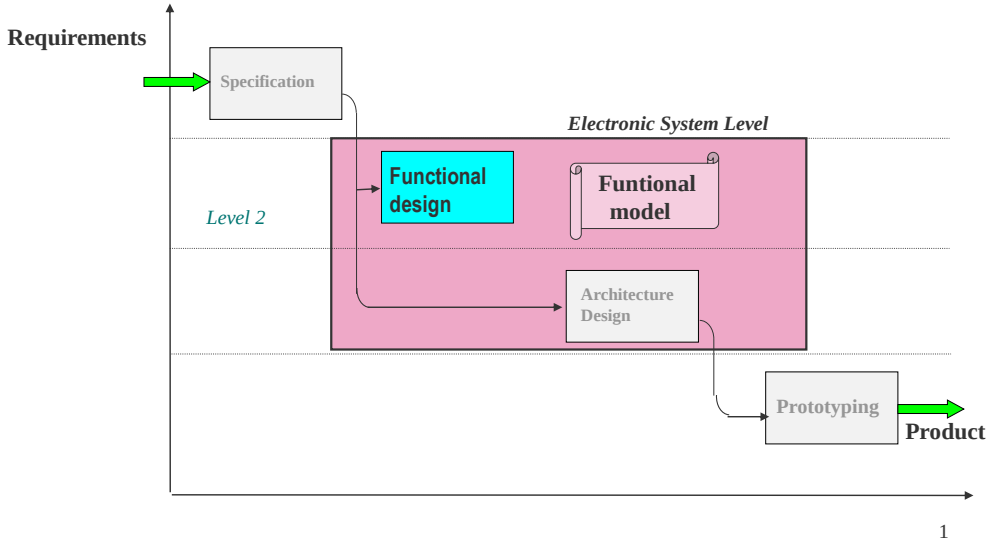
Conception fonctionnelle



1

On rappelle que l'étape de conception fonctionnelle a pour objectif d'apporter les éléments de décision pour le choix d'une implémentation capable de satisfaire l'ensemble des spécifications (fonctionnelles, non fonctionnelle, technologiques et économiques), sachant que cette implémentation doit être appropriée en terme de coût et temps de développement entre autres [c'est le COMMENT].

Conception fonctionnelle

A. *Le modèle fonctionnelle*

B. Analyse des performances

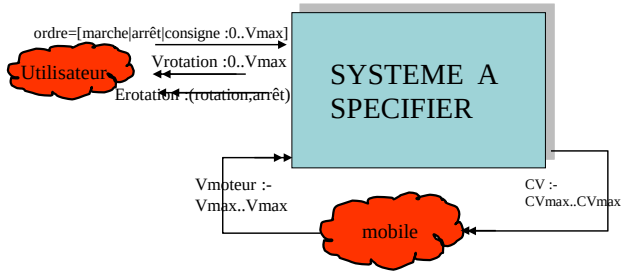
C. Démarche de conception

D. Modèles génériques

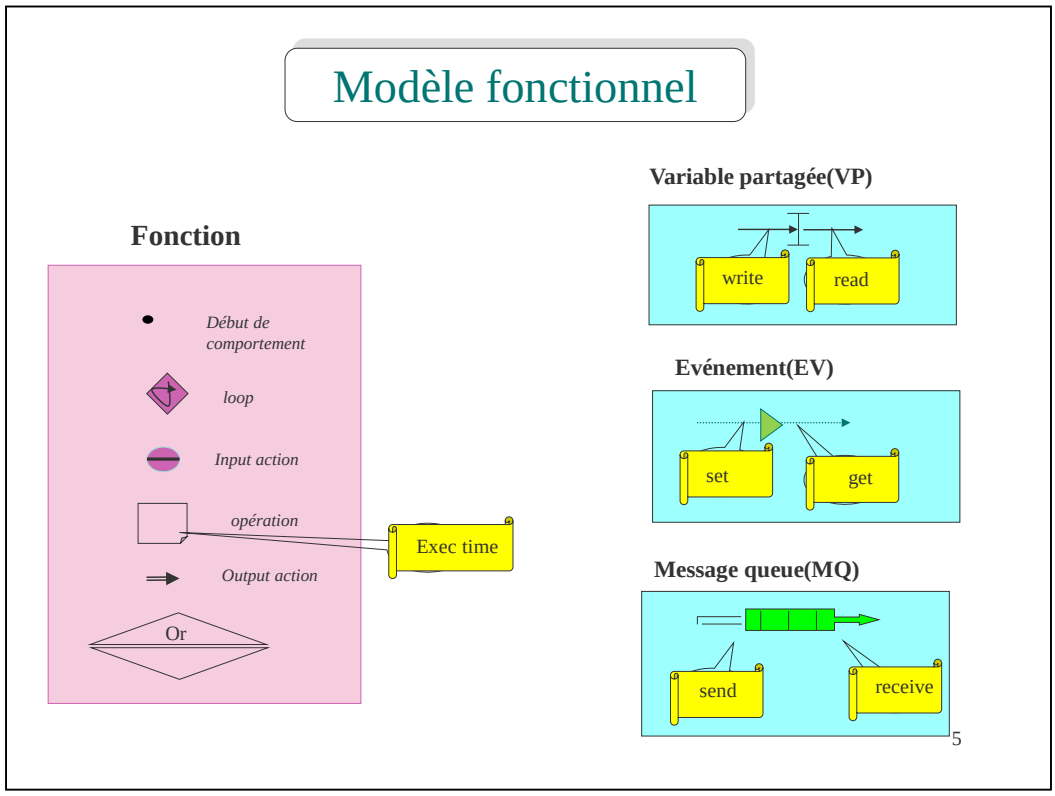
Illustration

L'objectif est de disposer d'un système autonome assurant le déplacement d'un chariot sur une trajectoire rectiligne selon un gabarit de vitesse donné. Plus précisément, il s'agit de faire tourner le moteur relié à l'axe à une vitesse constante déterminée par l'utilisateur avant le lancement de l'opération. Le chariot se déplace grâce à un moteur unique entrainant les 2 roues pendant une durée donnée, fixe ici, sachant que la montée en vitesse se fait à accélération constante, et de même pour la décélération. L'utilisateur peut décider de sa mise en route ou de son'arrêt. Il fixe également la consigne de vitesse (affichée en permanence à l'utilisateur). Le tapis est supposé commande par l'utilisateur.

