Analyzing Schedulability of Energy-oriented Distributed Real-time Embedded Software

Liqiong Chen[1], Guisheng Fan[1,2] and Yunxiang Liu[1+]
[1]Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China
Email: lqchen@sit.edu.cn
[2]Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China
[3] Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai, China
Email: gsfan@ecust.edu.cn, yxliu@sit.edu.cn
Corresponding author : yxliu@sit.edu.cn

***Abstract*—As computer systems become increasingly inter-networked, most of critical systems are distributed real-time embedded (*DRE*) system. A challenging problem faced by researchers and developers of *DRE* system is devising and implementing an effective method that can analyze requirements in varying operational conditions. In this paper, we analyze the requirements of DRE software and construct the corresponding energy consumption model, which is divided into fork module and leaf module based on its characteristics, and an energy consumption schema with time constrains is proposed for *DRE* software. The concept of critical task is presented according to the different position of task in the module, then constructing divide task set for module based on the characteristics of module and its critical task's position, the idle time allocation strategy and DVS adjustment method of sub task set are also advanced. Finally, a specific example is given to simulate the analysis process, the results show that the method can be a good solution to analyze *DRE* software.**

***Index Terms*—Distributed real-time and embedded system; Petri net; energy consumption; DVS; critical task**

I. INTRODUCTION

Distributed real-time embedded (*DRE*) systems are becoming increasingly widespread and important. Most of critical application is embedded systems that control physical, biological, or defense processes and devices [1]. For example, a typical networked DRE system will consist of multiple subsystems, which may involve combinations of both local and distributed deployment. In this environment, developers require an effective method that can help identify requirements and design defects before a commitment is made to a particular design strategy. In these early development phases, the cost effectiveness and ease of use of validation tools is significant, as well as the level of rigor supplied by the modeling language and environment. Besides meeting timing constraints, energy consumption has become a major consideration in DRE design. Even in an energy-rich platform, energy consumption has raised a serious of concerns with respect to reliability and cost [2].

Power consumption is one of the critical design considerations for embedded systems. Reducing power consumption can extend battery lifetime of portable systems, decrease chip cooling costs, as well as increase system reliability. Dynamic voltage scaling (DVS)[3] is a popular technique for reducing energy consumption, especially in DRE systems where each component could take hundreds of cycles to execute. Based on the DVS technique, energy management schemes in real-time application have been extensively explored. Fewer works, however, have focused on energy management for DRE systems. Therefore, energy consumption has become a hot research in DRE software not only to reduce energy consumption but also to reduce associated cost.

To address the schedulability of energy-oriented DRE systems, this paper makes three main contributions to the state of the art in DRE. First, it uses formal techniques in an accessible and cost-effective manner to support optimizing energy consumption of DRE systems. The approach is based on Petri nets, an established formal method which has been widely used to model and analyze concurrent and distributed systems. Second, we extend for Place Timed Petri net and propose an Hierarchical Distributed Real-time Embedded net (HDRE-net) model. A optimizing energy consumption schema by using DVS capability is advanced. Third, We propose the concept of critical task and divide task set for module based on critical task. The DVS adjustment method of task set and optimizing energy consumption steps of whole application are advanced, and its enforcement algorithm is also given. Finally, we explain the effectiveness and feasibility of method by using ELC sub system.

II. COMPUTATION MODEL

A. Definition of HDRE-net

Timed Petri nets (TPN) is a mathematical formalism, which allows to model for the features present in most concurrent and real-time systems[4], such as concurrent, asynchronism and distribution, etc. Some recent researches indicate that TPN is powerful enough to describe behavioral features of DRE software. The basic concepts of it can refer to [5]. In this paper, we extend for TPN and establish a model for analyzing DRE software.

doi:10.4304/jsw.6.9.1771-1778

Definition 1: A tuple $\Sigma=(TPN,I,\gamma,\mu)$ is called Distributed Real-time Embedded net (*DRE-net*) iff:

(1) $TPN=(PN, C, M_0)$ is a Timed Place Petri net;

(2) $I \subset P$ is a special place, which is called the interface of Σ and denoted by the dotted circle;

(3) γ is the priority function of transition. $\gamma(t_i)=(\alpha_i,\beta_i)$, where α_i,β_i are called the primary and secondary priority of transition t_i;

(4) μ: $T \rightarrow N^*$ is the unit energy consumption of transition, the default value is 0.

The distribution of token in each place at time θ is called the marking of DRE-net model, denoted by *M*. The marking $M(p)$ denotes the number of tokens in the place *p*. $M=M^a \cup M^u$, where M^a is the available tokens of *M*, M^u is the unavailable tokens of *M*. For any $x \in (P \cup T)$, we denote the pre-set of *x* as ${}^\bullet x=\{y | y \in (P \cup T) \wedge (y,x) \in F\}$ and the post-set of *x* as $x^\bullet=\{y | y \in (P \cup T) \wedge (x,y) \in F\}$. $\lambda(t_i)=(\alpha_i^S,\beta_i^S)$ is called the initial priority of transition t_i.

Definition 2: A six tuple $\Omega=(\Sigma,\Gamma,TI,TA,PI,PA)$ is called Hierarchical Distributed Real-time Embedded Net (HDRE-net) model, where:

(1) Σ is a *DRE-net* model, which describes the basic structure of Ω;

(2) $\Gamma=\{\Gamma_i | i \in Z^*\}$ is the finite set of *DRE-net* and HDRE-net, each element is called a page of Ω;

(3) $TI \subset T$ is the set of substituted operation, each page of HDRE-net corresponds to a substituted node and denoted by the double rectangle;

(4) *TA* is the page allocation function, whose function is to allocate the page to the substituted node;

(5) $PI \subset P$ is the set of interface node, which describes the input and output of substituted node, and denoted by double circle;

(6) *PA* is the mapping function of interface, which maps the interface node into the input and output of the operation.

From the definition, we can get that *DRE-net* is a special case of *HDRE-net*, that is, Γ of *HDRE-net* model is empty. We will analyze the operation mechanism of *HDRE-net* model in the following.

