LANNUZEL Timothé
e21530160
M2 LSE   2020/2021

Optimization of energy consumption with times constraints

Reports

15 February 2021

Université de Bretagne Occidentale

Table of contents

I. Introduction

Energy consumptions is an issue that affects every embedded system. Some solution has already been found like the use of Dynamic Voltage Scaling (DVS). However, due to DVS change the frequency of the processor, leading create issues for Real-Time systems.

This report explains the work done during the project that aims to reduce energy consumption with time constraints systems. For the first part, we explain briefly the state of art to the optimization of energy in a Real-Time system. In the second part, we describe in detail some algorithm that proposes to reduce energy consumption. Finally, in the third part, we talk about the implementation of those algorithms explained in Cheddar and their results.

II. The state of art

In this part, we explain what has been already found in the optimisation of energy consumption, and aim to find which papers can be implemented in Cheddar.

We mostly search scientific papers that aim to reduce the energy consumption of real-time systems All the papers read can be found in the reference part [ref 1-11]

Among the scientific papers, several approaches have been found to reduce energy consumption. One of the main common approaches is the use of Dynamic Voltage Scaling specifically for real-time systems. This means that the DVS algorithm considers tasks deadline and period.

In the end, we decided to implement a paper called: “Real-time dynamic voltage scaling for low-power embedded operating systems”, by Padmanabhan Pillal et Kang G. Shin [ref: 1]. It suggests static and dynamic voltage scaling solutions to reduce the energy consumption of real-time systems. One of the main reason this paper was chosen, is because it looks like it was this paper that introduces DVS solutions for real-time systems.

III. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems

In this part, we explain how Padmanabhan Pillal et Kang G. Shin achieves to reduce energy consumption in a Real-Time system.

The authors suggest solutions only for the EDF and the RM scheduler. In total, there are 5 algorithms of 3 different types:

The first two algorithms work with an EDF and RM scheduler, and is called “Static Voltage Scaling”. Static means that the frequency chooses by the algorithm will stay the same until a modification in the tasks set.

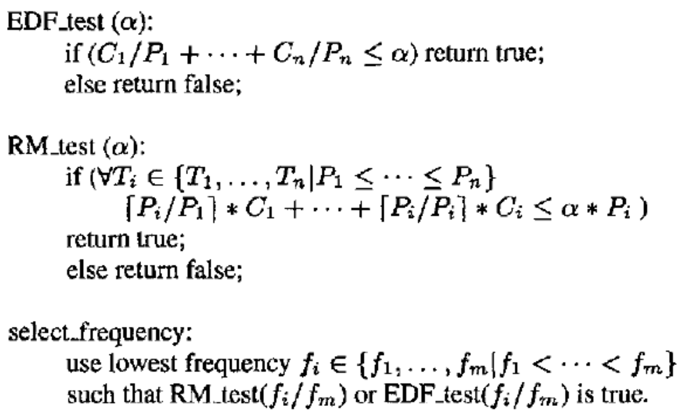
The next two algorithms are called “Cycle-conserving DVS”. By contrast, to the “Static Voltage Scaling” algorithm, “Cycle-conserving DVS” is dynamic which means during the task execution, the frequency can change at any moment to provide a better energy consumption. In addition, like “Static Voltage Scaling” there is one algorithm for EDF and RM schedulers.

The final algorithm is called “Look-Ahead RT-DVS” and like “Cycle-conserving DVS” it is a dynamic algorithm. However, unlike the two previous types of algorithm. It only works with an EDF scheduler.

• Static voltage scaling

Both Static Voltage Scaling algorithms work pretty much the same, the only main difference is that one work with the EDF scheduler the other one work with the RM scheduler.

Static Voltage scaling provides a very simple method for selecting a new frequency while maintaining real-time schedulability. Indeed the method only calculates a schedulability test of the task set. The result of this schedulability test is then compared with a set of given frequency.



```
EDF_test ($\alpha$):
    if ($C_1/P_1 + \cdots + C_n/P_n \leq \alpha$) return true;
    else return false;

RM_test ($\alpha$):
    if ($\forall T_i \in \{T_1, \ldots, T_n | P_1 \leq \cdots \leq P_n\}$
        $\lceil P_i/P_1 \rceil * C_1 + \cdots + \lceil P_i/P_i \rceil * C_i \leq \alpha * P_i$ )
    return true;
    else return false;

select_frequency:
    use lowest frequency $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$
    such that RM_test($f_i/f_m$) or EDF_test($f_i/f_m$) is true.
```

Figure 1 Static Voltage algorithm algorithms for EDF and RM schedulers [ref 1]:

This solution is very simple to implement. However, due to being a static algorithm, energy saving is not optimal by any means.