manutentionaire



Exemple de document type d'entrée de l'étape pour le probleme du contrôle du chariot. Ce modèle est UNTIMED, i.e les actions sont considérées comme s'exécutant en temps nul



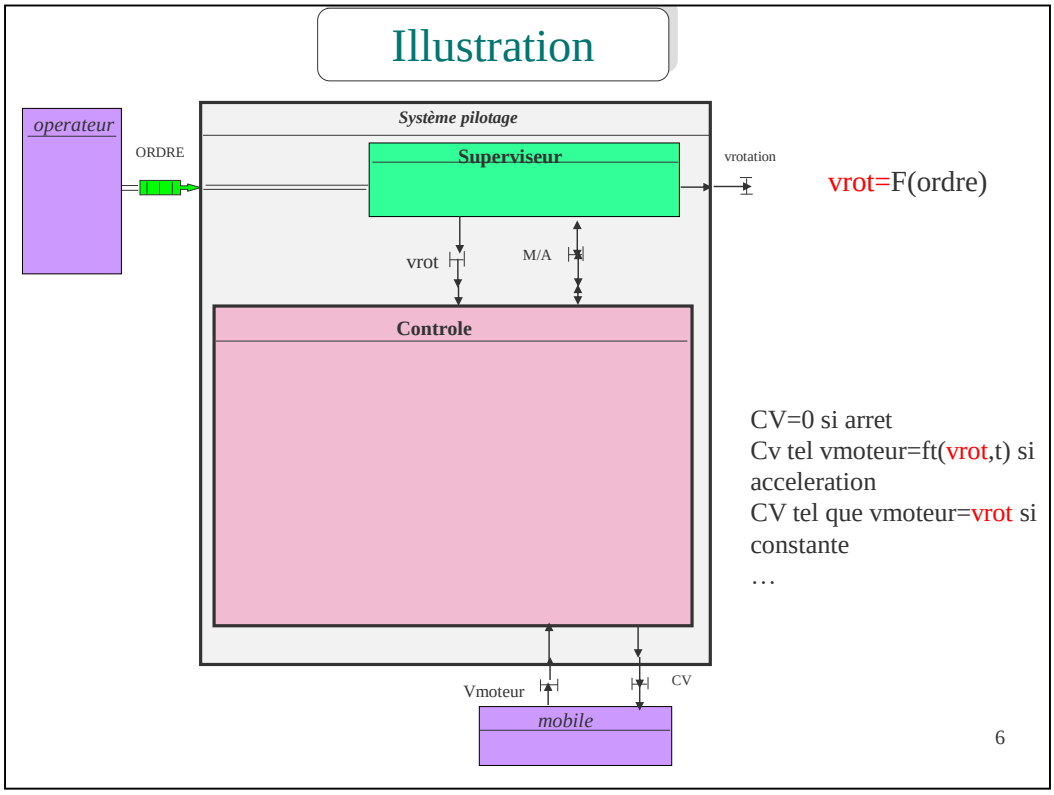
Modèle dit system-level (transactionnel) décrivant une évolution parallèle d'un ensemble de programmes autonomes (assimilables à des machines séquentiels avec leur contexte propre). Le modèle dit timé se situe à un niveau intermédiaire entre les spécifications et la structure physique. Sa mission est de permettre de valider l'architecture fonctionnelle et de déduire

une solution d'implémentation particulière(microprocesseur, FPGA, ASIC, etc) par exploration d'architectures. Le modèle repose sur deux points de vue complémentaires et orthogonaux:

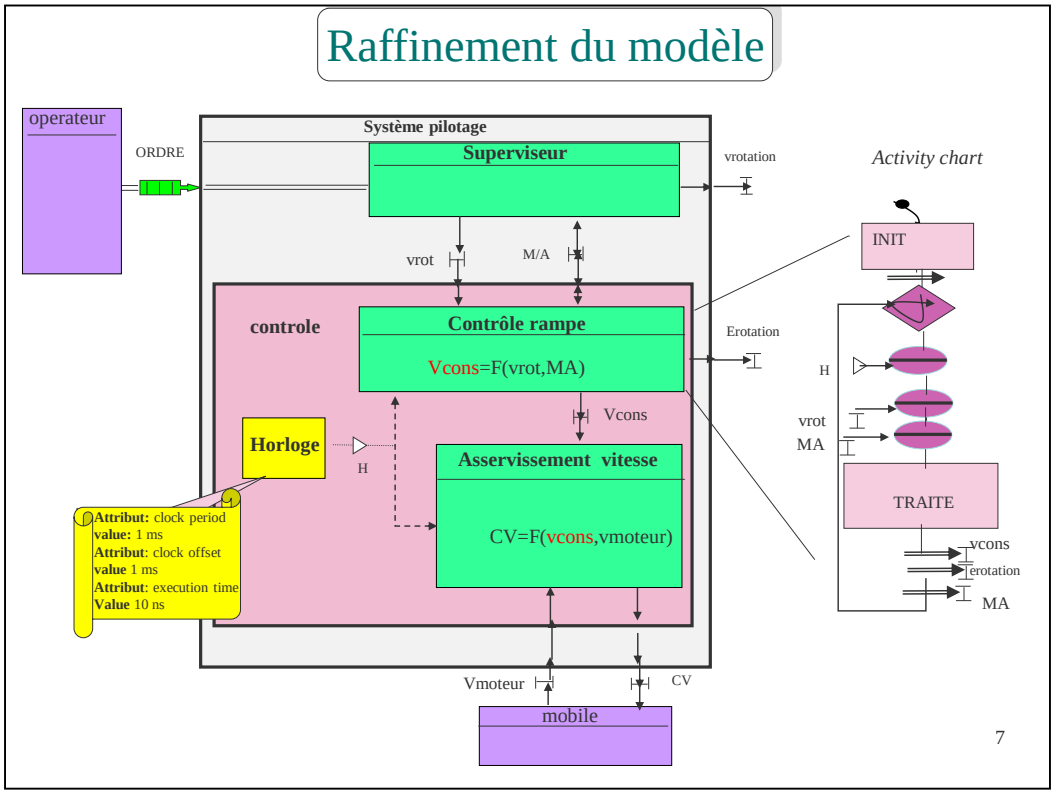
* le point de vue organisationnel: décrit les fonctions ou composants actifs et leurs relations (ports, VE, événements).Dimension spatiale. Le modèle est hiérarchique, une fonction pouvant se raffiner en sous fonctions.

