CHAPTER 8

SOFTWARE

There are two areas in which software can influence the electromagnetic compatibility (EMC) of a system or circuit. One is in the programming of microprocessors or controllers in digital systems and the other is in the use of design software to predict EMC performance. The fit of software into a book of this title may seem odd, but at the code level the software can be considered as another component in the system. At the design level there is little other than to review the available software techniques that can be covered within the scope of this book.

The usefulness of changing certain aspects of program code can quite easily be observed to improve the EMC performance of a circuit. The usefulness of some of the design software available is more difficult to assess as it is often difficult to know how something may have been done without the influence of the software.

8.1 Programming Issues for EMC

A program within a system, microcontroller or even a simple logic controller can have an influence on both the immunity and emissions of a system. Immunity is easiest to consider as many error checking and code validation routines already exist to ensure that corrupted transmissions are not accepted. Hence many programmers are already aware of the issues of software immunity, although they may not have considered this as an EMC issue.

On the emissions side it is more likely to be a case of acknowledging the best practices for addressing and operating the hardware from software codes. For example, setting a UART to tristate from input, then to outputs may reduce the transient power demand compared with a direct input to output change of state. This is generally known as defensive programming.

Many techniques, as with the watchdog circuit (see below and the section on ICs), do not actually improve emissions or immunity but simply provide a controlled method of recovery. Techniques involving simple known recovery states are usually less program intensive than performance improving techniques. The latter tend to have a higher programming overhead (i.e. require more code to execute) and more memory for storage.

8.1.1 Watchdog Programming

When incorporating a watchdog circuit (see IC section) into a design, some programming may be required for the timer reset of the watchdog. If possible the code should poll the watchdog pin once per program cycle. This is not too much of a problem with short microcontroller code or sequential routines, but with general operation processors and long programs this may be difficult. There may therefore need to be additional timed interrupts or hardware-generated interrupts to address the watchdog (e.g. dedicated watchdog interrupt handling code, Figure 8.1).

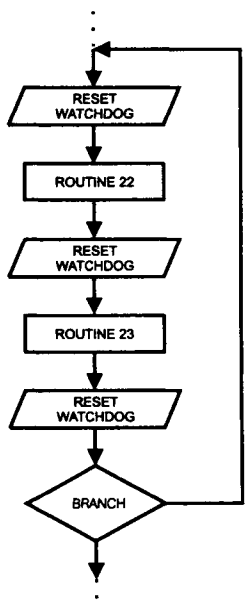

Figure 8.1
Sequential watchdog resets

The watchdog timer must be short enough to ensure non-catastrophic failure of the circuit and long enough not to interfere with functionality of the program (durations between 10 ms and 2 s are typical for watchdog timers). The trade-off in timing can be difficult to gauge correctly for some coding. Processors with a Harvard bus structure (independent data and instruction busses) are easier in this respect as each instruction is known to require a single clock cycle; therefore code timing is simple, count the instructions and divide by the clock frequency:

$$watchdog\ time\ out = \frac{number\ of\ instruction\ cycles}{watchdog\ clock\ frequency} \qquad 8.1$$

Another potential problem could occur if the watchdog code is not included within the sleep cycle for those processors with low power sleep modes. When the processor goes into sleep mode and the internal clock frequency is reduced, unless the watchdog interrupt or poll function is similarly adjusted, a reset could be implemented simply because the sleep mode was activated.

Watchdog programming has memory overhead as it requires additional program routines or interrupts. This may not be possible with fixed ROM microcontrollers and a hardware solution may be required. Likewise, in real time operating systems (RTOS), interrupts or software-based polling may not be possible due to their effect on program functionality. One simple hardware solution is to monitor lines that are known to change frequently, such as the lowest order address or data line, and use this for the watchdog polling line (Figure 8.2). The biggest problem here is ensuring that these lines will always change with variable code and within the watchdog timing regime. The advantage of hardware is that it requires no programming, therefore less memory. The disadvantage is the increase in component count to implement the function. It is most likely to cost more to implement the watchdog function in software than in hardware, but code is generally the best implementation, especially as a crashed program could continue to toggle the lines a hardware-based watchdog system is monitoring.

