Chapter 3

PROCESSOR SELECTION

3.1 Introduction

Traditionally, tabular method has been used for the selection of various components [5, 6, 10, 18, and 24]. In this method, for each processor, processor specifications are evaluated and ranked with other alternatives and a comparative table is prepared based on the importance of the parameters for a specific application of the embedded system. The designer selects the processor that is on the top of the list. Being manual it is manageable when the numbers of components are few but is quite difficult for very large number of components. Further new processors are continuously added and new applications of embedded systems are forever growing. Both these aspects make manual selection of processors all the more disadvantageous. Precious time of the designer is spent in choosing a processor. This chapter proposes a Hybrid Algorithm for the processor selection.

3.2 Proposed Algorithm

A Hybrid Algorithm is proposed to automate the selection of the processor. This involves the following steps:
1. A random processor is chosen and the application is run using Keil Integrated Development Environment (IDE). The output of the tool gives the parameters of the application.

2. The extracted parameters are mapped to the parameters of the processors as specified by the vendors.

3. The extracted application parameters are used to normalize the processor database.

4. For Weighted Sum Algorithm (WSA), the user is asked to specify weights for the parameters. For Kepner-Tregoe (KT) method the designer is asked to specify the musts and weigh the wants.

5. WSA and KT method are executed using the weights. These algorithms select the most suitable processors (three) from the database.

6. The application is run using the Keil IDE for each of the selected processors and the performance of the chosen processors is compared.

The flowchart shown in Figure 3.1 depicts the proposed Hybrid Algorithm. As mentioned above we use Keil IDE for extraction of application characteristics. Keil is one of the widely used IDE for embedded systems development and it integrates many tools such as: C Compilers, Macro Assemblers, Debuggers and Real-time Kernels to support ARM, and 8051 architectures. User can create the application (either written in C or Assembly), for which characteristics are to be extracted, through the project management utility provided in Keil IDE or High Tech IDE. In our experiments we noticed that both these IDE's gave almost the same results after compilation and hence the rest of the work deals with the Keil IDE.
Run the Application with Keil IDE

Extract the Application Characteristics

Map the Extracted Characteristics into Processor Specifications

Choose the Algorithm to Select the Processor

WSA
Select the Top Three Matched Processors

KT
Select the Top Three Matched Processors

Analyze the results after running Application with Selected Processors and Choose the Best

Stop

Figure 3.1: Hybrid Algorithm for Processor Selection.
3.3 Processor Parameters

In this study, we consider only 8 bit microcontrollers with 16 bit address lines. We have created a database of more than 200 microcontrollers from four major vendors Atmel, Intel, Infenion, and Philips. Out of the different parameters we have concentrated on those which are relevant to a large number of applications. As a result, some applications which need ADC, Interrupt Controller, and Watch Dog Timers are not catered to in this study. We have considered Flash Memory, Internal RAM (IRAM), EPROM, I/O Lines, Buses, MIPS, Timers, Universal Asynchronous Receiver and Transmitter (UART), Data Pointer (DPTR) and In System Programmer (ISP) for the processor selection.

3.3.1 Extraction of Application Characteristics

The IDE environment shown in Figure 3.2 expects that a project be created by the designer to run the application. After creation of the project, the application is compiled to extract the memory requirements and the requirement of the on-chip peripherals is simulated. In order to do so, we have to run the application using some processor. Notice that this choice of the processor has no bearing on the final processor selection. Since applications have different parameter requirement, we run the application on a processor which has most of the peripherals on the chip. As a result, different requirements of different applications can be gauged. Keil IDE provides the facility to simulate different on-chip peripherals like I/O Ports, Timers, Interrupts, ADC and DAC etc. Before running the application, we select all these peripherals and observe the result and the snapshot of the tool with the simulation features of peripherals is shown in the Figure 3.2. Thus, we are able to identify the peripherals that are actually required by this application.
3.3.2 Mapping of Extracted Parameters

The extracted parameters from Keil IDE are code, data, xdata, execution time, timers, I/O lines etc. However, the processors in the database have different specifications. Here we show how extracted parameters are mapped on to the processor specifications.

