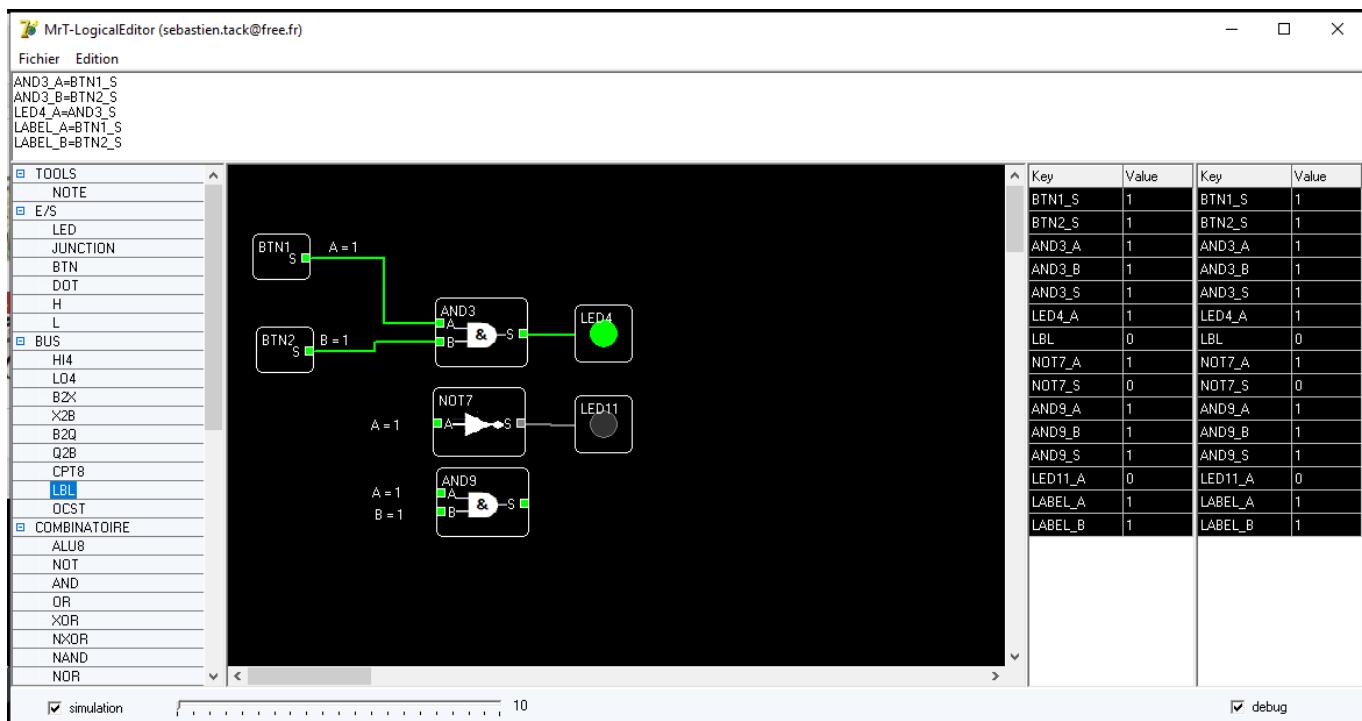


LOGICIEL LOGICAL EDITOR



SimuLogic — Simulateur logique et micro-contrôleur pédagogique *Lazarus / FreePascal — Octobre 2025*

Objectif du projet

SimuLogic est un environnement développé sous Lazarus / FreePascal permettant :

- de construire et simuler des circuits logiques combinatoires et séquentiels ;
- de composer progressivement un micro-contrôleur rudimentaire, avec :
- un gestionnaire de bus,
- une mémoire ROM contenant un programme,
- une RAM 8 bits,
- un registre d'instruction (IR),
- un registre accumulateur (A),
- une unité arithmétique et logique (ALU),
- un compteur ordinal (PC),
- un contrôleur / unité de commande capable de décoder les instructions.

L'objectif est un outil pédagogique puissant, simple d'accès, adapté à l'enseignement des systèmes logiques, des microarchitectures et de la programmation bas-niveau.