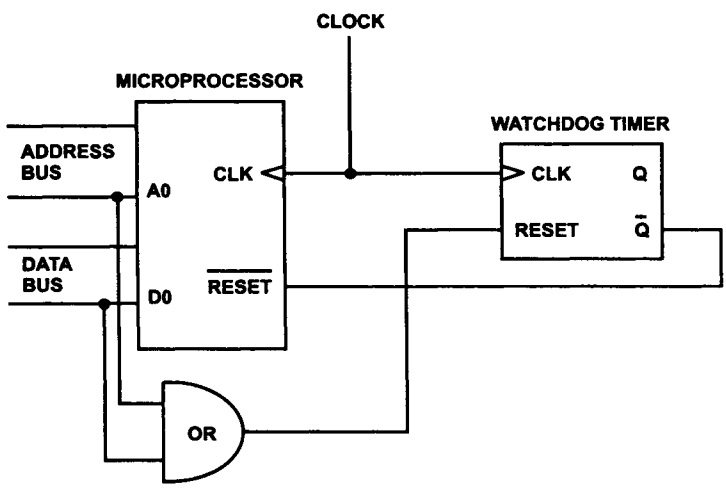

Figure 8.2
Hardware-generated watchdog interrupt

The most effective reset coding for a watchdog timer is to dedicate an output pin of the microprocessor to the watchdog (Figure 8.3). This pin should be set and cleared alternately as the program routines are executed (Figure 8.4). By using an AC coupled watchdog set and reset, if the code loops within a single routine that includes a watchdog

set command, the watchdog is still effective and this single port condition status will operate the reset. This effectively doubles the robustness of the watchdog with only a small code overhead (alternating port set and port clear commands).

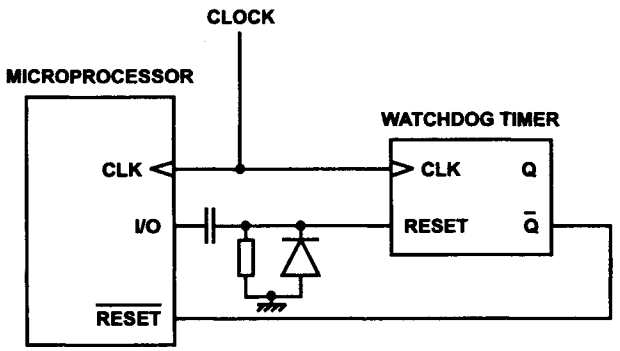

Figure 8.3
AC coupled watchdog circuit

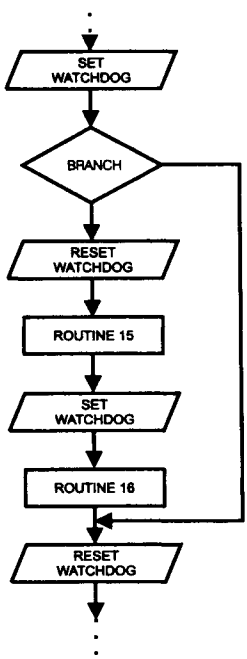

Figure 8.4
Watchdog set/reset routines

8.1.2 Refresh Port Connections

The data direction registers and the input/output port data registers are usually located near the edge of the processor's package if included in the microprocessor and may be connected directly to the external circuits. Consequently, these are highly likely to have noise on their lines. A simple way of minimising this noise disturbing the data settings and propagating into the microprocessor integrated circuit (IC) or system is to refresh these registers regularly.

The action of the microprocessor rewriting these data registers stabilises the interfaced circuits and minimises the risk that noise on these ports is corrupting other internal registers. This is a simple task to perform regularly and involves minimal programming overhead, just occasional write commands to the ports and data registers.

Some care does need exercising to ensure that writing the port status is appropriate to the program activity at each re-write command. It can not be assumed that the value in the port status is the correct setting so a read and then re-write could result in enforcing an erroneous setting.

8.1.3 Polling Interface Ports (Oversampling)

Input and output to interfaced functions usually occurs at a much slower rate than the microprocessor clock. This enables the microprocessor to poll these pins several times to ensure that either the level is set for an output or that the incoming signal is stable. This is analogous with the types of oversampling used in digital audio circuits (e.g. CD players).

The microprocessor can be programmed with an oversampling scheme, either to ignore the first and last and take the mean of a middle sample for instance, or simple average. The actual scheme for polling can be left to the programmer and will depend on the type of interface being addressed (e.g. audio, keyboard, serial data link) and the processing rate.