B. Operation mechanism of HDRE-net

Because the tokens in *HDRE-net* model include time factor, therefore, we will introduce the concept of wait time in this paper.

Definition 3: Let Ω be a *HDRE-net* model, which reaches marking *M* at time θ, $\forall P_i \in P$, place P_i has *j* tokens in marking *M*, P_i^k is the *k*th token of P_i. Vector : $TS(P_i)=(TS_i^1,TS_i^2,\ldots\ldots,TS_i^j)$ is the wait time of place P_i, where $TS(P_i^k)= max\{c_i-(\theta-\xi_k),0\}$, $TS(P_i)$ and $TS(P_i^k)$ are the wait time of P_i and P_i^k

$TS(P_i^k)=m$ explains the model must wait *M* time units before using token P_i^k. While $TS(P_i^k)=0$ represents the token is available. Recorded $TS(M,\theta)$ as the wait time set of places under marking *M*. A triple $S = (M, TE, TR)$ is called a state of Ω at time θ. Where *M* is marking, which describes the distribution of resources; $TS(M, \theta)$ is the time stamp of marking *M*, which depicts time properties of system.; *TE* is the energy consumption of reaching state *S*. Initial state $S_0=(M_0, TS_0, TE_0)$, where TS_0 is a zero vector, that is, all tokens are available in the initial state, $TE_0 = 0$, which means the energy consumption of initial state is 0.

Definition 4: Let Ω be a *HDRE-net* model, *S* is a state of Ω at time θ, for transition $t_i \in T$, iff:

(1) $\forall p_j \in P: p_j \in {}^\bullet t_i \rightarrow M^a(p_j) \geq W(p_j,t_i)$, then transition t_i is strong enabled under marking *S*, denoted by $S[t_i>$, all strong enabled transitions under state *S* are denoted by set *SET(S)*.

(2) $\forall p_j \in P$: $p_j \in {}^\bullet t_i \rightarrow M(p_j) \geq F(p_j,t_i) \wedge M^a < F(p_j, t_i)$, then transition t_i is weak enabled under state *S*, denoted by $S[t_i\geq$, all weak enabled transitions under state S are denoted by set *WET(S)*.

The set $ET(S)=SET(S) \cup WET(S)$. If transition t_i has weak enabled under state *S* and at least pass through ω time units to be strong enabled, then ω is called firing delay of transition t_i under state *S*, denoted by $FD(S,t_i)$ From the definition, we can get that the firing delay of strong enabled transition is 0.

Definition 5: Let Ω be a *HDRE-net* model, *S* is a state of Ω at time θ, $\forall t_i \in ET(S), \omega \in N^*$, the firing of transition t_i is effective iff it meets one of the following conditions:

(1) $t_i \in SET(S)$: $\alpha_i \leq min(\alpha_j) \wedge \beta_i \leq min(\beta_k)$, where $t_j \in SET(S)$, $t_k \in U(t_i)$

(2) $t_i \in WET(S)$: $SET(M)=\Phi \wedge FD(S,t_i) \leq \min((FD(S,t_j))$, $t_j \in WET(S)$

The set $U(t_i)=\{t_k | t_k \in SET(S) \wedge \alpha_k=\alpha_i\}$. All the effective firing transitions under state *S* are denoted by $FT(S)$.

Definition 6: Let Ω be a *HDRE-net* model, *S* is a state of Ω at time θ, the model will reach a new state *S'* by effectively firing enabled transition t_i at time $\theta+\omega$, denoted by $S[(t_i,\omega)>S'$, S' is called the reachable state of *S*, the computation of *M'* , *TS'*, *TE*' are based on the following rules:

(1) Computing marking:

$\forall P_j \in {}^\bullet t_i \cup t_i^\bullet, M'(P_j)=M(P_j)- W(P_j,t_i)+W(t_i,P_j)$

(2) Computing wait time:

First, adding wait time to the new generated marking:

$TS'(P_i^k)=c_i$, P_i^k is generated when firing transition t_i;

Second, modifying the wait time of tokens which are generated before the firing of transition t_i:

$TS'(P_i^k)=max\{(TS(P_i^k)-\omega),0\}, TS(P_i^k) \geq 0$.

(3) Computing energy consumption:

$TE_0=TE'+FD(S, t_i) \times \mu_i$

III. MODELING DRE SOFTWARE

A. Requirements of DRE software

DRE software can be regarded as a number of modules; each module also contains a number of partially ordered, serial or parallel implemented sub tasks [9, 10]. The function of *DRE* systems will be distributed to a number of interrelated embedded devices, each device is responsible for certain functions, and has certain autonomy, but relies on the computation of other embedded devices. Among them, *DRE* system has *n* tasks; each task is composed by a series of interrelated sub tasks set and a bus controller. The effective and

reliable communication between tasks is done by bus and bus controller. In this paper, we assume the communication between tasks is done by Controller Area Network (CAN).

As *DRE* software has strong performance requirements such as predictability, efficiency, reliability and security, et al. Therefore, it is necessary to consider above characteristics when describe the requirements of *DRE* software.

Definition 7: DRE software requirement model is a 9-tuple Ξ={*TK,NT,RS,RL,CP,RT,D, En, MPS*}:

(1) *TK, NT, RS* are the finite tasks set, module set and resource set, the *j*th task of module N_i is denoted by $TK_{i,j}$;

(2) *RL* is the relation between tasks, which may be sequence(>), choice(+), parallel(||) and exclusive(◊);

(3) CP:$TK\rightarrow(N^*\times N^*\times N^*)$ is the attribution function of task, which describes the running time and priority of task;

(4) RT:$TK\rightarrow RS^*$ is the resource function of task, whose function is to assign necessary resources to each task, RS^* represents the multiple set of resource, that is, a task can use multiple sources;

(5) *D* is the deadline of whole application;

(6)*En*: $TK\rightarrow N^*$ is the unit energy consumption of task, the unit energy consumption of task $TK_{i,j}$ is denoted by $e_{i,j}$;

(7) MPS:$N\rightarrow (N^*\times N^*)$ is the max and min supply voltage of module.