**Code:** It refers to the program memory whose maximum size is 64 K Bytes in microcontroller. This may be Flash or EPROM or EEPROM. The designer is asked to choose which one to use for the application.

**Data:** It refers to directly addressable internal data memory which gives fastest access to variables. This to IRAM is mapped.
**Xdata:** It refers to external data memory whose maximum size can be 64 K. External data memory is read/write. Access to external data is slower than access to internal data memory because the external data memory is indirectly accessed through a data pointer register. This refers to the XRAM of the processor specifications.

**Execution Time:** The μVision2 Performance Analyzer (Keil IDE) displays the execution time recorded for given application. This helps to estimate the frequency of the processor required for running the application (\( \text{time} = 1/\text{f} \)). We further convert it into MIPS to select the processor. Remaining parameters like Timers, I/O lines etc are directly mapped to the processor specifications without any interpretations.

It may be noted that different processor parameters have a lower and an upper bound as shown in Table 3.1. It is possible that extracted specifications are outside this range. In case the extracted specification values are lesser than the lower bound then it is recommended to choose the lower bound value. However, in case the extracted specification values are greater than the upper bound then it is recommended to choose the external components to meet the requirements of the application.

<table>
<thead>
<tr>
<th></th>
<th>Flash (K. Bytes)</th>
<th>IRAM (Bytes)</th>
<th>XRAM (Bytes)</th>
<th>EEROM (Bytes)</th>
<th>I/O Lines</th>
<th>Buses</th>
<th>MIPS</th>
<th>Timers</th>
<th>UART</th>
<th>DPTR</th>
<th>ISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>64</td>
<td>256</td>
<td>2792</td>
<td>131072</td>
<td>80</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1: Range (upper and lower bound of the processors specifications).
3.4 Normalization

We need to search the processor database for the processor that matches the application characteristics extracted with the help of Keil IDE. It is noted that different parameters of the processor have different ranges. If the actual values are used in any ranking algorithm, then the parameters with large range of values of the processor will dominate the result. Therefore, we normalize the processor database using the following general principle. If a processor parameter exactly matches with application characteristic then the value is 1. If it matches 99 % (on either side) then the value 0.99 is assigned. As we move away from the exact value, the normalized value will tend towards 0. The algorithm used for normalization is shown in Figure 3.3. Table 3.2 depicts the normalized values of the two processors.

```
if (N(j,i)==R(i))
a(j,i)=W(i);
elseif ((N(j,i)>(R(i)*0.995))&&(N(j,i)<=(R(i)*0.995)))
a(j,i)=0.995*W(i);
elseif ((N(j,i)>(R(i)*0.99))&&(N(j,i)<=(R(i)*0.99)))
a(j,i)=0.99*W(i);
elseif ((N(j,i)>(R(i)*0.98))&&(N(j,i)<=(R(i)*0.98)))
.................
elseif ((N(j,i)>(R(i)*0.1))&&(N(j,i)<=(R(i)*0.1)))
a(j,i)=0.1*W(i);
else
a(j,i)=0;
```

Figure 3.3: Algorithm used for Normalization.
Table 3.2: Normalized Values of two Processors.

However, in some cases normalization is not done on the lower side where it falls short of the requirement. Consider a parameter like Timers. If the application requires two timers, then a processor having only one timer cannot be used. An ideal processor would be the one with two timers. However, a processor with more than 2 timers can also be used. Therefore, while normalizing for timers we do not consider a value lower than the required one but only the one on the higher side. Such parameters where a minimum number is required to run the application are Buses, UART, ISP, Timers Flash or EPROM and I/O lines. On the other hand, the application can run on a processor where the values of RAM and MIPS are less than the application requirements. Therefore, for these parameters we normalize on either side.