This type of programming has a major software and run time overhead as it requires several repeats of an operation plus usually some mathematical interpretation. The result is an improved immunity to noisy input signals and less susceptible output ports. This may best be implemented using a dedicated input/output microcontroller with this type of software programmed on-chip rather than as an auxiliary program for a main processor.

Dedicated input/output interface controllers are quite popular in larger systems hence adding local code to improve the immunity should be relatively easy. The same code could then be used for any system with that type of interface controller making the immunity improvement portable to other systems.

8.1.4 Token Passing

Token passing is a deliberate method of ensuring the program is progressing in a controlled manner and has not been jumped into due to a code corruption. This

requires additional programming steps at each critical subroutine or program block within the final program.

Token passing requires a token (value) to be either updated in memory or in a register. As each routine is called it checks to ensure that the call is from the previous set of code by comparing the token with a programmed value or memory location. If a jump has occurred prematurely the token will not be set correctly, likewise if the call has come from a random corruption the token will be in the wrong state. On the discovery of a wrong token a reset routine is initiated.

The advantages of token passing over other methods are that the program continuously looks for errors rather than waits for an error to occur and the token itself can be used to reset to a predefined location in the program. It may not be necessary to completely reset the system depending on the location of the error and the token value, this obviously saves program time but also gives a genuine level of immunity rather than a simple reset on error recovery. The extra programming overhead is in adding the compare program to the start of each sub-routine and the token update to the end, plus any error handling software control if system reset is to be avoided.

```
Code            Comment
...
:sub21          start of sub routine
LDA    &token   load accumulator with token
LDB    &value   load correct token value
CMP             compare token and value
BNE    #reset   jump to reset routine if token incorrect
XXX             otherwise proceed with program
...
XXX
LDA    &token   load token
INC             increment token value for next routine
STA    &token   save token
:end_sub21      end of sub routine program
...
```

If attempting to use the token passing technique to recover to a known program position more programming effort will be required and probably several tokens to ensure register and flag conditions are suitable for re-entry into areas of code. Using the token value alone is insufficient to reset routines where register or memory writes have occurred as these will most likely be incorrectly set.

8.1.5 Unused Memory Addresses

In most applications it is impossible to exactly fill all memory locations, either ROM or RAM. These locations could be accessed due to a software or hardware addressing error caused by a corrupted code. These unused memory locations should therefore have a known operation placed in them, usually either a no operation (NOP) or an unconditional jump to a reset routine (JMP RESET).

```
Address        Code          Comment
0000           XXX           functional program area
...
00FA           XXX           end of program area
00FB           NOP           start of unused memory block
00FC           NOP
...
010F           JMP RESET     jump to reset routine
0110           XXX           functional program area
...
```

Where blocks of memory are unused, the jump to reset should be at the bottom of the memory with no operation commands above. These blocks may be scattered about the ROM/RAM locations and need to be carefully mapped if this type of feature is to be used.

Writing these areas will be required by the initialisation program if they are in RAM locations. There is of course the possibility that a corrupted write code could rewrite these NOP instructions in the RAM space, similarly with the program itself, but it is impossible to program for all eventualities and a reasonable trade between effectiveness and efficiency of programming has to be drawn.

This technique requires only a small amount of additional programming to the start-up (boot) routine and should not require any run-time program overhead.

8.1.6 Code Ghosting

This is one of the most memory intensive and possibly expensive techniques, usually reserved for fault tolerant software. The technique requires each code or data value to be stored in two parallel locations (either in identical or in complement form). The code or data are then compared with their ghost value prior to use.

The potential for electromagnetic interference (EMI) causing both codes to be corrupted is extremely low; however, the microprocessor still has no way of knowing which of the two codes is correct and which corrupt. The microprocessor would have to handle a code mismatch by reloading to check if the error was in the load operation or abort to a known routine if the error is in the stored memory values.

```
Code            Comment
...
:sub32          start of sub routine
LDA    &code    load accumulator with code value
LDB    &ghost   load ghost value
CMP             compare value and ghost
BNE    #error   jump to error routine if values do not match
XXX             otherwise proceed with program
...
```

This method is a hardware implementation of the oversampling technique and could be extended if necessary, although at some cost if there is a lot of code (i.e. three or

four copies could be used for comparison). If multiple ghost values are used a mathematical method of selecting the correct value could be applied (e.g. numerical digital average of each bit). The method can also be selectively applied to code or data values from a known problematic location or EMC critical area of the system or program.