Philosophie du logiciel

1. Le concret d'abord

SimuLogic rend visibles :

- la propagation des bits,
- le rôle des bascules et des fronts d'horloge,

- la circulation des données dans les bus,
- le fonctionnement interne d'un micro-contrôleur.

L'élève observe le système fonctionner étape par étape, ce qui ancre les concepts abstraits dans l'expérience concrète.

2. Une logique unifiée : combinatoire + séquentielle

Le moteur interne repose sur deux passes :

1. Passage séquentiel – traitement des signaux mémorisés (@signal)
2. Passage combinatoire – propagation logique instantanée

Cette architecture clarifie la différence entre mémoire et logique et permet de créer des circuits fiables et pédagogiques.

3. Construire un micro-contrôleur, brique par brique

Chaque composant du micro-contrôleur est représenté par un bloc :

- PC (compteur ordinal)
- IR (registre d'instruction)
- A (accumulateur)
- ALU (arithmétique et logique)
- ROM (programme)
- RAM (mémoire de données)
- BUS (sélection de sources)
- CTRL (décodage d'instruction)

L'utilisateur peut câbler son propre micro-contrôleur et observer son fonctionnement interne, ce qu'un matériel réel ne permet pas de manière aussi transparente.

Construction de circuits

L'utilisateur dispose de nombreux composants :

- portes logiques (AND, OR, XOR, NOT...)
- multiplexeurs
- bascules (RS, JK, D, T)
- compteurs
- bus 4 ou 8 bits
- registres
- RAM et ROM
- afficheurs (LED, 7 segments)
- labels (renommage et routage local)
- notes

Chaque bloc possède :

- des entrées et sorties nommées,
- des équations logiques internes en notation RPN,
- un préfixe automatique évitant les collisions de noms.

Les connexions se font intuitivement par clic, même dans des circuits complexes.

Simulation

- Simulation en temps réel via timer ou en mode pas-à-pas.
- Affichage direct de :
 - l'état des bits,
 - les valeurs des bus,
 - l'état des registres,
 - le cycle d'exécution d'une instruction.

La mémoire interne peut être inspectée et figée pour analyser un cycle.

Simulation d'un micro-contrôleur rudimentaire

Pipeline minimal

- ```

1. **FETCH**
 * PC fournit l'adresse
 * ROM renvoie l'octet d'instruction
 * IR se charge

2. **DECODE**
 * Le contrôleur active LOAD_A, ALU_SEL, BUS_SEL, RAM_RW, etc.

3. **EXECUTE**
 * ALU calcule
 * Accumulateur ou RAM mis à jour selon l'instruction

```

## Jeu d'instructions typique

1. LDA imm : charger une constante
2. ADD imm : addition immédiate
3. STA addr : stocker A en RAM
4. LDA addr : charger depuis la RAM
5. JMP addr : saut inconditionnel

L'élève voit littéralement les signaux s'activer et le bus commuter, ce qui concrétise la micro-architecture.

## Public visé

- Enseignants en sciences de l'ingénieur (STI2D / SSI)
- Étudiants en électronique ou informatique
- Makers souhaitant comprendre l'intérieur d'un CPU
- Élèves débutants en logique numérique

## Pourquoi ce logiciel est unique ?

- Il combine éditeur visuel, simulateur logique, gestion des bus et micro-architecture complète.
- Il est basé sur Lazarus/FreePascal : libre, modifiable, pédagogiquement clair.
- Il offre une visualisation interne du micro-contrôleur, habituellement invisible.
- Sa logique interne simple mais cohérente est idéale pour la formation.

## Évolutions possibles

1. Breakpoints et debug instruction par instruction
2. Registres supplémentaires
3. ALU paramétrable
4. Jeu d'instructions étendu
5. Export en VHDL/Verilog
6. Interaction avec microcontrôleurs réels via liaison série

From:

<https://mistert.freeboxos.fr/dokuwiki/> - **Wiki de Sébastien TACK**



Permanent link:

[https://mistert.freeboxos.fr/dokuwiki/doku.php?id=logical\\_editor&rev=1765192156](https://mistert.freeboxos.fr/dokuwiki/doku.php?id=logical_editor&rev=1765192156)

Last update: **2025/12/08 11:09**