3.5 Processor Selection Algorithms

In this section we explain the two techniques that are used for the processor selection. In the end we compare the results obtained by these algorithms.
3.5.1 WSA

Weighted sum algorithm has been used for different applications. For example, Schoeber [26] has used WSA for RTOS selection. We have adopted this technique for processor selection and enhanced it with scaling and dynamic weight assignment (WSA). We are not aware of any work on this method for processor selection. The developer or designer chooses the weights to be assigned for each parameter of the processor as per the importance of the project. The designer can choose the weights between 0 and 1. Let \( W_i \) be the weight of each parameter and \( F_i \) be the normalized values of the actual specifications of the processor as derived in section 3.4.

The algorithm is given below:

1. Calculate the overall weighted sum for each processor in the database using the equation given below where \( S \) is the score or cumulated weighted sum of a processor.

\[
S = \sum_{i=0}^{11} (W_i F_i) \quad (1)
\]

2. Choose three processors which have the highest score and display their non-normalized parameters for verification.

Assume that there exists an ideal processor which exactly matches the application specifications. That is, \( F_i \) is 1 for each parameter. Compute the ideal weighted sum using the formula of step 1. Compare the ideal processor with each of the processor selected in step 2. The deviation from the ideal can then be viewed. The entire implementation details of the WSA are depicted with the help of flowchart in Figure 3.4.
Figure 3.4: Flowchart of the WSA.
3.5.2 KT Method

The Kepner-Tregoe [11] process divides the decision-making process into six logical steps. The activity for each step for processor selection is explained below.

1. *State the Purpose:* Processor selection for the embedded systems.

2. *Establish Objectives:* The objectives of processor selection are analyzed and 11 parameters are considered for processor selection as given in section 3.3.

3. *Classify Objectives:* Separate the objectives into Musts and Wants based on the requirement of the application. The Musts and Wants are specified by the designer.

4. *Weigh the Wants:* Rate importance of each Want on a scale of 0 to 1 for processor selection. In this step the designer specifies the weights in a manner similar to WSA.

5. *Compare Alternatives:* Select the processors that match Musts from the normalized database which is created as explained in section 3.4. These are now the candidate processors for selection. Apply the equation 1 to calculate the score S for these processors.

6. *Choose the Best Course of Action:* Select the top three processors which have the highest scores and verify the results further with the evaluation tool.

The designer will have an option to choose any of these algorithms to select the processor for a given application. WSA is used to select the best fit processor that just matches the application requirements and KT is used to select the processor which is having more features than any other processor. The selection of the algorithm is based on the application.
Table 3.3: Results of Mapping after Extraction of the Applications.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Code Size (Bytes)</th>
<th>Execution Time (Ticks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philips</td>
<td>P87LPC778</td>
<td>112</td>
<td>1261</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89LV51</td>
<td>112</td>
<td>459</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89C511C2/D2</td>
<td>114</td>
<td>378</td>
</tr>
</tbody>
</table>

Table 3.4: Performance of the Selected Processors for LED Application.

3.6 Examples

The architecture of the system shown in Figure 3.5 that uses the following setup to achieve the results of WSA and KT:

*Host System Configuration:* Processor: Intel P4 with 2.8 GHz

- **OS:** Windows XP
- **RAM:** 256MB
- **Database:** 204 Processors
- **Software:** 1. Matlab7
  - 2. Keil IDE

**Application 1: Dancing lights application**

In this section we evaluate Hybrid Algorithm using the dancing lights application.

**Step1. Creation of the Application:** Keil IDE is used to create the dancing lights application as shown in Figure 3.6 and its characteristics are extracted.

**Step2. Mapping the characteristics to processor specifications:** The entire code side is less the 1KB and hence we considered flash as 1. The data memory is also less then 128
bytes and hence we considered as 128. We extracted timers as one (to introduce the delay) MIPS as 1. We have chosen flash and hence required ISP. Table 3.3 shows the results of the mapping.

**Step3. Specifying Musts (for KT method only) and Weights:** The designer is asked to specify the weights for both the processor selection algorithms. In this example, let us assume that the weights assigned are 0.9, 0.8, 0, 0.1, 0, 0.3, 0.2 and 0.7 for IRAM, XRAM, EEPROM, Buses, MIPS, Timers, UAST, DPTR, and ISP respectively. For the KT Method, Flash and I/O lines are considered as musts and the weights assigned to the rest of the parameters are 0.9, 0.8, 0, 0.1, 0, 1, 0.3, 0, 0.2 and 0.7 for IRAM, XRAM, EEPROM, Buses, MIPS, Timers, UAST, DPTR, and ISP respectively.

