk's surface. So a great deal of work is underway into the sequence of the spindle motor start-up commutation to reliably provide for the correct rotation direction.

- **Signal processing**
  The signal processing function controls the data as it is written to and read from the disk. It encodes the data to be stored, and writes it to the read/write-head. When reading from the read/write-head, the data is pre-amplified and retrieved, using sophisticated techniques like the partial response maximum likelihood method. To allow for even more density — i.e. an even worse signal to noise ratio — the data on the disk contains additional information which is used for error correction. This often is based on Reed/Solomon techniques.

- **Buffer management**
  Data written to or read from the disk has to be fed through a buffer of substantial size to reliably rule out over- and under-runs. For this, DRAM of typically 0.5–2 MBytes is employed, which may be stand-alone or embedded in a system on a chip. In addition, a buffer manager is necessary that offers caching facilities and manages access to the memory for data buffering and other purposes, e.g. program execution from the DRAM.

- **Host interface**
  The host interface implements the communication with the host computer. It interprets the IDE or SCSI commands given and arranges their completion.

In the design process these functions are mapped to an architecture. The list of functions and the overall architecture is more or less generic and is the same for most of the disk drive designs, see Figure 7.14. The overall architecture of the disk electronics is based on five circuits, see Figure 7.14: motion-control, hard disk controller (HDC), RW-channel (stand-alone or embedded), pre-amp and DRAM (stand-alone or embedded). The HDC provides the micro-controller and also looks after the host interface and — together with the DRAM — the buffer management. The RW-Channel — together with the pre-amp — incorporates the signal processing function. Finally motion-control looks after the analogue and power side of servo

[6] Electro-motive force.

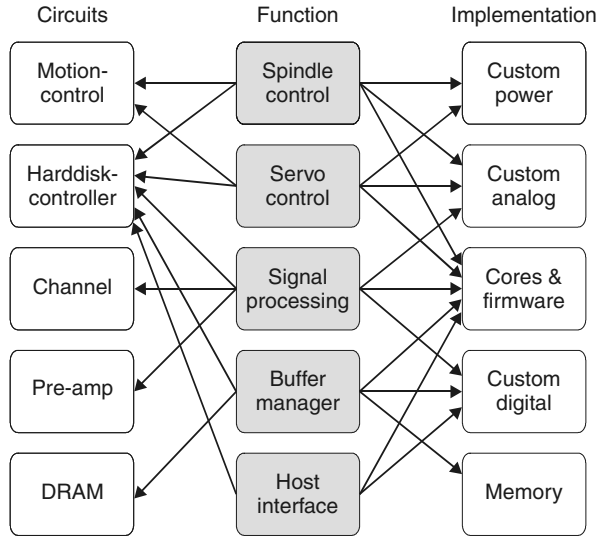

Figure 7.14 Typical mapping of functions to circuits and implementations

and spindle control. When it comes to the detailed architecture, a couple of trade-offs, e.g. performance (hardware) vs. flexibility (software), lead to a vast number of variants in use. For instance, for the control-related parts of the disk function, one may choose to use a standard micro-controller with some DSP-capabilities, e.g. multiply-accumulate operation. If performance plays a more important role, a DSP core may be added. Moreover, in some cases it makes sense to introduce application specific logic for the control tasks.

For the implementation, the classical options are available, i.e. processor cores with firmware, custom digital circuitry, custom analogue circuitry, custom power circuitry and memory. Figure 7.14 shows the mapping of the functions into their implementation.

Under the system-on-chip regime, the above five circuits may be arranged on three to five chips. The pre-amp will probably remain in the form of a stand-alone device in the future, since it is located on the load-beam, i.e. the shortest possible distance from the read/write-head. All other circuits are on the printed circuit board which is on the reverse of the disk drive. The motion-control circuit contains a couple of power devices for which integration into the digital standard process would not be cost-effective. Thus it probably will not be embedded into current or future SoCs.[7] On the other hand, the hard disk controller may be combined with an embedded channel and embedded DRAM on a chip to reduce package cost, save board space and increase the electrical performance. Figure 7.15 shows a couple of alternatives for the mapping of the five circuits to chips.

[7] System on a chip.