B. Modeling DRE software

The model of task $TK_{i,j}$ is shown in Fig.1, where place $p_{ac}^{i,j}$ describes the state of task $TK_{i,j}$, and its delay time is equal to the running time of task $TC_{i,j}$. While transition $t_{st}^{i,j}$, $t_{en}^{i,j}$ describe the beginning and termination operation of task, place $p_{pa}^{i,j}$, $p_{ou}^{i,j}$ describe the input and output parameters of task. And energy consumption of task $TK_{i,j}$ is regarded as firing energy consumption of transition $t_{en}^{i,j}$, that is $\mu(t_{en}^{i,j}) = e_{i,j}$. Transition $t_{ab}^{i,j}$, $t_{ac}^{i,j}$ represent the operation of module has overtime. We introduce place $p_{cn}^{i,j}$ to control the running process of task.

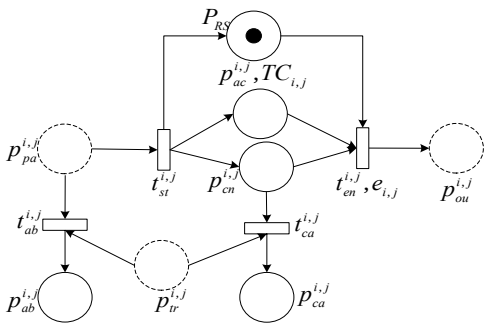

Fig. 1 HDRE-net Model of Task

Operator > represents the sequence relationship: If the firing of task $TK_{i,j}$ can lead to the firing of task $TK_{i,k}$, then the relationships between task $TK_{i,j}$ and $TK_{i,k}$ is sequence. $TK_{i,j}$ is the forward task of $TK_{i,k}$, while $TK_{i,k}$ is the afterward task of $TK_{i,j}$. The set $Forw(TK_{i,j})$, $Back(TK_{i,j})\subset TK$ are the forward and afterward task set of task $TK_{i,j}$. The *HDRE-net* model of $TK_{i,j}>TK_{i,k}$ is shown in Figure 2(a), the substituted node $TK_{i,j}$ and $TK_{i,k}$ corresponds to the page of task $TK_{i,j}$ and $TK_{i,k}$, while interface node $P^{i,j}{}_{pa}$, $P^{i,j}{}_{ou}$ represent the input and output of substituted node $TK_{i,j}$, which are mapped into the interface $p^{i,j}{}_{pa}$, $p^{i,j}{}_{ou}$ of task $TK_{i,j}$. Because the relationship between task and substituted node is one by one, the substituted node is also called task in the following. In order to describe the sequence relationship, we introduce transition t_{ou} to transfer the result of task $TK_{i,j}$ to the input interface of task $TK_{i,k}$.

We can construct the model of $TK_{i,j} + TK_{i,k}$, $TK_{i,j} \parallel TK_{i,k}$ and $TK_{i,j} \lozenge TK_{i,k}$ in the similar way, which are shown in Fig.2(b)-(d).

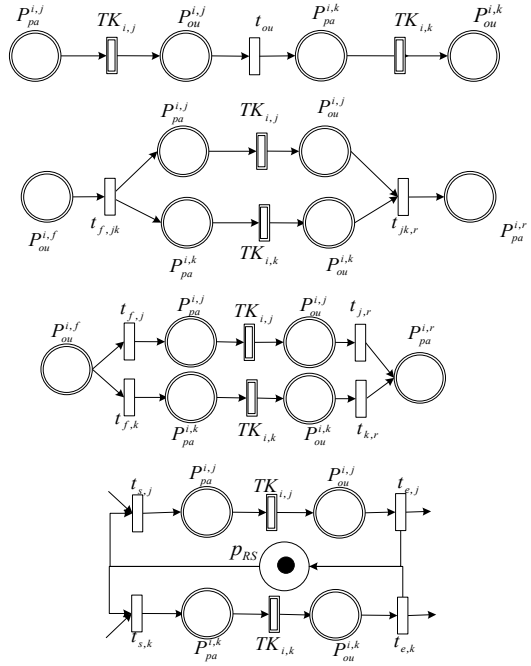

Fig. 2 HDRE-net Model of basic Relation

We will construct *HDRE-net* model of each module from bottom to up based on the relationships between task, as shown in Fig.3.

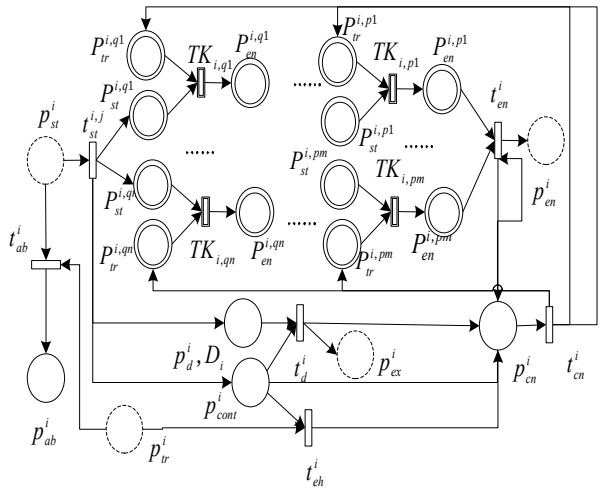

Fig. 3 HDRE-net Model of *Module* N_i

The operation process of module N_i is: the system will invoke the tasks in the module according to the relationship between task after initialing (t_{st}^{i}), and setting

${}^{\bullet}t_{st}^{\ i} = p_{st}^{\ i}$, $t_{st}^{\ i\bullet} = \{P_{pa}^{\ i,j} | \text{Forw}(TK_{i,j}) = \varnothing\}$; meanwhile, the local clock will begin to time, if all tasks can finish operating before the deadline D_i, then invoking termination operation ($t_{en}^{\ i}$) to make it be in the termination operation ($p_{en}^{\ i}$), and setting $t_{st}^{\ i\bullet} = p_{en}^{\ i}$, ${}^{\bullet}t_{en}^{\ i} = \{P_{ou}^{\ i,j} | \text{Back}(TK_{i,j}) = \varnothing\}$, otherwise, the system will invoke the overtime handling operation($t_d^{\ i}$) and do overtime handling for all tasks. The overtime handling of inner module is realized by introducing place $p_{cn}^{\ i}$ and transition $t_{cn}^{\ i}$, which make ${}^{\bullet}p_{cn}^{\ i} = t_d^{\ i}$, $p_{cn}^{\ i\bullet} = t_{cn}^{\ i}$, ${}^{\bullet}t_{cn}^{\ i} = p_{cn}^{\ i}$, $t_{cn}^{\ i\bullet} = \{P_{tr}^{\ i,1}, P_{tr}^{\ i,2}, \ldots, P_{tr}^{\ i,|NT|}\}$.