**Step4. Results of processor selection algorithms:** Figure 3.7 shows the results of the WSA and Table 3.5 show the results of the KT Method for the LED application.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Flash</th>
<th>IRAM</th>
<th>XRAM</th>
<th>EEPROM</th>
<th>Lines</th>
<th>Buses</th>
<th>MIPS</th>
<th>Timers</th>
<th>UART</th>
<th>DPTR</th>
<th>ISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel</td>
<td>AT89C51ID2</td>
<td>64</td>
<td>256</td>
<td>1792</td>
<td>2048</td>
<td>48</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT83/89C5132</td>
<td>64</td>
<td>256</td>
<td>2048</td>
<td>0</td>
<td>44</td>
<td>2</td>
<td>1.66</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89C51SND1</td>
<td>64</td>
<td>256</td>
<td>2048</td>
<td>0</td>
<td>44</td>
<td>2</td>
<td>1.66</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.5: Results of the KT Method for LED Application.

Figure 3.7 show the percentage of match of the specifications of the processors against the application generated ones. For example, all the selected processors match the required ISP value where as for timers the required is 1 but the selected processor 1 and 2 are having 3 and the third processor is having 3. XRAM and EEPROM are not required for the application and only third processor is not matched. Now the designer may arrive at a decision to choose one processor among the three.

**Step5. We run the application on the selected processors through Keil IDE and analyze the performance of these processes.**
Figure 3.5: Architecture of the System.

```c
void wait (void)
{

}

void main (void) {
unsigned int   i,j,k;  k=1000;
while (k) {
    /* Loop for 1000 times */
    for (j=0x01; j< 0x80; j<<=1) { /* Blink LED 0 to 6 of ports p1 and other*/
        P1=P0 = j; /* Output to LED Port */
        for (i = 0; i < 10000; i++) { /* Delay for 10000 Times */
            wait ( );  } /* call wait function */
        for (j=0x80; j> 0x01; j>>=1) { /* Same as above in reverse order
            P3 =P2 = j; /* Output to LED Port */
            for (i = 0; i < 10000; i++) { /* Delay for 10000 Counts */
                wait ( ); /* call wait function */
            }
        }
    }
    k=k-1;
}
```

Figure 3.6: LED Application.
All the three processors which were selected are studied while running the application under Keil IDE and the results are given in Table 3.4. It shows that the first processor took more time for running the application than the others but the code size is almost the same for all the processors. If time is not a constraint then the first processor can be chosen. So the Philips P87LPC778 can be considered because it is operated with less frequency and time is not a constraint in this application. And hence the first choice is meeting application requirements.

Table 3.6 depicts the deviation (error) in the parameters of the selected processors. This value is arrived at by using the weighted sum algorithm. That is, it is the difference between the weighted sum of the processor and the ideal processor. The first processor which has the maximum weight has matched 6 requirements out of 11. The second and third processor, even though do not exactly match the requirements are relatively very close to the requirements. We have shown the results of the KT method for the LED application in Table 3.5. We found from these results that the WSA and KT method have selected different processors for the given LED application. The KT method selects those processors which sometimes more than meet the requirements as the Musts have to be always satisfied. On the other hand, the WSA gives the best fit (just sufficient) processors for the given application. The results have shown that the processors, even with variable features of the on-chip components, have almost the same performance for all the candidate processors.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philips</td>
<td>P87LPC778</td>
<td>4.5</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89LV51</td>
<td>5.4</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89C511C2/D2</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3.6: Error between the Ideal Processor and Selected Processors.
Application 2: Display Name through Serial Port