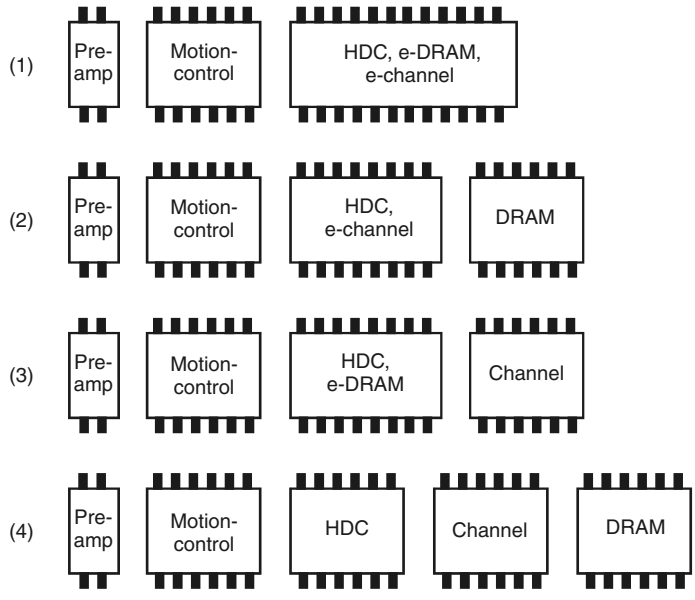

Figure 7.15 Several alternatives for the mapping of circuits to chips

7.5.4 The virtual disk drive

The design of circuits for disk drives has to take into account analogue–digital, hardware–software and electronics–mechanics interfaces at the same time. The Virtual Disk Drive was created to cope with that. It provides simulation models for all the parts of a disk drive and is meant to support an electronics product for a disk drive throughout its entire life-cycle.

- In the concept phase, high-level models form an executable specification which can be validated by simulation. This model also serves as a framework for design space exploration and even allows for early firmware development.
- In the design phase, the virtual prototype creates a general test bench. In this high-level system model, it is possible to 'zoom' into the components being designed, replacing high-level component models by their implementations. This may be done for any part of the system, whereas the system modelling has to be carried out just once. Moreover, this reveals information on the real-life system behaviour rather than more or less synthetic data on signals at a component's interface. For a disk drive the focus may, for instance, be on spin-up, track-following or seeking.
- The development of a high-level (sub)system model is also indispensable for the virtual testing of the analogue circuitry. The test program development may be carried out much earlier in combination with a model of the testing equipment.

- Finally, when the product is with the customer, i.e. a disk drive company, or even in the field, the analysis of spurious behaviour on the system model is much easier for application engineers, since every signal or quantity is visible. Unfortunately, this does not cover all possible faults of a device, since implementation related effects are not taken into account in system modelling. On the other hand, with adequate fault modelling it should be easy to prove or disprove any hypothesis that the application engineers may have on the root cause of system failures.

In the following, this will be illustrated on the basis of the example of the servo control of a disk drive system.

7.5.5 System modelling

A disk drive comprises digital and analogue/power electronics, firmware and mechanics which — as pointed out before — results in three major interfaces: hardware–software, analogue–digital and electronics–mechanics. These interfaces have to be handled at the same time, which does not necessarily mean that the complete drive has to be present in any system simulation, but for instance when assessing the servo control for controlling the track following or seeking, all interfaces are present, see Figure 7.16. Without loss of generality, we restrict ourselves to the servo control in the following.

The model comprises the seeking and track-following controller which is formulated in C and in the real system is implemented in firmware. Its inputs are the current track and the required track. Its output is a digital value for the required current to drive the head assembly. According to this value, the current is regulated

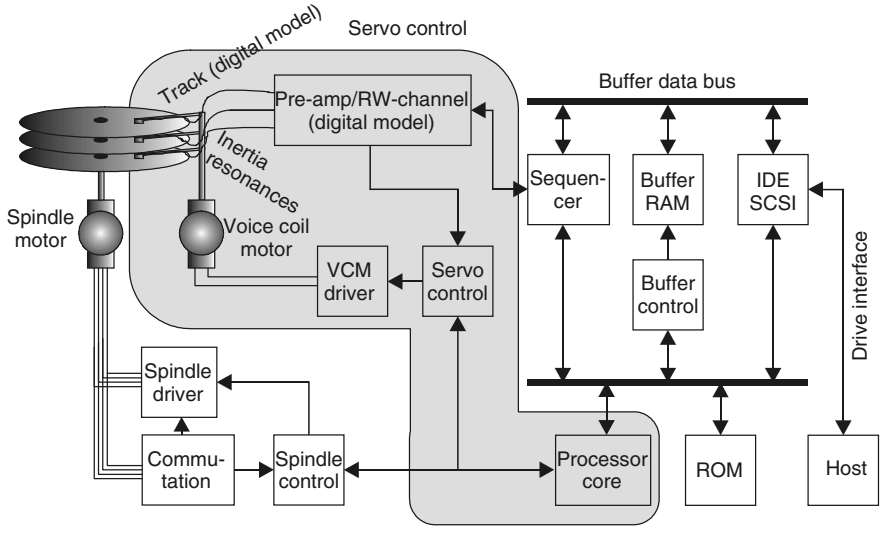

Figure 7.16 Disk drive schematic with servo control

by an analogue/power circuit, i.e. motion-control. The voice coil motor transforms the current into motion of the head assembly which may or may not contain resonances. The mechanical motion results in a track position which is fed into the firmware controller. The track content including track-id — as well as the logic to detect it — is reduced to a more or less trivial digital model, since it does not add to the overall function of the servo control.

The simulation of the servo control is performed on the basis of the mixed-mode simulator Saber. The analogue part of the system — motion-control and mechanics — are modelled in the analogue hardware description language MAST. Some basic digital modelling including the synchronisation between the regulator C-routine and the rest of the system is carried out using the digital capabilities of MAST.