8.1.7 Other Techniques

There are other techniques that are more specific to the hardware being used and the best methods of operating interfaces and associated circuits. For example the interface with a tristate setting mentioned in the first part of this section. If a setting of the port from logical outputs to inputs causes a large switch in internal states of an IC, resulting in a large current demand, but an intermediate state is available that has only a fraction of the transient current demand when switched (Figure 8.5). Using the intermediate state between transitions requires only one additional code instruction and will reduce transient supply demand and hence conducted noise levels.

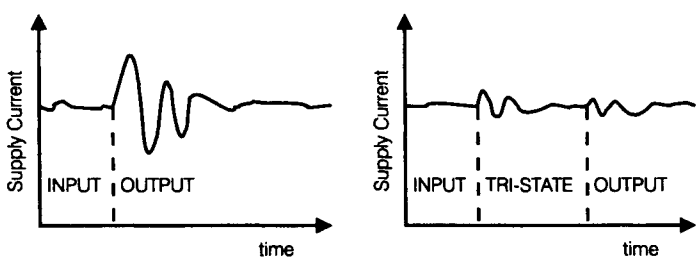

Figure 8.5
Effect of port status changes on supply current

The intermediate state transition technique may be applicable to other circuits, such as clearing selective flags prior to rewriting may save the number of actual data or address lines driven and therefore again reduce transient demand. Actual transient savings in many circuits will be negligible and the idea is best reserved for those functions which are known to require larger current supply (e.g. line drivers, interface circuits and bi-directional ports).

Another technique that can help with interfaces is to use a coding scheme, such as Manchester coding, which has only a few frequencies in the transmission. In Manchester coding only two frequencies are used, a '1' is indicated by a single transition and a '0' by a double transition within the clock cycle (Figure 8.6). The signal is therefore always changing state, hence a latched condition or end of

transmission are also easy to detect. The receiving circuit can potentially determine if the sender is experiencing EMC problems. With standard non-return to zero (NRZ) coding the frequencies present in the transmission can be from the clock frequency to whatever the maximum bit code length period is (i.e. 1/8 the clock frequency for an eight-bit code, Figure 8.7).

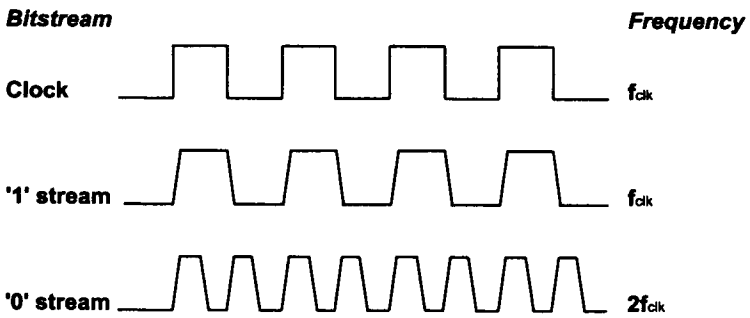

Figure 8.6
Manchester code

Manchester coding makes filtering easier, if required, and allows for a simpler method of triggering on levels rather than edges that will also improve immunity. The clock of the transmission circuit is derivable from the signal hence, using a fixed offset delay, a level triggered receive circuit is easy to implement in either software (using a delay routine) or in hardware (using a delay line).

Converting standard edge triggered circuits to level triggered is possible at specific ports by using a software code delay to allow the value to settle prior to reading. The

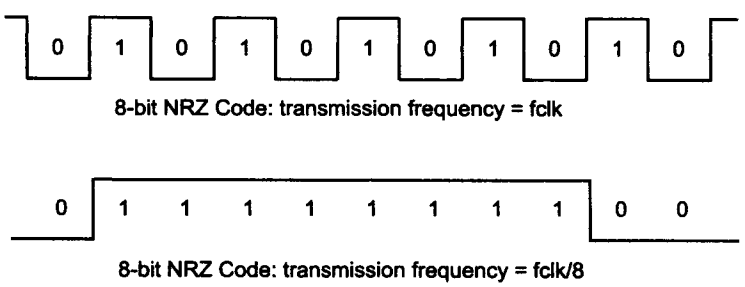

Figure 8.7
NRZ coding

length of delay will depend on the function, but should allow the ringing or overshoot to settle and consequently not affect the value read at the port (Figure 8.8), a typical settling time should be 10% of the maximum clock period. This can also be implemented in hardware using a delayed latch trigger with the data ready signal coming from the delayed latch set condition (Figure 8.9).