Program to display the name through serial port is shown in Figure 3.8. Table 3.7 shows the selected processors by the WSA. The first processor matched is the Philips P87LPC778 followed by Atmel AT 87F51 and AT89C1051 respectively. The Table 3.7 shows the actual specifications of the selected processors for the application2 against the designer requirements.

```c
#include <REG52.H> /* special function register declarations */
#include <stdio.h> /* prototype declarations for I/O functions */
#ifdef MONITORS /* Debugging with Monitor-51 needs */
char code reserve [3] _at_ Ox23; /* space for serial interrupt if */
#endif /* Stop Exection with Serial Intr. */
void main (void) {
    unsigned int k;
    SCON = 0x50; /* SCON: mode 1, 8-bit UART, enable rcvr */
    TMOD = 0x20; /* TMOD: timer 1, mode 2, 8-bit reload */
    TH1 = 221; /* TH1: reload value for 1200 baud @ 16MHz */
    TR1 = 1; /* TR1: timer 1 run */
    TI = 1; /* TI: set TI to send first char of UART */
    #endif
    K=10;
    while (1) { P1 ^= 0x01; /* Toggle P1.0 each time we print */
        printf("Hello World\n"); /* Print "Hello World" */
        k=k-1; }}
```

Figure 3.8: Display Name through Serial Port (Application2).
These three processors are analyzed with the Keil IDE by running the Application2. All three processors are producing the same code size and the first processor is taking the less time than the others shown as in Table 3.8

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Flash</th>
<th>IRAM</th>
<th>XRAM</th>
<th>EEPROM</th>
<th>Lines</th>
<th>Busses</th>
<th>MIPS</th>
<th>Timers</th>
<th>UART</th>
<th>DPTT</th>
<th>ISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td></td>
<td>1024</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Philips</td>
<td>P87LPC778</td>
<td>1024</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>1</td>
<td>3.33</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT87F51</td>
<td>1024</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89C51051</td>
<td>1024</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.7: Results of the Weighted Sum Algo for Application2.

Table 3.8: Performance of the Processors for Application2.

3.7 Comparison of KT and WSA

Table 3.9 depicts the various characteristics of these algorithms used for the processor selection along with its merits and demerits.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>KT Method</th>
<th>Weighted Sum Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of Use</td>
<td>Two Step Process</td>
<td>One Step Process</td>
</tr>
<tr>
<td>Requirements</td>
<td>Specified as Wants and Musts</td>
<td>Specify as Wants</td>
</tr>
<tr>
<td>Restrictions Use</td>
<td>Mandatory to satisfy the specifications of the processor while selection</td>
<td>No such restriction</td>
</tr>
<tr>
<td>Drawbacks/Adventages</td>
<td>It always selects high-end processor because musts must be satisfied.</td>
<td>It always gives the best-fit processor for the given requirements.</td>
</tr>
<tr>
<td>Applicability</td>
<td>Best suitable for the real time applications</td>
<td>Suitable for non real time applications</td>
</tr>
</tbody>
</table>

Table 3.9: Comparison of KT and WSA.
3.8 Summary

The performance of the embedded system depends mostly on the processor which is common for any embedded system. Each processor is characterized by a set of parameters and there are almost infinite alternatives available for the designer to select from. In this framework, different algorithms are described to select the suitable processor for an application of embedded system. The approach described in this thesis takes into account the application characteristics and thus is going to choose the most suitable processor for that application. We have illustrated the approach for the LED application. We believe that the technique can be further extended from 16 bit to 64 bit processors of different architectures. Note that the processors that are available vary in terms of the size of the data bus, the architectures and different vendors. For example, we have 8 bit, 16 bit, 32 bit and 64 bit processors with different architectures such as RISC or CISC or DSP developed by different vendors like ARM, Free Scale, Intel, TI etc. A change in the processor architecture leads to a change in the specifications of the processors. For example, the number of functional units, number of buses, the number of DMA channels etc. may change from DSP architectures to CISC architecture, while some parameters may remain unchanged. Hence, the more the architectures of the processors to be accounted for, the more will be the number of parameters that will have to be considered in the processor selection algorithm. Additionally, the size of the database will grow. However, this will not affect the choice of the right processor for a given application. It may be recalled that in our system we use normalization and assign weights in such a way that the processor, which is the best fit is chosen. Therefore, our technique can be easily extended to processors with different architectures.