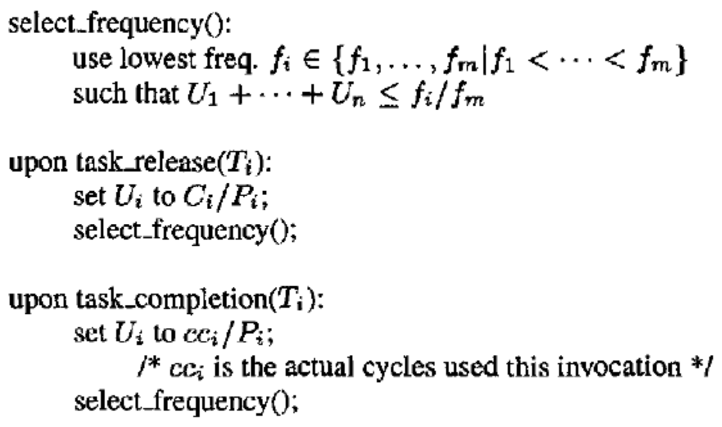
- ## Cycle-conserving DVS

As explain before Cycle-conserving DVS algorithms are dynamic algorithm.

Contrary to Static voltage scaling, algorithms for Cycle-conserving DVS for EDF and RM schedulers are very different. Therefore, we will explain each algorithm separately.

  - ### Algorithm for EDF

The EDF Cycle-conserving DVS algorithm resumes the schedulability test introduces in Static voltage scaling but add some tweaks. When a task is ready to be executed, the schedulability test of this task will use the worst computation time of the task. However, when a task is complete, the schedulability will use the actual computation time of the task. The authors of the paper suggest this solution because they note that tasks generally uses less than the worst-case specified.



```
select_frequency():
    use lowest freq. $f_i \in \{f_1, \dots, f_m | f_1 < \cdots < f_m\}$
    such that $U_1 + \cdots + U_n \leq f_i / f_m$

upon task_release($T_i$):
    set $U_i$ to $C_i / P_i$;
    select_frequency();

upon task_completion($T_i$):
    set $U_i$ to $cc_i / P_i$;
        /* $cc_i$ is the actual cycles used this invocation */
    select_frequency();
```

Figure 2 Cycle-conserving DVS algorithm for EDF scheduler [ref 1]

  - ### Algorithm for RM

For RM Cycle–conserving DVS, the authors of the papers decide to not follow the same approach they use in the EDF algorithm. Instead, the authors decide that each task should have two counters. The first one “c_left” serve to keep a track of the worst-case remaining cycles of computation need for the task. The second one “d” serve to know how much compute time the task will execute until the next deadline.

```
assume $f_j$ is frequency set by static scaling algorithm

select_frequency():
    set $s_m$ = max_cycles_until_next_deadline();
    use lowest freq. $f_i \in \{f_1, \dots, f_m | f_1 < \cdots < f_m\}$
    such that $(d_1 + \cdots + d_n)/s_m \leq f_i/f_m$

upon task_release($T_i$):
    set $c\_left_i = C_i$;
    set $s_m$ = max_cycles_until_next_deadline();
    set $s_j = s_m * f_j/f_m$;
    allocate_cycles ($s_j$);
    select_frequency();

upon task_completion($T_i$):
    set $c\_left_i = 0$;
    set $d_i = 0$;
    select_frequency();

during task_execution($T_i$):
    decrement $c\_left_i$ and $d_i$;

allocate_cycles(k):
    for $i = 1$ to $n$,   $T_i \in \{T_1, \dots, T_n | P_1 \leq \cdots \leq P_n\}$
                              /* tasks sorted by period */
        if ( $c\_left_i < k$ )
            set $d_i = c\_left_i$;
            set $k = k - c\_left_i$;
        else
            set $d_i = k$;
            set $k = 0$;
```

Figure 3 Cycle-conserving algorithm for RM scheduler [ref 1]