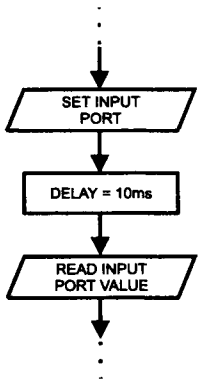

Figure 8.8
Software delayed port read

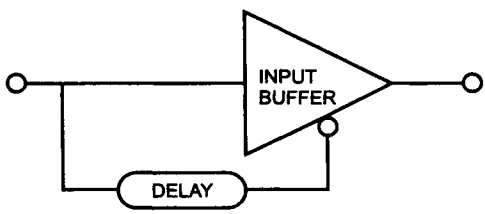

Figure 8.9
Hardware delayed signal

8.2 Design Software

Software tools for design for EMC tends to fall into two distinct categories of program, one is the simulator the other the advisor. The simulator attempts to predict precise values of field or conducted noise at any chosen point in a circuit system or structure. The advisor usually consists of a knowledge base and is applied to an otherwise complete design to check its compliance with the rules of the knowledge base.

The effectiveness of the software is difficult to gauge accurately without comparison with measured results and unfortunately little comparative testing with finished

circuits is recorded. Most of the packages currently available concentrate on the physical side of circuit design, being concerned with the dimensional construction of PCBs and component packages. To date simulation of circuit behaviour and component performance is still performed where possible on existing standard circuit simulators (e.g. SPICE).

8.2.1 PCB Design Software

PCB design software is available from a bewildering number of suppliers with a long list of various features of each package to differentiate it from its rivals. This is usually one area of design that is almost exclusively done by computer, even the simplest of circuits, designed on the back of an envelope, bread boarded rather than simulated, is still usually laid out for production in a PCB design package.

The PCB design software does allow the designer to control more of the EMC performance than component selection, as the designer can choose the dimensions of interconnect, the number of layers present and their use. One feature that is now prevalent on PCB design packages that is a potential problem for EMC is the auto-router. The PCB design software does not necessarily know which are the fastest signals and therefore which require priority in the layout, consequently initial clock and high speed tracking may have to be done manually with the auto-router used only for the bias components, low speed and DC tracks.

The major thrust for advisor type software for EMC is in the area of PCB design and these can often operate on prelaid designs and give an indication of likely problem

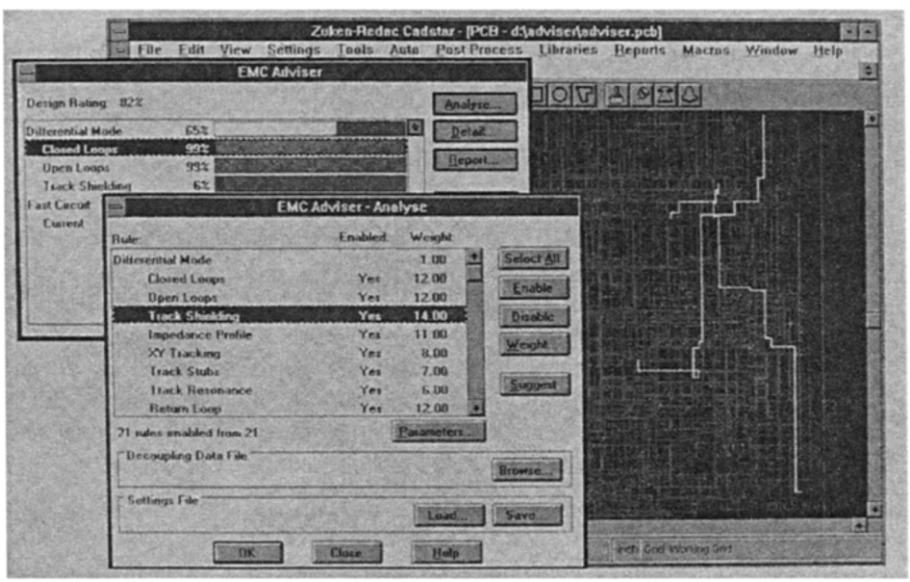

Figure 8.10

EMC Advisor Analysis Software System from Zuken-Redac

areas. Often the advisor may need some considerable input from the user to establish some rules of layout specific to the user's applications. The amount of input can be tedious and many advisor programs are little more than an additional design rule checker (DRC) added to the existing DRC program. In fact if your PCB software has an edit function for its DRC it is possible to create your own EMC advisor.