According to the communication principle of *CAN* bus, the communication process of task $TK_{i,j}$ sending message to task $TK_{g,f}$ is abstracted as a communication task $TK_{i,j\to g,f}$. The *HDRE-net* model of task $TK_{i,j\to g,f}$ is shown in Fig.4. Where place $p_{ou}^{\ i,j}$ and $p_{pa}^{\ g,f}$ are the input and output interface of task, while place p_b and p_h describe the idle buffer and bus token resource. The process of communication task $TK_{i,j\to g,f}$ is: getting data packet after beginning to operate($t_{st}^{\ i,j\to g,f}$), and being in waiting for idle buffer ($p_{gp}^{\ i,j\to g,f}$); the system will release buffer and bus token ($t_{en}^{\ i,j\to g,f}$) after finishing sending data packet. The energy consumption of task $TK_{i,j\to g,f}$ is regarded as firing energy consumption of transition $t_{en}^{\ i,j\to g,f}$, that is $\mu(t_{en}^{\ i,j\to g,f}) = e_{i,j\to g,f}$.

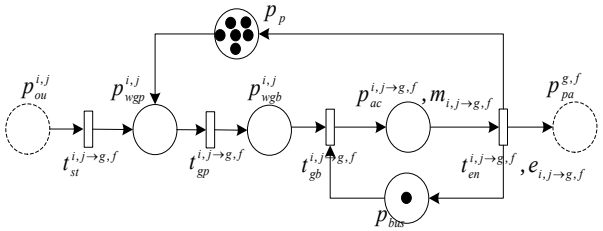

Fig. 4 HDRE-net Model of Communication Process

According to the requirement model and relationships between module, we can construct *HDRE-net* model of whole application from bottom to up, which is shown in Fig.5. Similarly, place p_{st}, P_{en} are introduced to describe the initial and termination state of whole application, transition p_{st}, p_{en} are introduced to describe the initial and termination operation of whole application. Place $P_{tr}^{\ 1}$, $P_{tr}^{\ 2}$, … , $P_{tr}^{\ |NT|}$ is the coordination input of module, interface node $P_{en}^{\ 1}$, $P_{en}^{\ 2}$, …, $P_{en}^{\ |N|}$ are the failure output of module. Transition $t_{e,1}$, $t_{e,2}$, …, $t_{e,|NT|}$ are used to transfer the overtime info of each module to the coordination place p_{cn}. If any one module has overtime, the system will invoke transition t_{cn} to do overtime handling for all modules.

Based on the constructed *HDRE-net* model, we will allocate priority to transition: in the *HDRE-net* model of task, the primary priority of transitions are equal to the priority of the corresponding task, while the priority of transition that introduced to describe process is (0,0), and the priority of substituted node is (0,0); the secondary priority of transition is divided into 5 grades according to its importance, the priority of transition $\beta^S(t_{en}^{\ i,j}) = 1$, $\beta^S(t_{ab}^{\ i,j}) = \beta^S(t_{ca}^{\ i,j}) = 0$, $\beta^S(t_c^{\ i,j}) = \beta^S(t_g^{\ i,j}) = 2$, $\beta^S(t_r^{\ i,j}) = 3$, $\beta^S(t_{st}^{\ i,j}) = 4$, $\beta^S(t_d^{\ i,j}) = 5$; while the secondary priority of transition $t_{ab}^{\ i,j}$ and $t_{ca}^{\ i,j}$ is the highest, which is used to describe overtime handling, for example, if the task is in the initial position, then aborting it, if the task is in the active position, then canceling it.

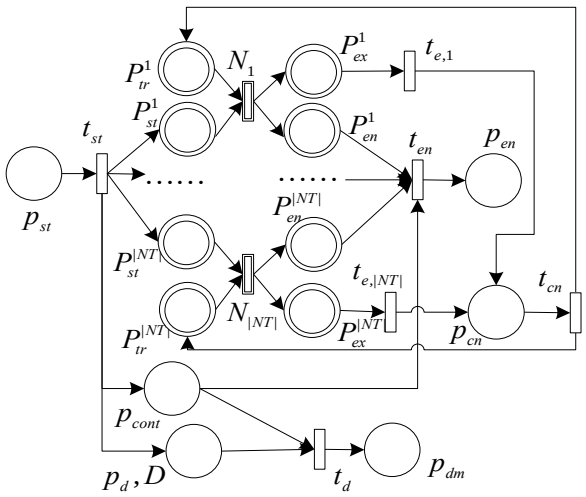

Fig. 5 HDRE-net Model of Whole Application

The primary priority of transitions in communication task $TK_{i,j\to g,f}$ are equal to the priority of recipient task, that is $TP_{g,f}$; the secondary priority of transition in task $TK_{i,j}$ is divided into 6 grades according to its importance, the priority of transition $t_{en}^{\ i,j}$, $t_c^{\ i,j}$, $t_g^{\ i,j}$, $t_r^{\ i,j}$, $t_{st}^{\ i,j}$, $t_d^{\ i,j}$ is incremental. The secondary priority of transitions in communication task $TK_{i,j\to g,f}$ are divided into 5 grades according to its importance, the priority of transition $t_{en}^{\ i,j\to g,f}$, $t_{gb}^{\ i,j\to g,f}$, $t_{gp}^{\ i,j\to g,f}$, $t_{st}^{\ i,j\to g,f}$, $t_d^{\ i,j}$ is incremental.

IV. OFFLINE DVS SCHEDULING OF DRE SOFTWARE

A. Diving task set for module

In this section, we will analyze the characteristics of task and module, then proposing the concept of communication task and critical task.