- ## Look-Ahead RT-DVS

Unlike Static voltage scaling and Cycle-conserving DVS, there is only one algorithm for Look-Ahead RT-DVS. This algorithm only works on the EDF scheduler. The objective of Look-Ahead RT-DVS algorithms is to defer all task beyond the earliest deadline in the systems. The main part of the algorithm is the "defer" procedure. "defer" calculates a frequency that depends on counters “c_left” and each task deadline.

```
select_frequency($x$):
    use lowest freq. $f_i \in \{f_1, \dots, f_m | f_1 < \cdots < f_m\}$
    such that $x \leq f_i/f_m$

upon task_release($T_i$):
    set $c\_left_i = C_i$;
    defer();

upon task_completion($T_i$):
    set $c\_left_i = 0$;
    defer();

during task_execution($T_i$):
    decrement $c\_left_i$;

defer():
    set $U = C_1/P_1 + \cdots + C_n/P_n$;
    set $s = 0$;
    for $i = 1$ to $n$,   $T_i \in \{T_1, \dots, T_n | D_1 \geq \cdots \geq D_n\}$
                          /* Note: reverse EDF order of tasks */
        set $U = U - C_i/P_i$;
        set $x = \max(0, c\_left_i - (1 - U)(D_i - D_n))$;
        set $U = U + (c\_left_i - x)/(D_i - D_n)$;
        set $s = s + x$;
    select_frequency ($s/(D_n -$ current_time));
```

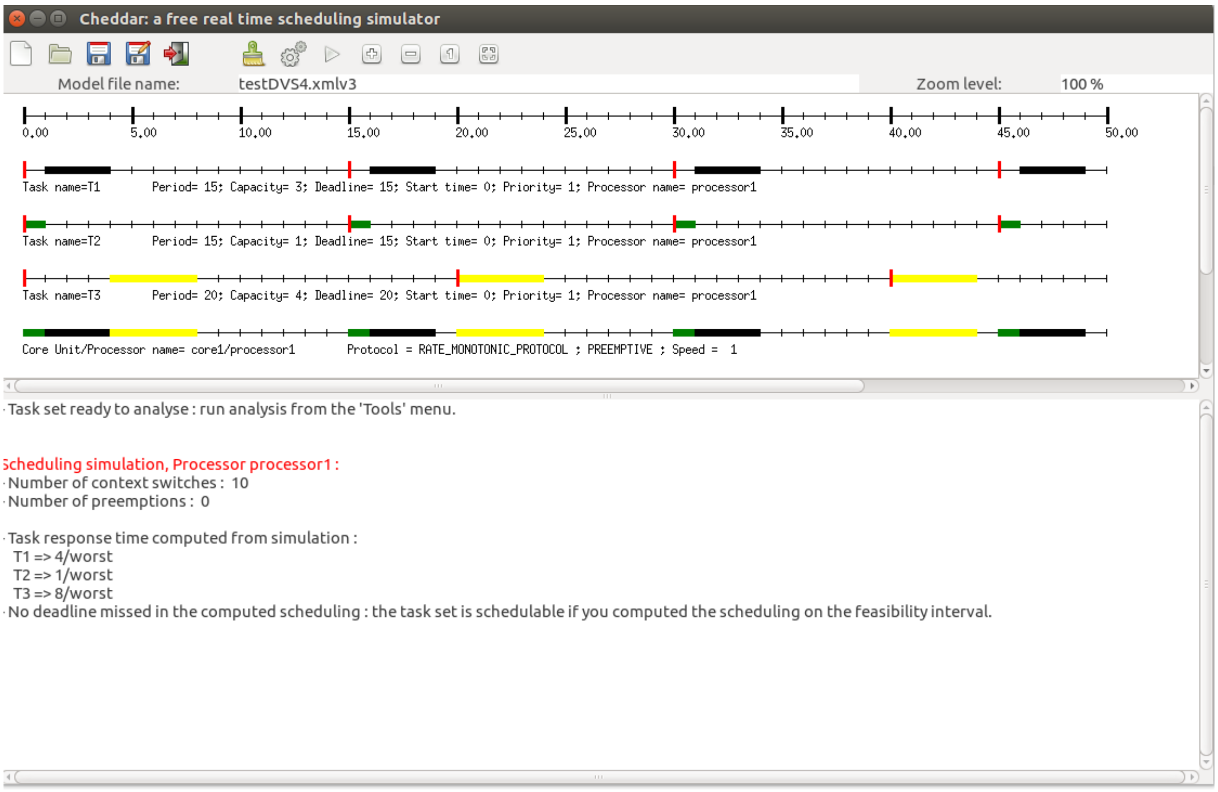
Figure 4 Look-Ahead RT-DVS algorithm [ref 1]

IV. Implementation in CHEDDAR

In this part, we explain the implementation of algorithms explains in the part earlier in Cheddar. We will also explain how to use this implementation. Moreover, we will show results obtain for each algorithm with a specific set of task.

• Cheddar

Cheddar is an open-source real-time scheduling tool/simulator that allows to modelise software architecture of real-time systems. This allows checking their schedulability or other performance criteria.