Simulation software at the PCB level takes the physical dimensions of the tracks and interconnect (again coupled with data on the PCB material, see Appendix B) to construct a transmission line model or aerial model for interconnect. This usually then requires the component model data to produce some idea of likely delays, propagation characteristics, termination values, etc. to predict if the layout will cause functional problems (usually determined by signal integrity) or EMC problems. Usually, the software is a conventional circuit simulator (electrical for conducted noise and electromagnetic (EM) for radiation) and the PCB data are extracted from a PCB layout package to produce the relevant interconnect models only. This, therefore, can require integration of software packages: a PCB layout package, a PCB interconnect model extraction tool, a circuit simulator and EM radiation modelling package.

As with any simulation the result is only as good as the models allow. In other words the data on the PCB characteristics and dimensions have to be accurate for the supplier you use. Another problem with this type of simulation is the time required for extraction and analysis. Consider a small PCB with 10 ICs in 14-pin packages, there are at least 140 interconnections as well as the components to simulate. Consequently, these simulation models can usually only be run on the few critical interconnects to check integrity and functionality within a reasonable simulation time, especially if the radiation is being examined.

The above simulator arguments generally apply to predicting emissions for which some models do exist. The susceptibility is, however, much more difficult, little component data are available and the EM software is highly complex for such tasks. In general, the assumption that low emission designs offer low susceptibility is usually used.

8.2.2 Component Simulation Software

At the component level electrical circuit simulators have been available for a long time and several *de facto* standard packages exist. For electrical simulation of analogue circuits SPICE has gained a wide acceptance and many examples of its accuracy exist as do many models of passive, discrete and integrated components. Digital circuit simulators are also common, but do not lend themselves to encompass the possibility of noise analysis for EMC as an analogue simulator does. Consequently, extensions to digital circuits to allow them to interface to analogue or mixed mode simulators is now a relatively common theme, also digital models in SPICE are available.

The main problems are that the EMC performance is often outside the normal operating range of the components, and hence their model. Some models do exist but component packaging models and interconnect (see section on PCB software) also have a significant effect on the EMC performance and models for these are not quite as common.

An area in which component simulation lacks behind PCB simulation is in emissions modelling. The component package is not usually within the control of the circuit designer, they require the device's function not its package. Consequently, unless the component supplier can provide package data (dimensions of tracking and bonding as well as electrical parasitics), modelling emissions from components is not feasible for a component user. Even the supplier will experience difficulty and the use of any model derived would be very limited as few people are attempting this level of modelling or have the time or facilities to do so. The EM software that can perform this level of modelling is expensive and requires some expertise to operate and a significant computing performance from its host platform.

An emerging modelling standard that will assist in the conducted emissions simulation is IBIS (Input/output Buffer Information Specification), which includes package electrical parasitic data and rise and fall time information. The device models are relatively basic in their functional performance within the IC, but give reasonably detailed information of the behaviour of the signals at the pins of the device. The models offer one of the best solutions to mixed signal simulations as well as offering possible EMC simulation data and without compromising the IC manufacturers' circuit details. As with any model there are limitations, and therefore due to the simplicity of the model it is unlikely to predict accurately reaction to incident phenomena, such as high frequency conducted input noise signals or accurately model transients impact behaviour (although some transient absorption models are included in some devices) and EM emissions are not modelled.

8.2.3 Design Software Overview

There is still a gap between component and PCB design software, although both are commonly used by circuit designers, they are considered disparate operations in the design cycle. If software is going to be used to predict circuit level EMC performance it is going to have to link the component, circuit and PCB information.

Some software vendors are offering the possibility of taking PCB files and adding models to a circuit simulator to model interconnect. There does need to be some feedback if the PCB layout is then going to determine which tracks carry the highest speed signals, therefore which to prioritise for layout. The simulator link approach requires collaboration between simulation and PCB CAD vendors, this is already occurring as the ECAD vendor base reduces by merger and acquisition. The main drawback with simulator linked PCB layout software is the cost and complexity of linking these packages together; each has a primary function which is not necessarily compatible with the others'. This applies to both the analogue and digital circuit simulators currently available as well as PCB layout packages.