* le point de vue comportemental: décrit l'ensemble des opération ainsi que leur séquencement partiel ou total pour chaque fonction ou composant. Dimension temporelle

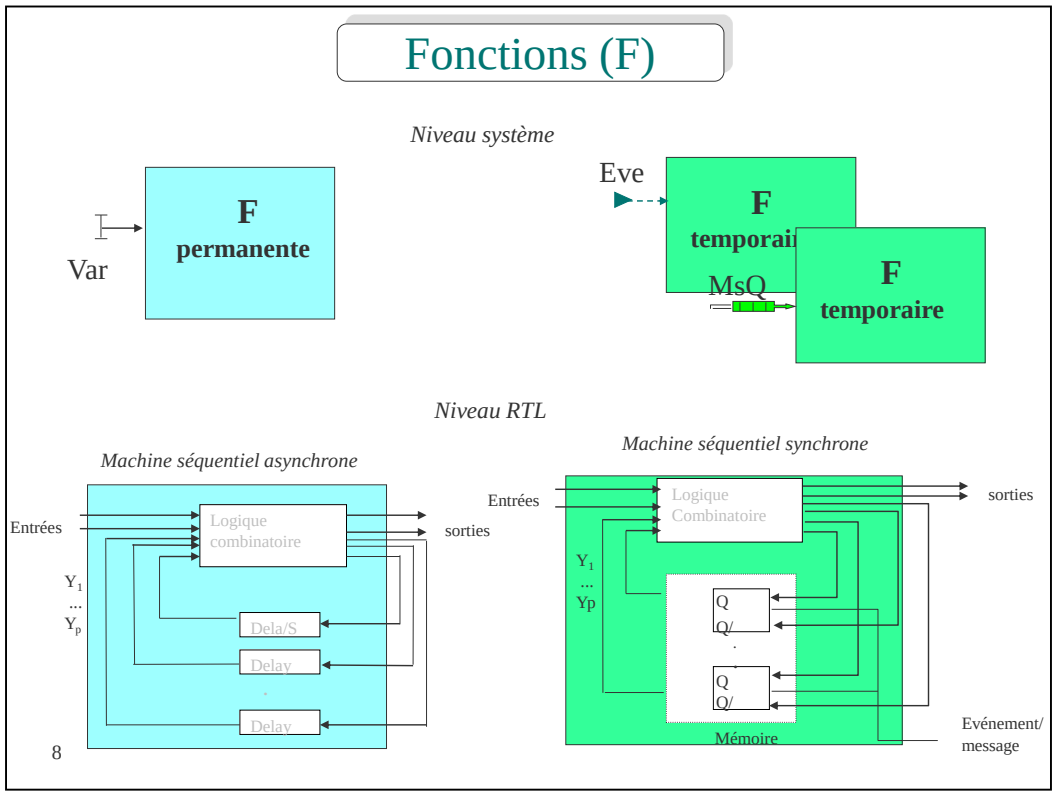
A ces éléments s'ajoute des attributs qui permettent de définir des contraintes que l'on ne peut pas définir sous forme graphique.



Une solution fonctionnelle pour le contrôle de la trajectoire du mobile en recherchant les variables internes nécessaires à un fonctionnement correct et optimale du système embarqué. L'analyse de la spécification montre que l'évolution de cv depend de la vitesse du moteur ,de la consigne vrot et du temps. L'évolution de cv ne depend pas de l'ordre et deux fonctions asynchrones sont construites autour de la variable partagée entre les 2 fonctions. On ajoute *m/A* pour controler l'activation/desactivation du contrôle.



Le modèle est hiérarchique avec ici une solution permettant de rendre les évolutions de cv discretes (solution numérique privilégiée) en fonction de la précision imposée . Au plus bas niveau, le comportement des fonctions élémentaires (séquentielles) est exprimé par un activity chart qui décrit l'évolution temporelle de la fonction



Une fonction est un opérateur permettant de produire des données en sortie à partir de données en entrées. Les fonctions peuvent être de deux types:

permanentes: elles évoluent librement sans aucune contrainte temporelle. Ces fonctions sont assimilables à un système séquentiel asynchrone: dès qu'une donnée est modifiée, le système peut réagir librement et changer d'état (cas de la perceuse vue en Elec3)

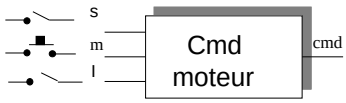
Temporaires: elles n'évoluent qu'à des instants précis liés à l'arrivée d'un ou plusieurs événements (horloge ou autre) ou messages. Ces fonctions sont assimilables à un système séquentiel synchrone. En dehors, la fonction est inactive et donc aucune évolution d'état interne n'est possible, même si une donnée change entre temps.

(cas d'un syst de clignotement d'une lampe pour un feu tricolore par exemple).

Règle d'usage: donner un nom de verbe suivi d'un nom de l'objet transformé (gérer le clavier).

Machine séquentielle asynchrone

Symbole



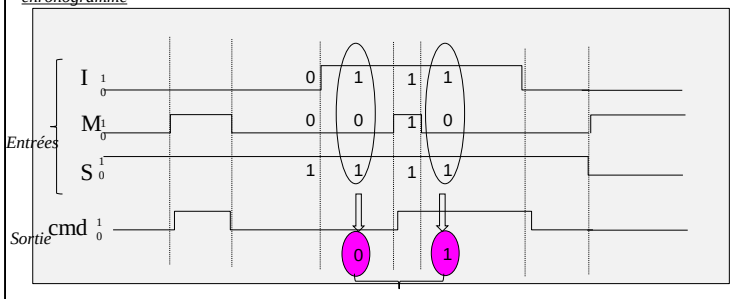
Un circuit de commande un moteur à l'aide de sortie *cmd*. 3 entrées s,m,I permettent de commander l'évolution du système uniquement à partir de leurs niveaux.

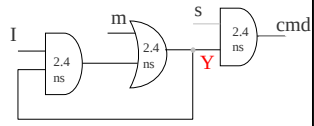
S est un interrupteur de sécurité à deux positions. s=0 verrouille le mécanisme.