Cheddar: a free real time scheduling simulator
Model file name: testDVS4.xmlv3
Zoom level: 100 %
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
50.00

```
Task name=T1        Period= 15; Capacity= 3; Deadline= 15; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2        Period= 15; Capacity= 1; Deadline= 15; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3        Period= 20; Capacity= 4; Deadline= 20; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1        Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE ; Speed =  1
```

```
Task set ready to analyse : run analysis from the 'Tools' menu.

Scheduling simulation, Processor processor1 :
Number of context switches : 10
Number of preemptions : 0

Task response time computed from simulation :
  T1 => 4/worst
  T2 => 1/worst
  T3 => 8/worst
No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure 5: Screen capture of Cheddar

One of the main argument we use with Cheddar was the variable “speed” of the processor. The variable “speed” allowed to change the execution speed of a task. For example, if we increase the speed of the processor, tasks will take less time to compute, conversely, if we slow the speed, tasks will take more time to compute.

The use of the variable “speed” has some drawback. Indeed, in cheddar the type of "speed" is natural. It means that “speed” cannot be a float, which is usually the case when we talk about frequency. This is why it was decided to multiply the base speed, and all tasks computing time (capacity) by 100 to see the variable “speed” modification by RT-DVS algorithms.

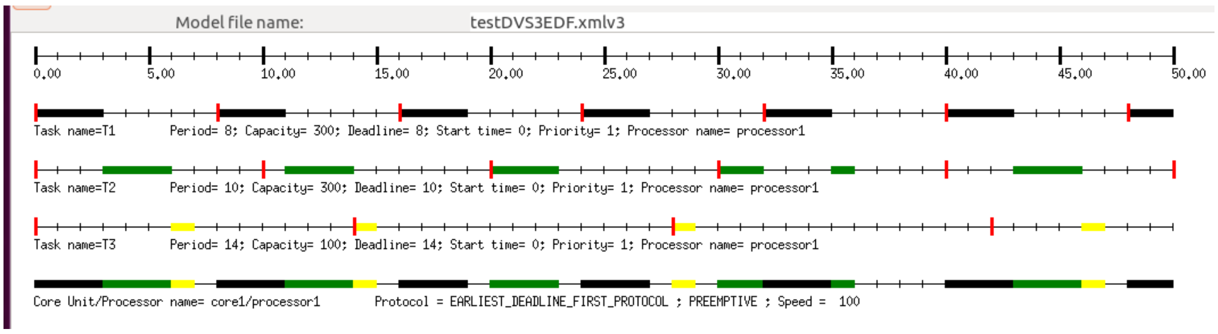
○ Task set

All the result show after will use the same set of tasks, which is as follow:

Task	Computing Time	Period
T1	30	8
T2	30	10
T3	100	14

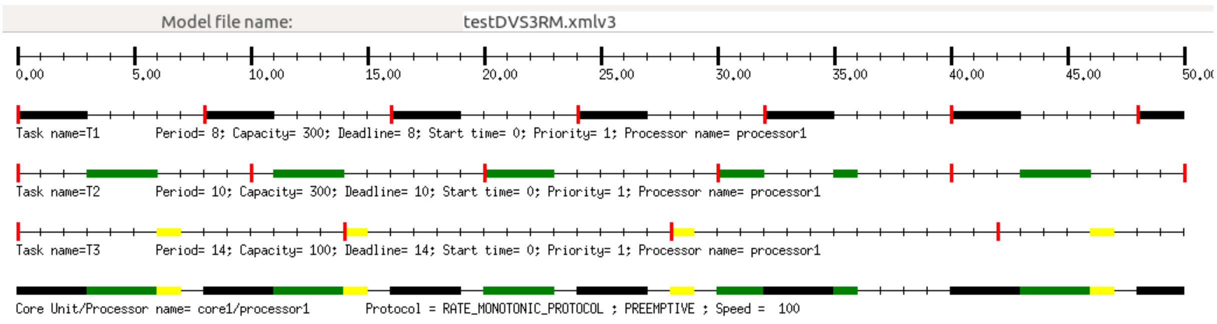
The entire tasks set will be compute by one processor with a frequency/speed between 25, 50, 75 and 100.

If we simulate the task set in cheddar with only a speed of 100 therefor without any attempt to change the energy consumption we obtain those results. [figure 6 and 7]