Another consideration for the ECAD supplier is are designers willing to pay for EMC add-ons to their software and are they willing to trust the results? No original equipment manufacturer (OEM) would go to market with a product simulation and no test data. As testing will still be required the value of a simulation tool will be based on a trade-off between convenience, accuracy and of course cost.

The value of simulation should be in reducing the design time. This requires the correct set of design rules and models within the simulator and the correct application to both circuits and systems. As in the early days of analogue and digital simulations, the loop closing the simulation versus measured performance still needs completing for EM software to enable designers to believe in the results the software provides. This is being performed and the software is continually being refined. Eventually the simulation will provide adequate information on EMC performance and tests will be performed solely as verification rather than an iterative process in the design cycle.

8.2.4 Available Commercial Packages

The following tables (Tables 8.1–8.4) list some of the available software packages aimed at EMC simulation or modelling. There are no doubt others that offer various features and new packages are appearing each year. There is no best solution as it will depend upon many factors, not least of all the price of the software.

Table 8.1 *Advisor-based software*

Product name	Supplier	Application
Design Advisor	Zuken-Redac	PCB DRC
EMC Expert Consultant	Seaward Ltd	General
EMC Toolkit	Continental Compliance	General
UniSolve	UniCAD	PCB DRC

Table 8.2 *Conducted emissions modelling software*

Product name	Supplier	Method	Application
em	Sonnet Software	MOM	circuits semiconductors
EMA3D	Electro Magnetic Applications	FDTD	3D structures
EMC Workbench	Incases	TLM/MOM	PCB design
EMIT	Altium	MOM/FDTD	conductors
Greenfield 2D	Quantic Laboratories	TLM	signal integrity PCB design
L-Edit/EM	Tanner Research	BEM	PCB layout
Maxwell Strata	Ansoft	MOM	signal integrity, cross-talk
Micro Stripes	Kimberley Communication Consultants Ltd	TLM	3S structures antennas
Motive	Quad Design	TLM	PCB layout
Superstar	Eagleware	TLM	PCB design

Table 8.3 *Radiated emissions modelling software*

Product name	Supplier	Method	Application
ContecRADIA	Contec Microelectronics	Antenna Mathematics	PCB traces, wires and structures
Greenfield 3D	Quantic Laboratories	BEM	3D structures and PCB traces
MAFIA	Computer Systems Technologies	FDTD	3D structures
Maxwell SI Eminance	Ansoft	FEM	3D structures
Momentum	HP-EEsof	MOM	planar structures
MSC/EMAS	MacNeal-Schwendler Corporation	FEM	3D structures, antennas, EMI and cross-talk
QUIET	Quad Design	MOM	Any 2D and 3D structures

Table 8.4 *General electromagnetic modelling software*

Product name	Supplier	Method	Application
HFSS	HP-EEsof	FEM	3D structures
Maxwell Eminence	Ansoft	FEM	3D structures
OERSTED	Integrated Engineering Software	BEM	time harmonic electromagnetic fields
TOSCA	Vector Fields	FEM	3D structures

There are many ways to handle EM simulation, the most complex is the finite element method (FEM) which is commonly used in mechanical stress design software. The EM versions solve Maxwell's equations in space along a grid or mesh either defined by the user or by the software. FEM is arguably the most accurate method but also the most computationally intensive and the most expensive. FEM can be difficult to use but is very versatile and can be applied to almost any structure. The boundary element method (BEM) is similar but deals with boundary and space conditions rather than within finite elements of a grid, it is therefore a little faster and of similar accuracy to FEM. The transmission line matrix (TLM) is one of the faster methods that attempts to create two- or three-dimensional transmission lines in space and calculate fields at various points based on transmission line equations. The TLM method is fast but not as accurate as BEM or FEM for radiated emissions.

Other methods exist for solving the radiated fields or conducted noise within circuits and structures. They all offer some feature that makes them different from the next, but usually each has a trade-off in speed, price, accuracy and versatility. Although FEM seems the 'best' solution, there are certain applications where the other

techniques can yield results of similar accuracy with faster computation times and at a lower cost in both software and in time spent on the model construction. For arbitrary geometries and non-linear materials FEM may be the only suitable method. As with any simulator, their accuracy ultimately lies with the accuracy of the models contained within the software or constructed by the user.

A list of the software vendor addresses is given in Appendix C.