Let Ω be a *HRDE-net* model, $\forall TK_{i,j}, TK_{g,f} \in TK$, $i \neq g$, if there exists communication task $TK_{i,j\to g,f}$ between task $TK_{i,j}$ and $TK_{g,f}$, then task $TK_{i,j}$ is the communication forward task of $TK_{g,f}$, while $TK_{g,f}$ is the communication afterward task of $TK_{i,j}$, the communication forward and afterward task set are denoted by $ConF(TK_{i,j})$ and $ConB(TK_{i,j})$; meanwhile, module N_g is the directly afterward module of N_i, the set $Next(N_i)$ is denoted as the directly afterward module set of module N_i; if the inner tasks of module don't output parameters to other module, then the module is leaf module, otherwise it is the fork module; the module that need interact with external module (other modules) is called critical tasks; the moment that module receive the outputs of other module's task is called critical point.

We will divide the task set of leaf module N_i according to the critical point set of module. We may set task $TK_{i,j}$ and $TK_{i,k}$ of module N_i need the output of other module, and task $TK_{i,j}$ is the forward task of $TK_{i,k}$. Then the task set NT_i of module N_i is divided into three sub set: the task set before task $TK_{i,j}$ is denoted by $Fbeg(TK_{i,j})$, the total running time is $Start_{i,j}$; the task set in the middle of task

$TK_{i,j}$ and $TK_{i,k}$ is denoted by $Bet\{TK_{i,j}, TK_{i,k}\}$, and running time is $Start_{i,k}$ - $Start_{i,j}$; the task set after task $Start_{i,j}$ is denoted by $Bbeg(TK_{i,j})$, the total running time is D-$Start_{i,j}$.

Fork module is divided into communication forward task and non-communication forward task according to whether the fork module need the parameters. We will divide fork task set into two type of fork module in the following.

We will firstly model for the fork module of non-communication forward module. Let task $TK_{g,k}$ of module N_g need output parameters to the task $TK_{i,j}$ of module N_i, while task $TK_{g,r}$ need output parameters to the task $TK_{p,q}$ of module N_p. And the dynamic voltage of task in module N_i and N_p have finished adjusting, $TK_{g,k}$ is operated before the beginning of $TK_{g,r}$, we can divide task set NT_g of module N_g into three sub set, which is shown in Fig.6(a). The task set before the termination of task $TK_{g,k}$ is denoted by $Fend(TK_{g,k})$, and the total running time is $End_{g,k}$, where $End_{g,k}$ = $Start_{i,j}$ - $m_{g,k\to i,j}$; the task set between the termination of task $TK_{g,k}$ and $TK_{g,r}$ is denoted by $Fbet(TK_{g,k}, TK_{g,r})$, and running time is $End_{g,r}$ - $End_{g,k,}$ where $End_{g,r}$ = $Start_{p,q}$ - $m_{g,r\to p.q}$; the task set after the termination of task $TK_{g,r}$ is denoted by $Bbeg(TK_{i,j})$, and running time is D-$End_{g,k}$.

Second, we will model for fork module of communication forward module. The corresponding model is shown in Fig.6(b), we can divide task set NT_g of module N_g into three sub set: The task set before the termination of task $TK_{g,k}$ is $Fbeg(TK_{g,k})$, and total running time is $Start_{g,k}$, where $Start_{g,k}$ = $Start_{i,j}$ - $m_{g,k\to i,j}$; the task set between the termination of task $TK_{g,k}$ and $TK_{g,r}$ is $Fbet(TK_{g,k}, TK_{g,r})$, and running time is $End_{g,r}$ - $Start_{g,k}$, where $End_{g,r}$ = $Start_{p,q}$ - $m_{g,r\to p,q}$; the task set after the termination of task $TK_{g,r}$ is $Bbeg(TK_{i,j})$, and running time is D-$End_{g,k}$.

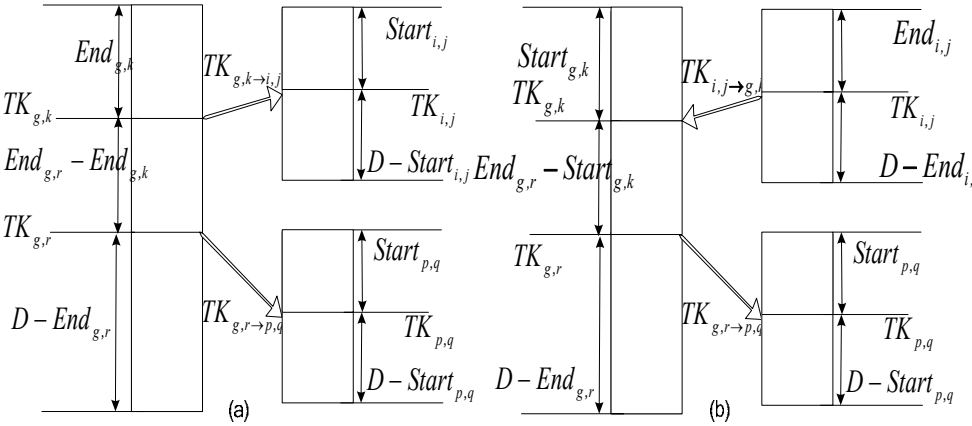

Fig. 6 Cross Module Task Set Division Example of Fork Module

B. DVS adjust for task set

We will allocate idle time to task according to its characteristics, let the deadline of all modules be D, the steps of allocating idle time are as following:

(1) Calculating running time of the longest path by combining time reachability graph, we may set it be TC_{max};

(2) Computing idle time: $D_{rem} = D - TC_{max}$;

(3) Adjusting running time of the longest path, assuming that the path is composed by task $TK_{i,q1}$, $TK_{i,q2}$, …, $TK_{i,qm}$, then:

$$TC_{i,j}{}' = TC_{i,j} + D_{rem} * e_{i,j} / \sum_{j=q1}^{qm} e_{i,j}$$

(4) Combining the process of diving task set, we can get running time of other tasks.