```
Model file name:                testDVS3EDF.xmlv3
0.00   5.00   10.00   15.00   20.00   25.00   30.00   35.00   40.00   45.00   50.00
Task name=T1     Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2     Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3     Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1       Protocol = EARLIEST_DEADLINE_FIRST_PROTOCOL ; PREEMPTIVE ; Speed =  100
```

Figure 6 : Cheddar simulation with EDF scheduler without any RT-DVS algorithms

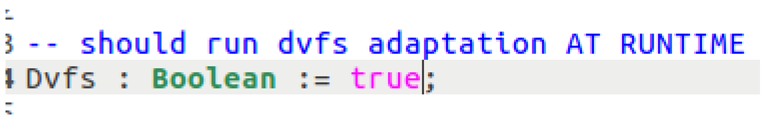


```
Model file name:                testDVS3RM.xmlv3
0.00   5.00   10.00   15.00   20.00   25.00   30.00   35.00   40.00   45.00   50.0
Task name=T1     Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2     Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3     Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1       Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE ; Speed =  100
```

Figure 7 : Cheddar simulation with RM scheduler without any RT-DVS algorithms

○ Requirement and setup of RT-DVS with Cheddar

To allow the use of DVS algorithms in cheddar, the user as to modify the file “/CHEDDAR/trunk/src/config/editor_config.ads", and change the Boolean variable “Dvfs” to true, just as the picture just below [figure 8].



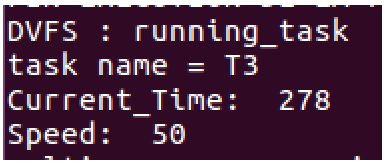
```
-- should run dvfs adaptation AT RUNTIME
Dvfs  : Boolean := true;
```

Figure 8: Screen capture of editor_config.ads

The user has to recompile cheddar after the modification of “editor_config.ads”.

Also as mentioned before the user, the user has to modify by 100 the base speed of the processor and task capacity.

One issue with the IHM of Cheddar is we can only see the last value of speed. This issue is not a problem for static algorithms but dynamic algorithms, it is. If the user wants to see the current at a current, it will have to look console log, just like the picture below. [figure 9]



```
DVFS : running_task
task name = T3
Current_Time:  278
Speed:  50
```

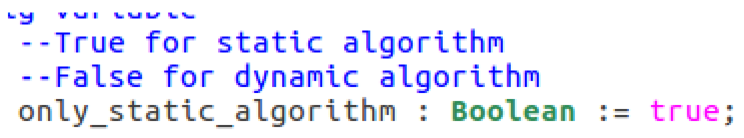
Figure 9 Screen capture of the console log of Cheddar

- ## Static voltage scaling

As said before with the Static Voltage Scaling algorithm, the frequency is change only one time, which is before any task execution

  - ### Setup

To be able to use Static Voltage Scaling algorithms, the user has to modify the file "/CHEDDAR/trunk/src/framework/voltage_scaling.ads", and change the Boolean variable "only_static_algorithm" to true, just as the picture just below. [figure 10]

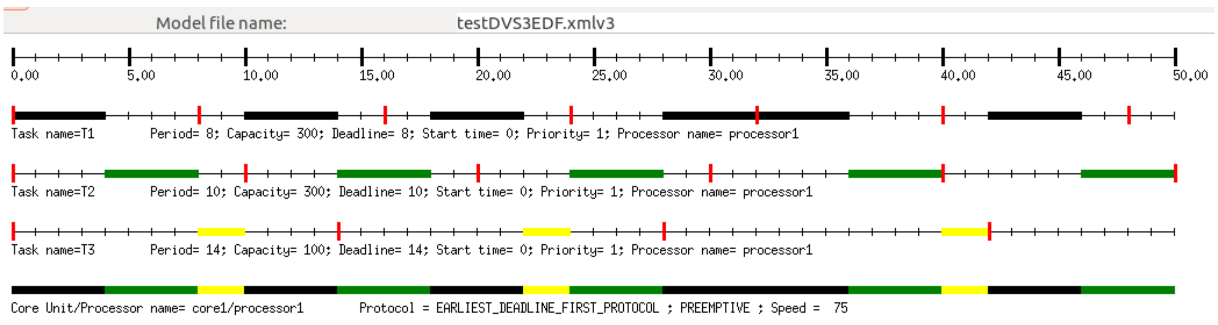