I est un interrupteur précisant le mode de fonctionnement (intermittent si *i=0* ou continu si *i=1*)

M est un bouton poussoir permettant de commander la mise en route du moteur.

chronogramme



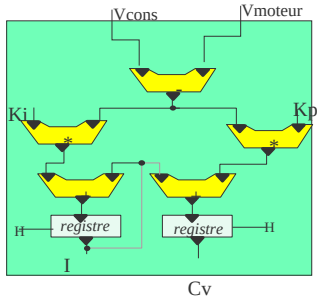


Solution RTL

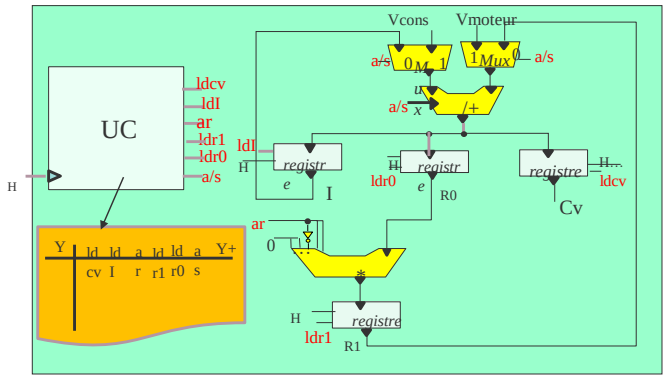
La modification de la sortie *cmd* peut s'effectuer sans contrainte d'événement sur modification d'une des entrées

Machine séquentielle synchrone

RTL faible latence



RTL faible surface

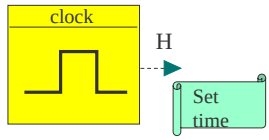


La fonction d'asservissement doit être correctement échantillonnée pour assurer sa fonctionnalité et pouvoir modifier la sortie CV à des instants précis et non pas à chaque modification de l'entree *vmoteur.* Ceci se traduit au niveau RTL par la présence d'un registre controlé par une horloge.

Une fonction opérateur modifie l'état d'un des 4 boutons poussoirs. L'interface ordre produit un message en réponse. On peut noter que l'écriture et la lecture de BP sont totalement asynchrones. Les intervalles de temps séparant ces opératoires sont donc non déterministes. Dans l'exemple, le BP est appuyé puis relacher. Il engendre l'émission d'un ordre à cette occasion.

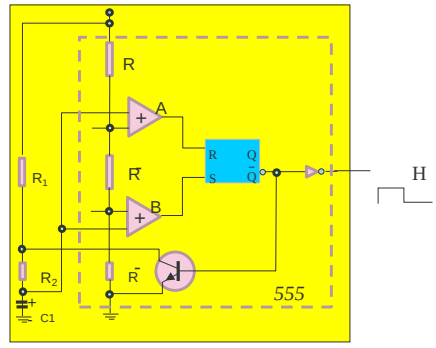
Fonction permanente prédéfinie *clock generator*