Let the total running time of task set $SubT = \{TK_{i,q1}, TK_{i,q2}, \ldots, TK_{i,qm}\} \subseteq NT_i$ be D_{SubT}, we will dynamically adjust supply voltage of task in $SubT$, the specific steps:

(1) Computing idle time: $D_{rem} = D_{SubT} - \sum_{j=q1}^{qm} TC_{i,j}$;

(2) Recomputing running time of each task: according to the relationship between unit energy consumption and voltage, we will allocate running time based on the proportion of energy consumption.

The adjusted running time of task $TK_{i,j}$ is:

$$TC_{i,j}{}' = TC_{i,j} + D_{rem} * e_{i,j} / \sum_{j=q1}^{qm} e_{i,j}$$

(3) Computing the adjusted probability of task's running time $T_{i,j}$: $ETC_{i,j} = \dfrac{TC_{i,j}}{TC_{i,j}{}'}$;

(4) Computing new supply voltage of task $TK_{i,j}$:

$$V_{dd}^{i,j}{}' = V_{\min}^{i} + \frac{V_0^i}{2ETC_{i,j}} + \sqrt{(V_{\min}^{i} + \frac{V_0^i}{2ETC_{i,j}})^2 - (V_{\min}^{i})^2}$$

where $V_0^i = \dfrac{(V_{\max}^{i} - V_{\min}^{i})^2}{V_{\max}^{i}}$;

(5) Computing new unit energy consumption of task $TK_{i,j}$: from the formula:

$$\frac{e_{V_{dd}}}{e_{V_{\max}}} = \frac{\alpha * C_L * f_{V_{dd}} * {V_{dd}}^2}{\alpha * C_L * f_{V_{\max}} * {V_{\max}}^2} = \frac{1}{ETC} * \frac{{V_{dd}}^2}{{V_{\max}}^2},$$

we can get new energy consumption of task $TK_{i,j}$ is:

$$e_{i,j}' = e_{i,j} * \frac{1}{ETC_{i,j}} * \frac{V_{dd}^{i,j}'}{V_{\max}^{i}};$$

(6) Computing the beginning and termination time of task $TK_{i,j}$:

$$Start_{i,j}' = \sum_{k=1}^{j-1} TC_{i,k}'$$

$$End_{i,j}' = Start_{i,j} + TC_{i,j}'$$

B. Optimizing energy consumption and enforcemen

Based on the state space of *HDRE-net* model, the energy consumption optimization of whole application is: (1) Constructing the state space of each module; (2) Dividing task set for each module; (3)Computing running time of task set; (4) Adjusting *DVS* for task set; (5) Computing energy consumption of whole application:

$$\sum_{i=1}^{|NT|} \sum_{j}^{|NT_i|} e_{i,j}' * TC_{i,j}'$$

For the bounded *HDRE-net*, because its reachability state set $R(S_0)$ is a finite set, therefore, $R(S_0)$ is viewed as a vertex set, and the direct reachability relationship between states is viewed as arc set, and the transition is marked in the corresponding arc, which constructs a directed arc. The directed graph is called Timed Reachable Graph in *HDRE-net* model. We can analyze state change, transition firing sequence and execution time of system by using Timed Reachable Graph, thus getting the related properties of *HDRE-net* model.

According to the construction algorithm of Petri net reachability graph, we can construct *HDRE-net* model. Based on the state space, we will use a heuristic algorithm energy consumption for *DRE* software. The algorithm will firstly compute running time of the longest path by combining with the requirement model and state space, then gradually optimizing energy consumption of other tasks, we will do following operation for module set *TNT*:

Step 1: Computing the largest leaf module set $NoB = NoB(TNT)$, then doing Step 2;

Step 2: If $NoB(TNT) \neq \varnothing$, then arbitrary choosing module $N_i \in NOB$, $NOB = NoB - N_i$, and doing Step 3, otherwise doing Step 6;

Step 3: Computing critical task set $Crut(N_i)$ of module N_i;

Step 4: If $Sub(NT_i) \neq \varnothing$, then arbitrary choosing task set $Sub_j \in Sub(NT_i)$, $Sub(NT_i) = Sub(NT_i) - Sub_j$; and doing Step 5, otherwise doing Step 2;

Step 5: Computing critical task's beginning time of Sub_j by combining with the path of state space, then computing running time of whole task set Sub_j, and dynamically adjusting each task in task set Sub_j according to *DVS* adjustment method of task set, then doing Step 4;

Step 6: Let $TNT = TNT - NOB(TNT)$, then doing next operation.

V. EXPERIMENTS

In order to better describe the above modeling process and explain the correctness of analysis process, we use an actual case - Electronic Toll Collection (*ETC*) as an example. *ETC* system is an advanced system which consists of high-tech equipment and software such as electronics technology, computer technology, communications and network technology, and can achieve the function of automatically charging the cost of road without stopping vehicle. Strictly speaking, *ETC* application is the typical *DRE* software.

The workflow of *ELC* system is: the system will display traffic light once starting to operate, then informing lane computer to send start-up instructions to antenna controller when trigger coil detects the passage of vehicles. The read information from *OBE* will send to data processing center for data processing and charging. If success, then charge information screen will display “charge successfully, the amount of consumption”; the system mainly adopt camera and capture on the spot, and traffic police department will do the corresponding handling. The general distance of charging is 30m. The design speed of *ETC* lanes is 40km/h, then handling time of whole applications is 0.09s.

According to the actual requirements, we can divide the whole application into four function modules. Module 1 responses for controlling auxiliary equipment, including traffic lights, display screen, lever and coil; Module 2 responses for reading *OBE* data. Module 3 responses for processing charged data; And module 4 responses for capturing peccancy vehicles. Because module 3 need handle a large number of data, *ARM9* is used in this module, the rest modules use 8051 Single-Chip. All modules use *SJA100* as bus controller. According to the structure of *ELC* sub system, and combining with the functions of each module, we can divide task set for each function module, the attributes of task including: deadline, the ways of accessing resource, required resource, running time. The required preemptive task and deadline are shown in table I, the resource mainly includes buffer, communication buffer and bus token(where time unit TTU is 2ms, the unit energy consumption is mw). The energy consumption of whole *ELC* sub system is 740J, because the total running time of *ELC* sub system is 34.5TTU, the rest time for optimizing energy consumption is 10.5TTU, let the increased unit of running time be 0.1TTU. The max and min supply voltage of Module 1, 2, 3, 4 is 3.3V and 0.8V, while module 3 is 5V and 1.2V.