```
 --True for static algorithm
 --False for dynamic algorithm
 only_static_algorithm : Boolean := true;
```

Figure 10 Screen capture of voltage_scaling.ads for Static Voltage Scaling

Like before when the user modify a file you have to recompile cheddar.

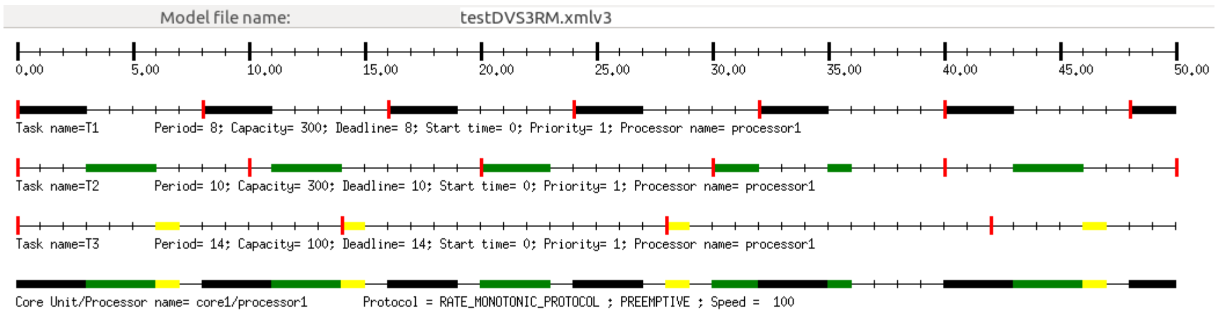
  - ### Results



```
Model file name:                    testDVS3EDF.xmlv3

Task name=T1      Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2      Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3      Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1      Protocol = EARLIEST_DEADLINE_FIRST_PROTOCOL ; PREEMPTIVE ; Speed =  75
```

Figure 11 : Cheddar simulation with EDF scheduler and Static voltage scaling algorithm



```
Model file name:                testDVS3RM.xmlv3

Task name=T1      Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2      Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3      Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1      Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE ; Speed =  100
```

Figure 12 : Cheddar simulation with RM scheduler and Static voltage scaling algorithm

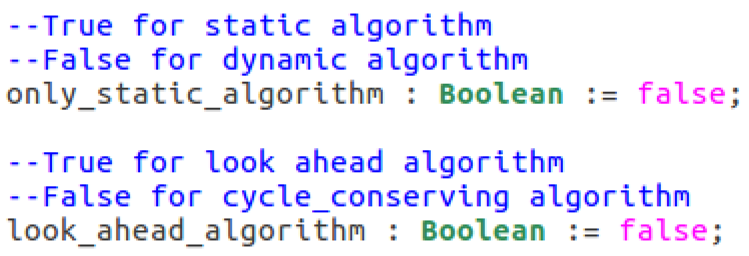
As shown in the result for the EDF scheduler [figure 10] the computation time of a task take more time to be fully executed. This due to the variable speed change to 75 because of the Static voltage scaling algorithm.

However, if the variable speed is changed to 75 for EDF, the speed stays at 1 for the RM scheduler as shown in the result [figure 11]. This is because the Static Voltage Scaling for RM scheduler is less aggressive compare to the EDF scheduler in energy saving.

- ## Cycle-conserving RT-DVS

  - ### Setup

To be able to use Cycle-conserving RT-DVS algorithms, the user has to modify the "*/CHEDDAR/trunk/src/framework/voltage_scaling.ads*", and change both Boolean variables "only_static_algorithm" and "look_ahead_algorithm" to false, just as the picture below. [figure 13]

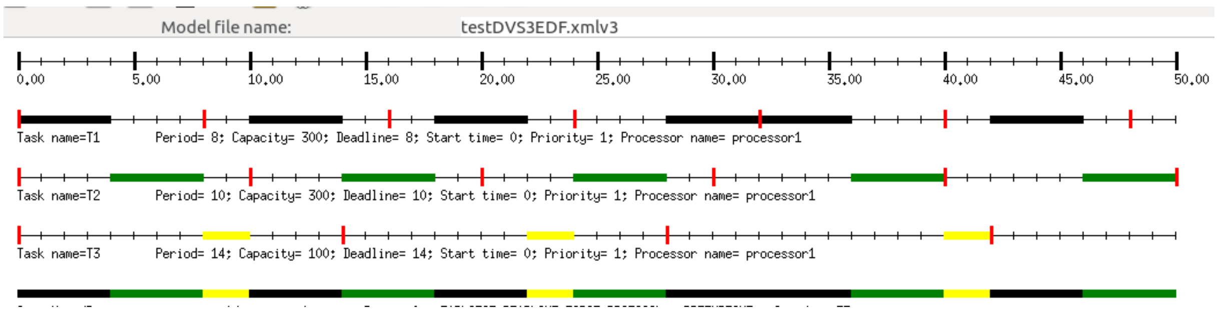


```
--True for static algorithm
--False for dynamic algorithm
only_static_algorithm : Boolean := false;