System Level

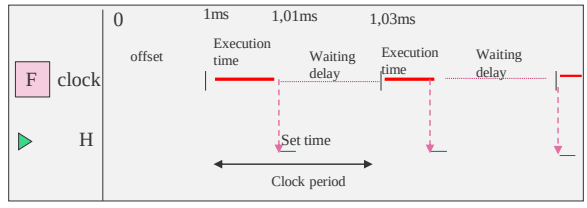


attribut	Value	Unit
Clock period	5	ms
Clock offset	1	ms
Execution time	10	us

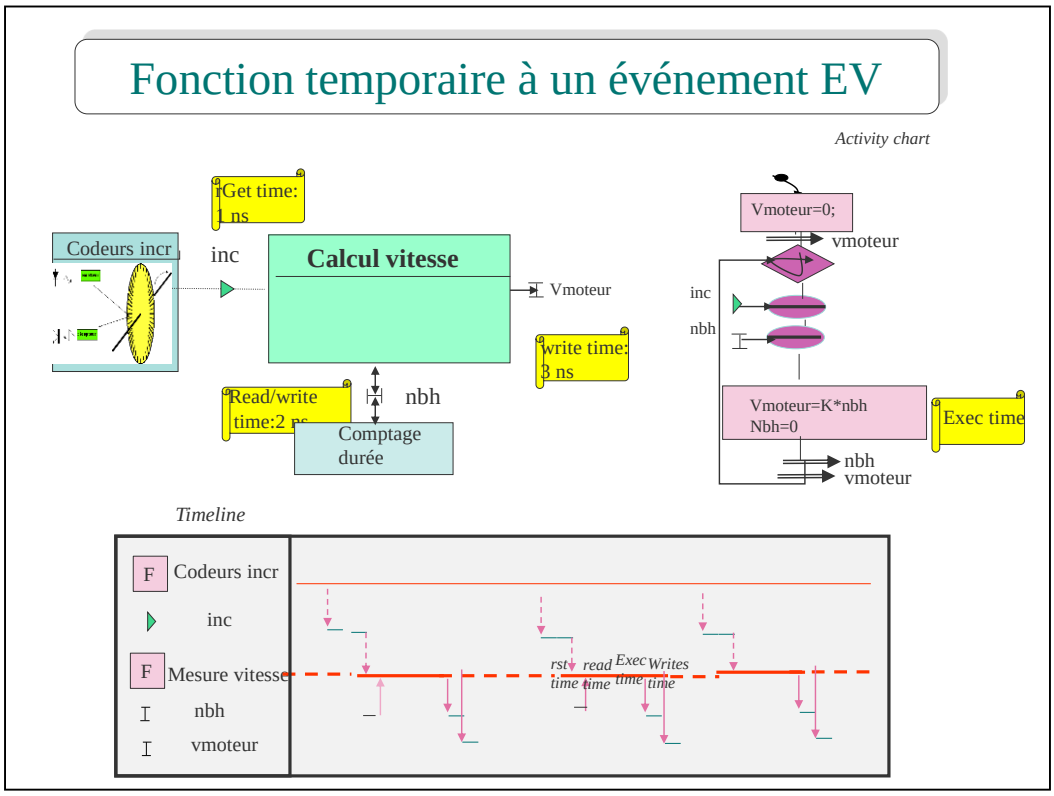
RTL



Timeline

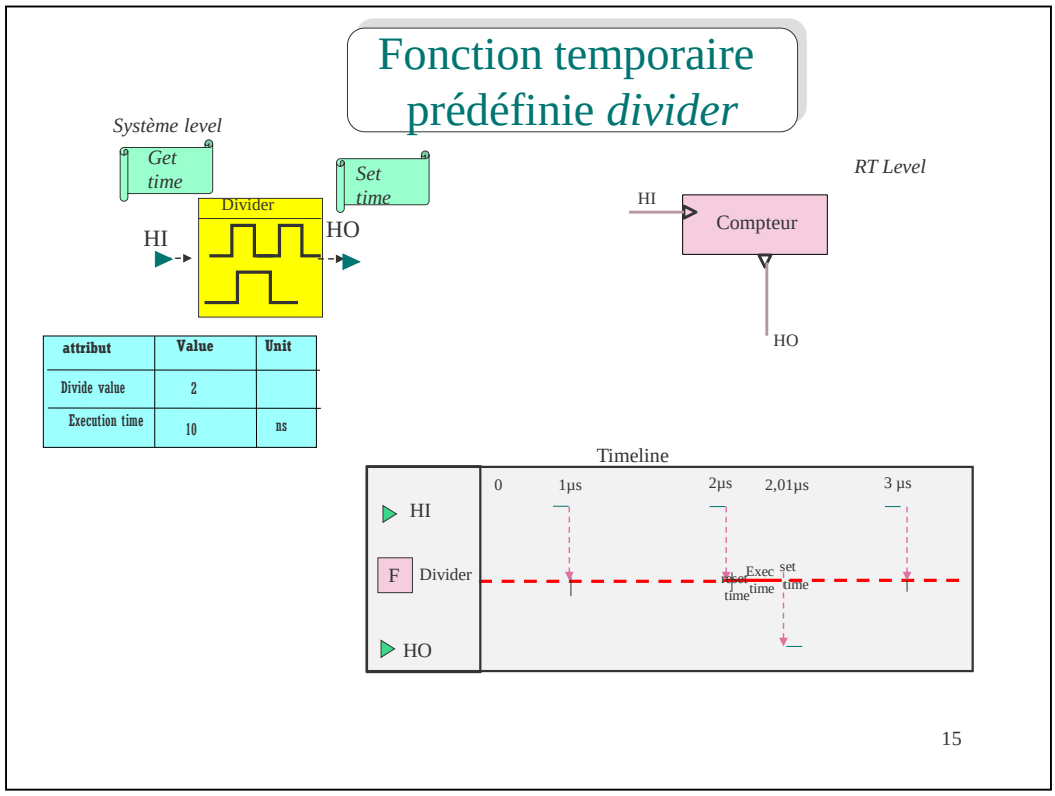


Des fonctions permanente prédéfinies peuvent être spécifiées tel que le *clock generator* Elles ne nécessitent aucune définition de comportement et des attributs permettent de définir les caractéristiques de timing.



un fonction temporaire synchrone à un événement permet de synchroniser les écritures sur la sortie. Ici, l'événement H permet de discrétiser les évolution discrètes de cv. Si la variable partagée peut etre assimilée à un registre au niveau RTL, l'événement peut être assimilé à un signal logique. IL N'A PAS DE VALEUR ASSOCIEE !!!

Lorsque l’on souhaite modifier une sortie (ici cv) à des instants périodiques, et si la fonction n’est synchrone à aucun autre événement, on peut utiliser une fonction échantillonnée et fixer la période d’échantillonnage plutôt qu’un evenement d’horloge. d’horloge



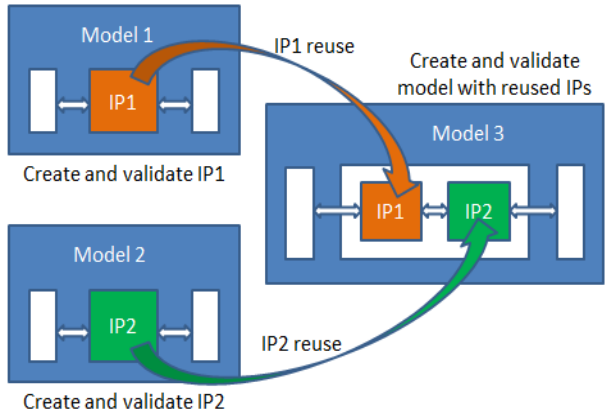
Le divider est une autre fonction prédéfinie. Le rapport entre la période de sortie et celle en entrée est fixé par l'attribut *divide value*. Le *execution time* correspond globalement au temps de traversée des FF.

Dans cette fonction, les evts modifie la même variable partagée et doivent de toutes les façon être traités séuentiellement. Il et possible que le traitement de l'événement précédent ne soit pas terminé. On peut mémoriser avec l'attribut eventpolicy à booleen ou counter

Une fonction temporaire synchrone à un MSQ s'execute tant qu'il y a des messages dans la file d'attente. LA fonction productrice des messages se bloque si la file n' a pas la capacité à absorber plus de message. Sur le scénario, cela se produit au 3eme message (115 ns) car un message est en traitement et un message est en attente pour une capacité de 1.

Sans protection de type semaphore, la valeur de la variable *nbh* qui est à 2 ecrivain peut être incohérente si les ecritures se chevauchent dans le temps. Selon l'ordre des acces, une place peut être indiquée en plus du nombre réel (cas du scénario ci-dessus) ou en moins (barriere de sortie modifie nbh en premier)

IP Reuse



19

Une fois le comportement d'une fonction ou d'un ensemble de fonctions validées dans un modèle, cette fonction peut être réutilisée comme IP dans un autre modèle. Cette fonctionnalité permet de capitaliser et réduire le temps de développement de nouveaux modèles time to market)

L’outil d’ESL cofluent autorise la modularité. Comme au niveau RTL, des fonctions peuvent être décrites et misent au point des diagrammes séparés, mis dans une bibliothèque virtuelle puis réutilisés dans différents projets(pilotage vehicule ou ascenseur par exemple) en ajustant éventuellement certains paramètres. On repère les composants à l’icone fleche en bas à gauche du cadre de la fonction

Conception fonctionnelle

A. Le modèle fonctionnelle

B. Analyse des performances

C. Démarche de conception

D. Modèles génériques

La validation fonctionnelle doit déterminer si le modèle construit est conforme aux spécifications fonctionnelle et non focntionnelles (contraintes de timing, precision, etc). Cette validation necessite la définition du comportemetn des entités de l'environnement. On reprend pour cela le modèle de comportement des entités défini lors de l'anayse et modelisation de l'environnement en phase de specifications. Le moteur du mobile est ainsi décrit à partir des equations aux recurrences. Certains parametres pourraient génériques tels que TAU pour visualiser la réponse de l'asservissement en fonction de la charge du mobile. Pour l'opérateur , on décrit un scénario d'exploitation du systeme(un use case)

L’analyse temporelle peut permettre d analyser l’activité des objets du modèle en traçant les modifications de VP, l’emission/reception des evenements, le depot/retrait de messages effectués par les fonctions. Il est possible de vérifier l’impact des valeurs d’attributs sur le modèle tel présence de blocage inattendu entre deux fonctions lié à la capacité des FIFO, la prise en compte de sémaphore, la police d’événements.