We can model for task, module and communication process, and construct the *HDRE-net* model of *ELC* sub system by merging the corresponding interface. Because the task has less conflict, the using of priority can reduce the corresponding level, and the communication buffer is set to 3.

Table I Requirement of Task in ELC Sub-System

TK	*TC*	*UE*	*TK*	*TC*	*UE*	*TK*		*TK*	*TC*	*UE*	*TK*	*TC*	*UE*	*TK*	
$TK_{1,1}$	1	2	$TK_{2,1}$	1	3	$TK_{3,1}$	4	$TK_{1,5}$	1	4	$TK_{2,5}$	3	5	$TK_{3,5}$	7
$TK_{1,2}$	1	3	$TK_{2,2}$	1	5	$TK_{3,2}$	5	$TK_{1,6}$	2	8	$TK_{2,6}$	1	4	$TK_{3,6}$	4
$TK_{1,3}$	2	4	$TK_{2,3}$	1	5	$TK_{3,3}$	4	$TK_{1,7}$	2	5	$TK_{2,7}$	3	10	$TK_{3,7}$	6
$TK_{1,4}$	1	4	$TK_{2,4}$	3	5	$TK_{3,4}$	5	$TK_{1,8}$	2	5	$TK_{2,8}$	1	5	$TK_{3,8}$	4
$TK_{4,1}$	1	2	$TK_{1,4\to2,1}$	0.5	2	$TK_{3,9}$	15	$TK_{4,4}$	2	5	$TK_{3,7\to2,6}$	0.5	2	$TK_{3,7\to2,6}$	2
$TK_{4,2}$	3	20	$TK_{2,5\to3,2}$	0.5	2	$TK_{2,5\to3,2}$	2	$TK_{4,5}$	2	5	$TK_{3,8\to4,1}$	0.5	2	$TK_{3,8\to4,1}$	2
$TK_{4,3}$	3	20	$TK_{3,7\to1,5}$	0.5	2	$TK_{3,7\to1,5}$	2								

According to the division method of module's task set, we can divide task set for each module, and computing new running time of each task by computing increase time of each task set, the computed running time, supply voltage and unit energy consumption are shown in Table II. The optimized energy consumption is 525.07319, the optimization probability is 0.70955835.

Table II Energy Consumption Properties of Task

Task	*TC'*	V_{dd}	*e'*	Task	*TC'*	V_{dd}	*e'*	Task	*TC'*	V_{dd}	*e'*
$TK_{1,1}$	1.18	2.99	1.54	$TK_{2,3}$	1.46	2.6	2.77	$TK_{3,5}$	2.50	4.38	4.91
$TK_{1,2}$	44	1.01	0.02	$TK_{2,4}$	3.46	3.0	3.99	$TK_{3,6}$	2.43	4.45	2.93
$TK_{1,3}$	2.36	2.99	3.07	$TK_{2,5}$	3.46	3.0	3.99	$TK_{3,7}$	2.65	4.24	3.84
$TK_{1,4}$	1.36	2.76	2.46	$TK_{2,6}$	4.18	1.6	0.48	$TK_{3,8}$	11.33	2.28	0.32
$TK_{1,5}$	7.75	1.38	0.22	$TK_{2,7}$	10.94	1.7	1.46	$TK_{3,9}$	16.00	2.06	0.77
$TK_{1,6}$	8.75	1.64	0.46	$TK_{2,8}$	3.38	1.8	0.49	$TK_{4,1}$	1.19	2.98	1.52
$TK_{1,7}$	28	1.07	0.06	$TK_{3,1}$	2	5	4	$TK_{4,2}$	4.88	2.51	9.34
$TK_{1,8}$	28	1.07	0.06	$TK_{3,2}$	1.54	3.9	2.53	$TK_{4,3}$	4.88	2.51	9.34
$TK_{2,1}$	1.27	2.86	2.04	$TK_{3,3}$	1.43	4.0	2.27	$TK_{4,4}$	2.47	2.91	3.57
$TK_{2,2}$	1.46	2.66	2.77	$TK_{3,4}$	1.54	3.9	2.53	$TK_{4,5}$	2.47	2.91	3.57

Using the method proposed by Schmitz[7] and Yan[8], the optimized energy consumption is 577 and 618, which are higher than the method proposed in this paper, the comparison results are shown in Fig.7. The application results of Schmitz's method show that the proposed method can not only describes the characteristics of *DRE* software, but also can effectively reduce energy consumption.

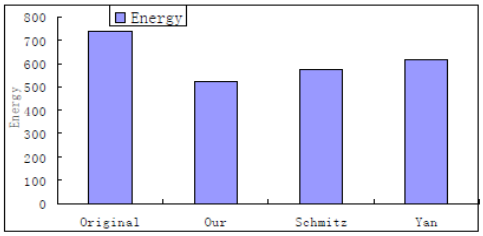

Fig. 8 Comparision Results of Optimization Method

VI. Related Works

The method that used to analyze energy consumption of *DRE* system are given in [9-11]. In[9], the authors addresses the problem of static and dynamic variable voltage scheduling of multi-rate periodic task graphs and aperiodic tasks in heterogeneous *DRE* systems. Reference[10] uses a comprehensive traffic description function at nodes and provides adequate information about the worst-case traffic behavior anywhere in the distributed network, thereby enhancing the system power management capabilities. A similar approach is given in [11], the authors incorporate a fault tolerance mechanism into *DRE* systems in order to utilize the available dynamic slack to maintain checkpoints and provide for rollbacks on faults. Previous approaches have relied on scaling down the supply voltage to reduce power consumption. However, supply voltage scaling can affect the performance. P.Paul et al[12, 13] used task graph to describe tasks and relations between tasks in *DRE* systems. More specifically, it discussed the schedulability analysis of *DRE* systems, and introduced several design optimization problems characteristic of this class of systems. The proposed method in this paper is easier to use because of the high abstraction level offered by using Petri nets. In addition, using the related tools of Petri net is also more suitable for *DRE* system.