7.5.6 Simulation and results

The high-level model of the servo control as described in the previous section can be used for concept engineering. For example, it is trivial to change the number of servo fields in a track, which at a fixed rotational speed determines the frequency of position measurements and thus the frequency of the digital controller. In the same manner, most of the other variables determining the function and performance of a drive may be varied. This can even be carried out systematically, e.g. by a parameter sweep or — if more than one parameter is involved — on a Monte-Carlo basis. All these variations can be simulated efficiently. For instance, the simulation of a long seek over 20 000 tracks takes about 80 CPU seconds for 20 ms real-time on a SUN Ultra 60 workstation, see Figure 7.17. One can easily spot the 'bang-bang' strategy of maximum acceleration, constant speed at the speed-limit and maximum deceleration to lock into the new track position. Incidentally, the speed limit is due to the fact that the read/write-head rides on an air cushion. This requires an air-stream in the direction of the tracks, which is created by the rotations of the disk(s). The movement for track seeking is perpendicular to that and could seriously disturb the mechanism described above beyond a certain speed. In the next step, the motion-control part of the servo control model was replaced by its implementation, represented by about 300 CMOS transistors and some DMOS power transistors. The real-life circuitry was thereby verified in The Virtual Disk Drive, which is far more meaningful than the results of classical analogue test benches. With the same configuration as the previous long-seek analysis, the simulation takes about 9 CPU hours. Note that most seek operations are performed much quicker and that track-following can be reasonably assessed in an even shorter period. In addition to the standard tasks of seeking and track following, it is now easily possible to review the implementation of special features, e.g. a request to park the heads in the landing zone.

The virtual testing of analogue circuitry is currently under evaluation. The development of a high-level system model is absolutely indispensable to this. This will

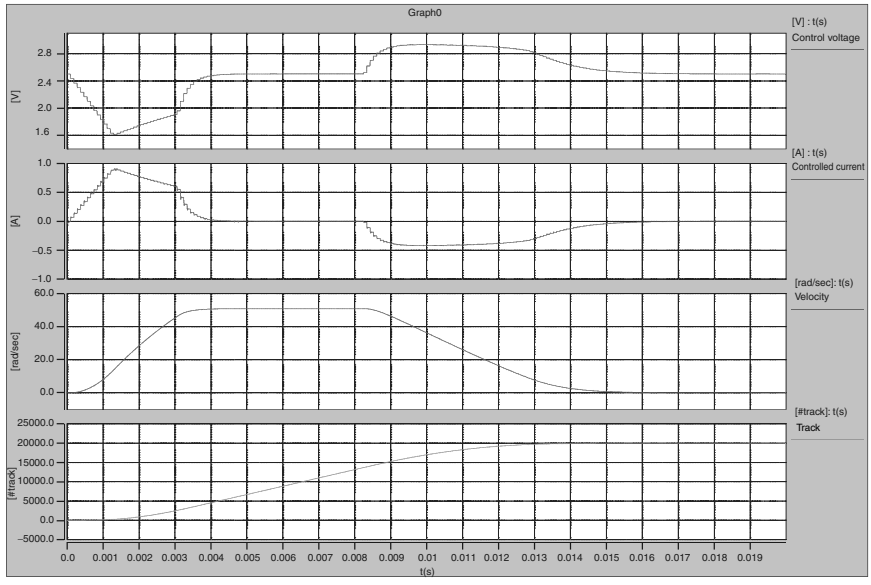

Figure 7.17 Simulation results for long seek over 20 000 tracks. From top to bottom: digital output of firmware controller, controlled current, velocity of rotational actuator and track-number

be combined with a model of the test equipment, which allows test program development to be initiated long before silicon is available.

The support of application engineering is mostly about analyzing the system to explain spurious behaviour, which in turn allows solutions to be devised and implemented. This includes fault modelling for the components of any domain. One might easily include, amongst many others:

- Head assembly resonances of certain frequencies (mechanics)
- On-resistances of power transistors higher than specified (analogue electronics)
- Timing problems in the serial communication between motion-control and disk controller (digital electronics)

This facilitates the assessment of the influence of the respective fault on system behaviour, which in turn can be compared to the behaviour observed in field.

7.5.7 Conclusion

The section has given an overview on state of the art disk drive technology and indicates how this translates into requirements for disk electronics. Moreover, it has shown how system modelling and simulation can support a related electronics product throughout its entire life-cycle. In that vein, the multi-domain nature of the system increases both the need for a systematic solution and the problems associated with modelling and simulation at the same time.

7.5.8 Acknowledgement

The author thanks his colleagues, Wolfgang Sereinig, Wolfgang Sauer, Felix Markhovsky and Scott Burnett for their valuable input and support.

7.6 Summary

Various examples of mechatronic systems have been presented in this chapter. These include electronics, mechanics and sometimes also software. In all cases a complete modelling could be achieved on the basis of hardware description languages, which means that the system in question can be investigated using a simulator.