Il est possible de visualiser l'activité moyenne des fonctions (le temps moyen en running) etc. On peut ainsi faire du profiling et disposer d'informations pour le choix futur de la plateforme d'exécution.

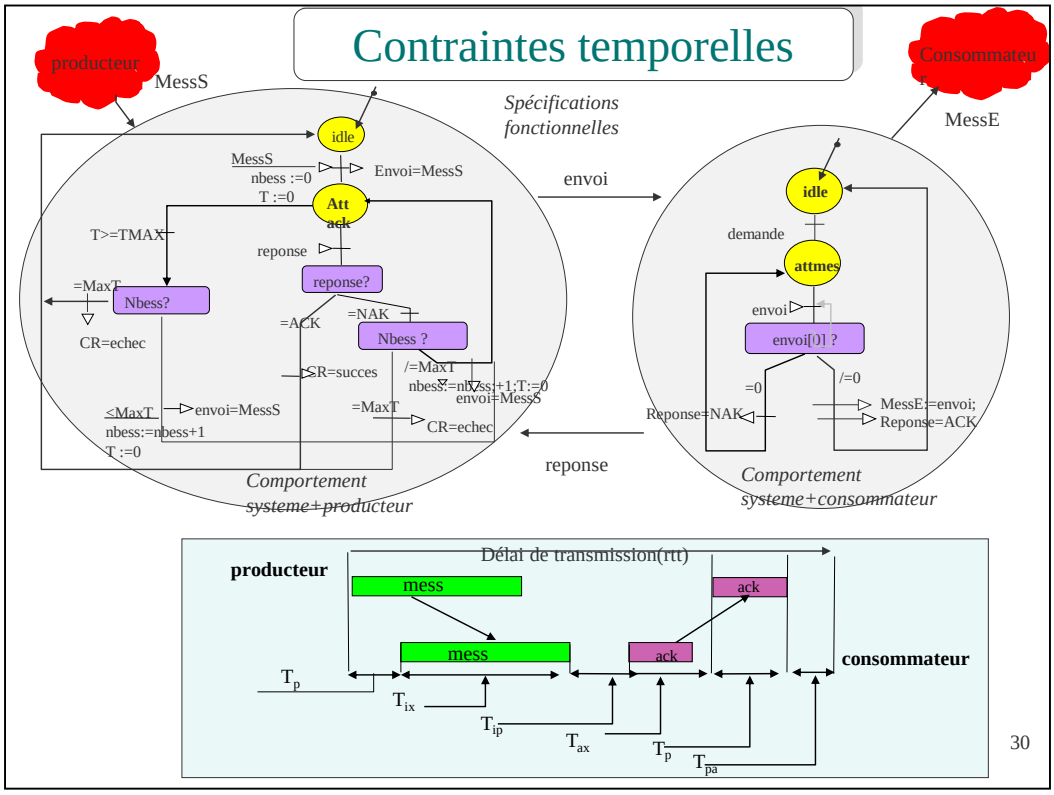
Les continous chart permettent d’afficher les valeurs de variables partagées comme ici vcons, vrot,cv et vmoteur. On peut constater que l’erreur statique est très faible.

On peut valider des contraintes de precision telle que la contrainte de précision de 10% sur vrot. On verifie la valeur grâce à des curseurs positionné sur 2 points successifs de la rampe descendante qui a une pente plus sevère que la montée et on observe une valeur de 3 rd/s = 10% de 30 rd/sec.

Grâce à la définition de paramètre générique, l'utilisateur peut réaliser une exploration des résultats du modèle par rapport aux variations des paramètres. On peut alors vérifier la variation de la précision en fonction du pas.

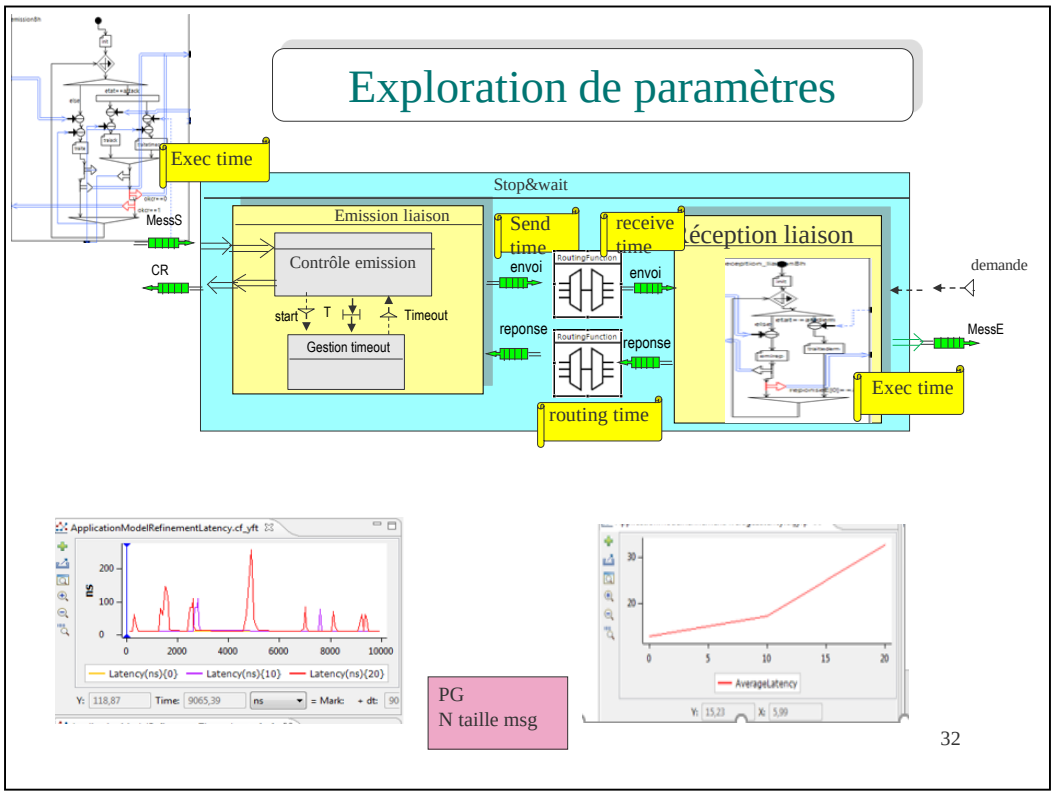
On peut également demander une exploration automatique des paramètres. Les courbes sont superposées dans la meme fenêtre avec une légende indiquant quelle est la valeur du paramètre ayant conduit à la courbe.