Although the non-formal methods has made some corresponding results in the design of *DRE* systems, but they may cause some of the semantic ambiguity, in order to solve this problem, there has been some formal methods: G. Madl et al[14] used Time Automata (TA) to model various components of the non-preemptive real-time distributed embedded systems and converted the system scheduling problem into TA state reachability. As the TA model implied the existence of global clock, it unfits for modeling the Distributed Systems. V. Marcel et al[15] used the time extended of Vienna Development Method (*VDM*++) to stipulate *DRE* systems, and used *VDM* verification tools to verify the properties of system. However, comparing with other formal methods, the *VDM* may be more difficult to understand and grasp for

developers. A resource-based time Petri Net is proposed in [16] to model the *DRE* systems and analyzed the corresponding semantic, properties. In our work, it allows user to describe and model on the non-function of *DRE* system by using Petri net, which can be helpful in analyzing its performance, and they didn't describe the communication between modules which is the key issue of *DRE* systems.

VII. Conclusion

In this paper, we have proposed a *HDRE-net* to model and analyze energy consumption of *DRE* software. This approach is based on a formal model, Petri net, that allows to consider different components of *DRE* software. According to the characteristics of module, we divide it into leaf module and fork module, then dividing task set for module based on the concept of critical task; the *DVS* adjustment method of task set and energy consumption steps of whole application are advanced, its enforcement algorithm is also given. Finally, we explain the effectiveness and feasibility of the method by *ELC* sub system. Comparing with other related works, the advantages of this paper are: Constructing energy consumption model of *DRE* software; proposing new idle time allocation method and offline *DVS* scheduling of *DRE* model.

The study of *DRE* software is still underway at present. Our current research is focused on exploring formal method as means to improve its mapping into *DRE*'s architecture. The following two aspects are the main work in the next phase: (1) further improves this method, consider the fault-tolerant of each task to assurance system's schedulability; (2) developing the corresponding tools to support the modeling.

Acknowledgment

This paper is supported by key Foundation of Shanghai Educational Committee (07ZZ164, 06OZ016), Foundation of Shanghai Institute of Technology (YJ2004-05) and key subject of Shanghai Institute of Technology (Computer science and technology), Fund of Key Laboratory of Shanghai Science and Technology (09DZ2272600), the Open Research Foundation Institute of Technology of China under Grant No. YJ2009-17.

References

[1] C. E. Pereira, L. Carro. “Distributed Real-time Embedded Systems: Recent Advances, Future Trends and Their Impact on Manufacturing Plant Control,” *Annual Reviews in Control*, 2007, 31(1). pp. 81-92.

[2] R. Mishra, N. Rastogi, Z. Dakai, D. Mosse. “Energy aware scheduling for distributed real-time systems,” *In: Proceedings of 2003 International Conference on Parallel and Distributed Processing Symposium*, 2003, p21.

[3] R. Min, T. Furrer, A. Chandrakasan. “Dynamic voltage scaling techniques for distributed microsensor networks,” *IEEE Computer Society Annual Workshop on VLSI*. 2000, pp.43.

[4] J. Sifakis. “Using Petri nets for performance evaluation,” *In: Proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems*. The Netherlands: North-Holland Publishing Co. 1977. pp. 75-93.

[5] T. Murata. “Petri nets: properties, analysis and application,” *In: Proceedings of the IEEE*. 1989,77(4). pp. 40-581.

[6] K. Etschberger. “Controller Area Network-Basics, Protocols, Chips and Applications,” *IXXAT Press*. Weingarten, Germany.2001.

[7] M. T.Schmitz, B. M. Al-Hashimi, P. Eles. “Iterative schedule optimization for voltage scalable distributed embedded systems,” *ACM Transactions on Embedded Computing Systems*. 2004, 3(1). pp.182-217.

[8] L. Yan, J. Luo, N. K.Jha .“ Joint dynamic vltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2005, 24(7):1030-1041.

[9] J. Luo, N. K.Jha. “Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems”. *In: Proceedings of the 2002 Conference on Asia South Pacific Design Automation/VLSI Design*. Washington, DC, USA: IEEE Computer Society. 2002. pp. 719-726.

[10] R. N. Mahapatra, W. Zhao. “An energy-efficient slack distribution technique for multimode distributed real-time embedded systems”. *IEEE Transactions on Parallel and Distributed Systems*. 2005, 16(7): 650-662.

[11] S. Acharya, R. Mahapatra. “A dynamic slack management technique for real-time distributed embedded systems”. *IEEE Transactions on Computers*, 2008, 57(2): 215-230.

[12] P. Pop, P. Eles, Z. Peng, T. Pop. “Analysis and optimization of distributed real-time embedded systems”. *ACM Transactions on Design Automation of Electronic Systems*. 2006, 11(3): 593-625.

[13] P. Pop, K. H.Poulsen, V. Izosimov, P. Eles. “Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems”. *In: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. ACM: New Yor, 2007: 233-238.

[14] G. Madl, S. Abdelwahed, D. C. Schmidt. “Verifying distributed real-time properties of embedded systems via graph transformations and model checking”. *Real-Time Systems*. 2006, 33(1-3): 77-100.

[15] M. Verhoef, P. G. Larsen, J. Hooman. “Modeling and validating distributed embedded real-time systems with VDM++”. *FM 2006: Formal Methods*. Springer, 2006,4085.

[16] H. T. Zhang, Y. F. Ai. “Time analysis of scheduling sequences based on *Petri nets for distributed real-time embedded systems”. In: Proceedings of the 2nd IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*. IEEE Computer Society, 2006:1-5.

Liqiong Chen. She was born in 1982, Ph. D. candidate. Her research interests include distributed computing, embedded systems and formal methods.

Guisheng Fan. He was born in 1980, Ph. D. candidate. His research interests include service oriented computing, distributed computing and formal methods.

Yunxiang Liu. He was born in 1967, professor, Ph. D. supervisor, IEEE senior member. His research interests include software engineering, information security and formal methods.