--True for look ahead algorithm
--False for cycle_conserving algorithm
look_ahead_algorithm : Boolean := false;
```

Figure 13 Screen capture of voltage_scaling.ads forCycle-conserving RT-DVS

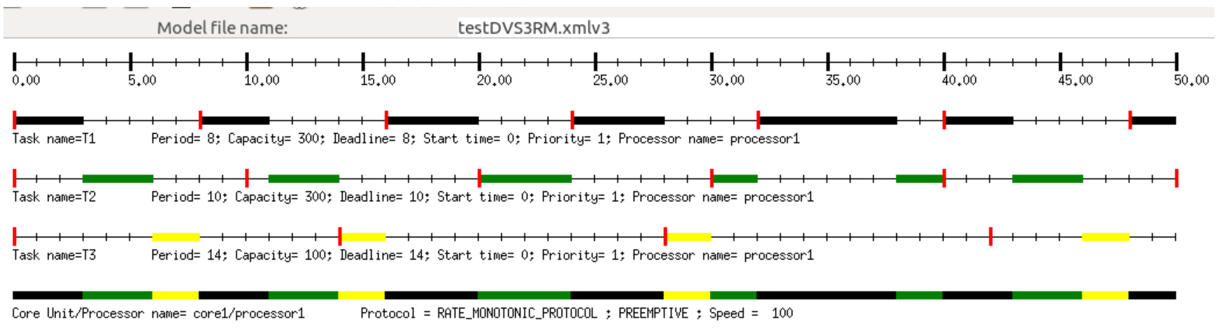
  - ### Results



```
Model file name:                testDVS3EDF.xmlv3

Task name=T1      Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2      Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3      Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
```

Figure 14 : Cheddar simulation with EDF scheduler and Static voltage scaling algorithm

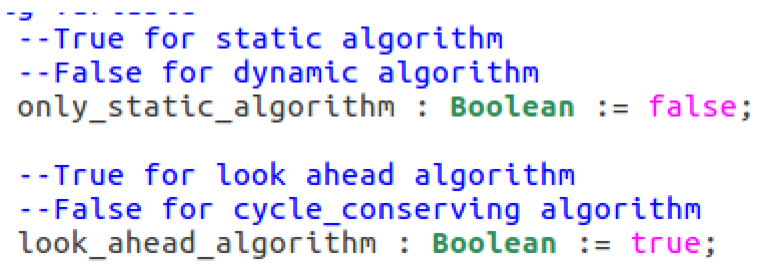

Model file name: testDVS3RM.xmlv3

```
Task name=T1        Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2        Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3        Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1        Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE ; Speed =  100
```

Figure 15 : Cheddar simulation with RM scheduler and Static voltage scaling algorithm

- ## Look-Ahead RT-DVS
  - ### Setup

To be able to use Cycle-conserving RT-DVS algorithms, the user has to modify the "*/CHEDDAR/trunk/src/framework/voltage_scaling.ads*", and change the Boolean variable "only_static_algorithm" to false and "look_ahead_algorithm" to true, just as the picture just below. [figure 16]

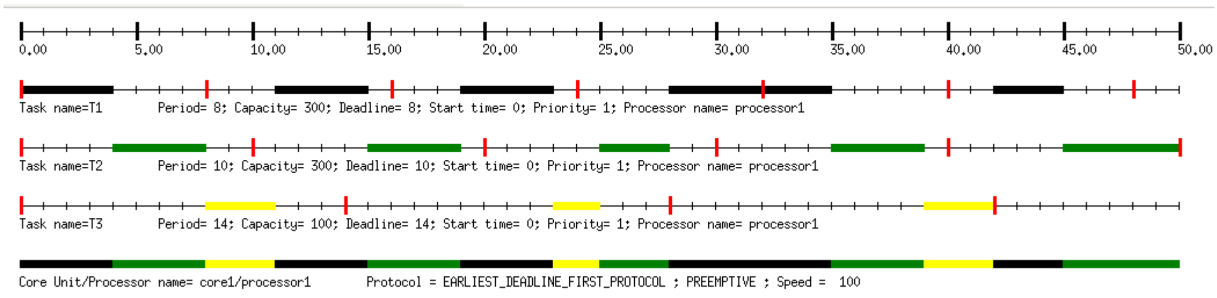


```
--True for static algorithm
--False for dynamic algorithm
only_static_algorithm : Boolean := false;

--True for look ahead algorithm
--False for cycle_conserving algorithm
look_ahead_algorithm : Boolean := true;
```

Figure 16 Screen capture of voltage_scaling.ads for look-Ahead RT-DVS

  - ### Result