On peut raffiner le modèle en introduisant des instructions permettant d’obtenir des mesures statistiques Ici, on visualise l’évolution de l’écart de vitesse en fonction du pas sous forme graphique ou textuelle.



La mise en œuvre d'un système de communication peut s'accompagner de contraintes temporelles tel qu'un temps de latence(délai de transmission) maximal entre deux stations. Ce délai est facteur de plusieurs paramètres de timing ou encore de la longueur du message, paramètres qui peuvent servir de base à la mesure et l'optimisation de ce délai.

Des API permettent de faciliter des mesures de temps de latence, mesures pour lesquelles il faut connaitre la taille du message si celle-ci est paramétrable (methode user_data_length).



Outre les attributs *exec time* qui peuvent modéliser les temps de traitement du message ou de l'acquittement, le *send time* peut modeliser le temps d'échantillonnage du message (tix et tax) et le *routing time* peut modéliser le temps de propagation du signal. On peut alors vérifier le temps de la latence instantanés en fonction de la taille du message (paramètre N) ou des temps élémentaires ou des statistiques tel que le délai de transmission moyen

Rappel: La valeur dans le MSQ (transfert de données) ne s’utilise qu’une seule fois contrairement à la VP (partage de données)

Si on observe bien les spécifications, c, cts, txrdy,txd apparaissent dans des tests sur leur valeur et/ou sont affectés dans des opérations. Pour ces liens, on sélectionne une variable partagée.

Wr est une impulsion. En dehors de l'attente du signal, il n' y a aucun test ni affectation de valeur. Pour ce lien, on selectionne une relation d'événement de synchronisation

data est un lien de type information. Il s'agit normalement d'une relation de type transfert de données (MSQ) Mais data n'apparait pas en condition car c'est l'événement Wr qui est utilisé indique l'occurrence du transfert. Donc on peut implémenter la

A. Analyse des spécifications et recherche des données et événements de couplage des entités

B. Construction des fonctions générant ou exploitant ces données et événements

C. Vérification cette première décomposition

Un enchainement purement séquentiel des opérations (algorithme) est il envisageable? Les opérations d'affectation de txd ne sont pas déclenchées sur *wr*. Dans la tentative d'algorithme ci-dessus, on attend l'occurrence de *wr* et on enchaine les opérations d'affectation de txd en séquence avant d'attende à nouveau l'événement *wr* si aucune nouvelle donnée n'est présentée dans l'intervalle, c'est ok. Dans le cas contraire (scénario ci-dessus), il y a de forte chance que cette donnée ne soit plus valide lorsqu'on sera en mesure de la lire (contrainte de deadline). Toute tentative de prendre en compte *wr* pendant la transmission de la donnée en cours est également voué à l'échec car txd n'aura pas jamais la forme souhaitée avec des périodes de bit totalement aléatoires.

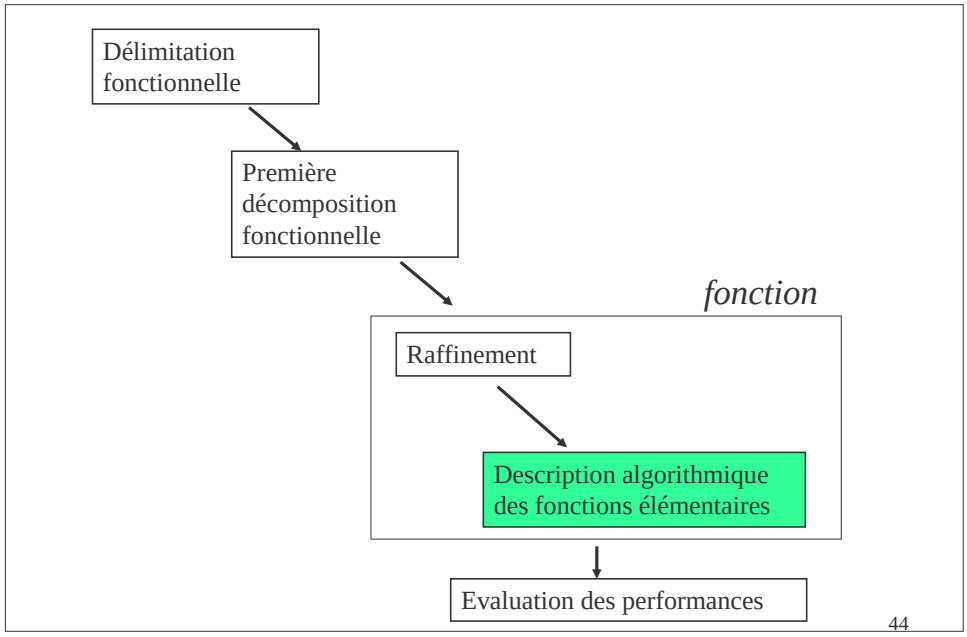
La décomposition fonctionnelle doit respecter la périodicité d'écriture des données: les modifications de txd ne sont pas liées à l'occurrence de l'evt *wr.* Il faut donc une variable interne *D* pour mémoriser la valeur de la nouvelle donnée à transmettre sur *wr* (opération réalisée par la fonction *transfert*) en attendant que l'émission de la donnée en cours soit achevée par la fonction *contrôle.*