```
Task name=T1        Period= 8; Capacity= 300; Deadline= 8; Start time= 0; Priority= 1; Processor name= processor1
Task name=T2        Period= 10; Capacity= 300; Deadline= 10; Start time= 0; Priority= 1; Processor name= processor1
Task name=T3        Period= 14; Capacity= 100; Deadline= 14; Start time= 0; Priority= 1; Processor name= processor1
Core Unit/Processor name= core1/processor1        Protocol = EARLIEST_DEADLINE_FIRST_PROTOCOL ; PREEMPTIVE ; Speed =  100
```

Figure 17 : Cheddar simulation with EDF scheduler and Look-Ahead RT-DVS

• Possible improvements and limits

Of course, some improvements can be made.

As an example, one improvement would be that the user no longer needs to modify the file “/CHEDDAR/trunk/src/framework/voltage_scaling.ads”, but instead indicate what type of RT-DVS algorithm directly in the XML file that contains the task set. Therefore, they will be no need to recompile Cheddar every time we want to try another RT-DVS algorithm.
Another improvement would be to use other scientific papers that give enhancement to the RT-DVS at energy saving.

One limit is we can just use EDF and RM schedulers with this implementation, other scheduler are not handle.

Another limit is we can only use periodic task. If the user tries to use an aperiodic task with this implementation, the program will crash.

V. Conclusion

To sum up, it possible to reduce the energy consumption of a Real-Time systems with deadline guarantee.

We decide to implement the work propose in the paper “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems” by Padmanabhan Pillal et Kang G. Shin in Cheddar.

Cheddar can now be used to simulate the schedulability test, while trying to reduce the energy consumption of Real-Time Systems with the used of RT-DVs algorithms.

VI. References

1. Padmanabhan Pillai and Kang G. Shin. 2001. Real-time dynamic voltage scaling for low-power embedded operating systems. In Proceedings of the eighteenth ACM symposium on Operating systems principles (SOSP '01). Association for Computing Machinery, New York, NY, USA, 89–102.
2. Gang Quan and Xiaobo Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232), Las Vegas, NV, USA, 2001, pp. 828-833, doi: 10.1109/DAC.2001.156251. Youngsoo Shin and Kiyoung Choi, "Power conscious fixed priority scheduling for hard real-time systems," Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361), New Orleans, LA, USA, 1999
3. Youngsoo Shin and Kiyoung Choi, "Power conscious fixed priority scheduling for hard real-time systems," Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361), New Orleans, LA, USA, 1999, pp. 134-139, doi: 10.1109/DAC.1999.781298.
4. Gang Quan and Xiaobo Sharon Hu, "Minimal energy fixed-priority scheduling for variable voltage processors," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 8, pp. 1062-1071, Aug. 2003, doi: 10.1109/TCAD.2003.814948.
5. Jiong Luo and N. K. Jha, "Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems," IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140), San Jose, CA, USA, 2000, pp. 357-364, doi: 10.1109/ICCAD.2000.896498.
6. I. Hong, D. Kirovski, Gang Qu, M. Potkonjak and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 12, pp. 1702-1714, Dec. 1999, doi: 10.1109/43.811318.
7. T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379), Monterey, CA, USA, 1998, pp. 197-202, doi: 10.1145/280756.280894.
8. F. Yao, A. Demers and S. Shenker, "A scheduling model for reduced CPU energy," Proceedings of IEEE 36th Annual Foundations of Computer Science, Milwaukee, WI, USA, 1995, pp. 374-382, doi: 10.1109/SFCS.1995.492493.
9. Weiser M., Welch B., Demers A., Shenker S. (1994) Scheduling for Reduced CPU Energy. In: Imielinski T., Korth H.F. (eds) Mobile Computing. The Kluwer International Series in Engineering and Computer Science, vol 353. Springer, Boston, MA
10. Youngsoo Shin, Kiyoung Choi and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140), San Jose, CA, USA, 2000, pp. 365-368, doi: 10.1109/ICCAD.2000.896499.
11. J. Pouwelse, K. Langendoen and H. Sips, "Energy priority scheduling for variable voltage processors," ISLPED'01: Proceedings of the 2001 International Symposium on Low Power Electronics and Design (IEEE Cat. No.01TH8581), Huntington Beach, CA, USA, 2001, pp. 28-33, doi: 10.1109/LPE.2001.945367.
12. F. Singhoff, J. Legrand, L. Nana, and L. Marcé. 2004. Cheddar: a flexible real time scheduling framework. In Proceedings of the 2004 annual ACM SIGAda international conference on Ada: The engineering of correct and reliable software for real-time &

distributed systems using Ada and related technologies (SIGAda '04). Association for Computing Machinery, New York, NY, USA, 1–8

13. Cheddar - http://beru.univ-brest.fr/cheddar/
14. Ada reference manual - http://stephe-leake.org/ada/arm.html