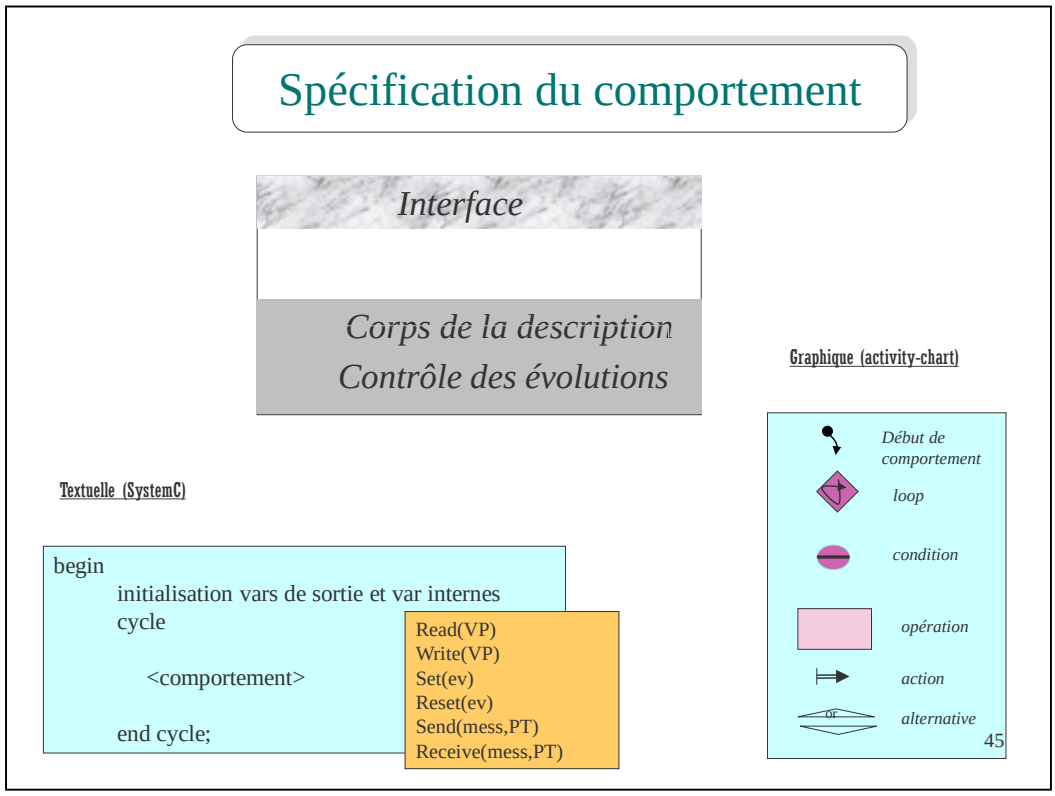
Plutôt que d'emettre un evenement *trans* à la fonction *transfert* pour qu'elle réalise l'opération txrdy=true, *Txrdy* est mise dans les 2 sens et on réalise l'opération dans *contrôle*. Cela élimine l'événement de couplage et il n'y a plus qu'un seul état dans la fonction de transfert (modifications en rouge sur les charts)

Les fonctions de la solution sont séquentielles (pas d'asynchronisme) mais se pose le probleme de l'évolution des valeurs de T. On rappelle qu'au niveau spécifications, l'évolution du temps est considérée comme un phénomène physique qu'il n'a pas besoin de préciser. Au niveau conception, on ajoute donc une fonction interne qui va s'occuper de produire les valeurs de cette variable nécessaires à l'évolution de txd.

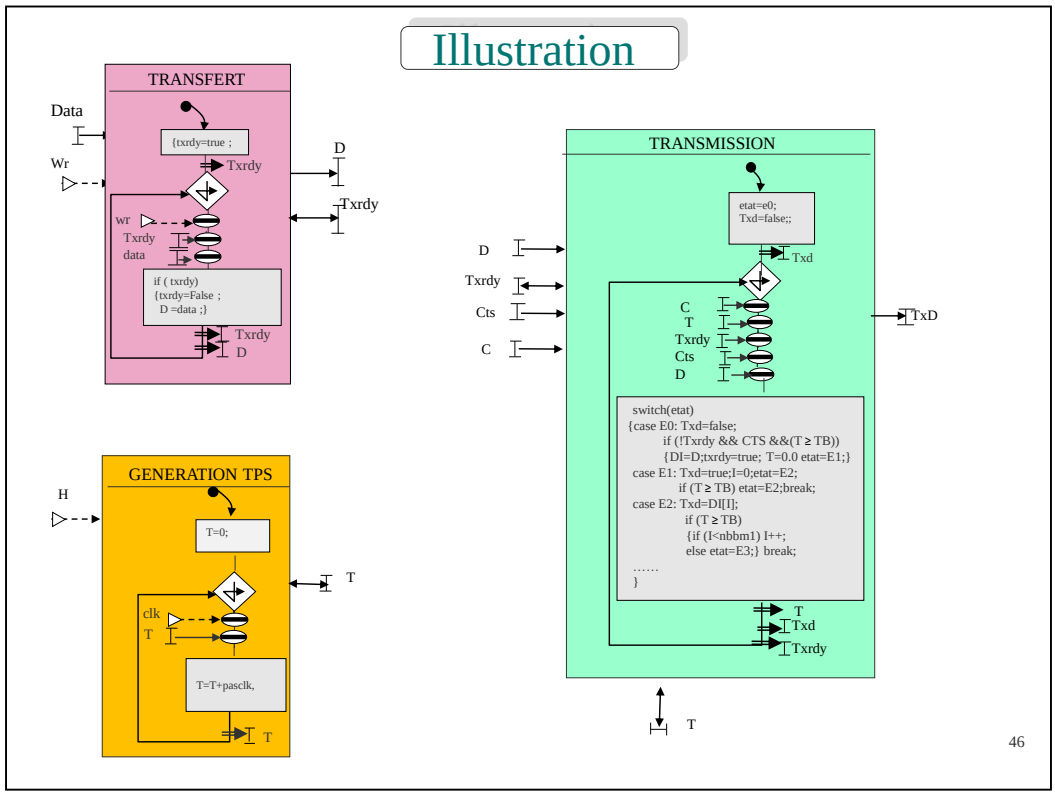
Sachant qu'une infinité de valeur de T (et donc une évolution continue de *generation-tps)* n'est pas utile, il est possible de re decomposer la fonction *generation_tps* pour discrétiser les évolutions de la variable *T* et améliorer la précision de ses valeurs. Cette solution est propice à une implémentation sur système numérique.

Description comportementale





Le modèle fonctionnel doit être complété par l'expression du séquencement des opérations à partir d'une composition d'opération élémentaires et de structures de contrôle. Il peut être exprimé textuellement ou graphiquement.



On décrit finalement le comportement de chaque fonction élémentaire avec un activity chart.