Atelier – Un simulateur logique pensé pour l'enseignement

Jean-Philippe Pellet[0000−0001−7559−397X] et Gabriel Parriaux[0000−0002−8921−5459]

Haute école pédagogique du canton de Vaud, Lausanne, Suisse
{jean-philippe.pellet,gabriel.parriaux}@hepl.ch

Résumé Une des plus-values d'un enseignement appliqué de l'informatique est de pouvoir amener les élèves à construire de petits systèmes informatiques avec lesquels on peut interagir. On pense par exemple à la programmation d'une interface graphique, d'un petit robot ou d'un système de carte électronique comme un Microbit ou Arduino. Dans cet atelier, nous souhaitons descendre d'un niveau dans les couches matérielles tout en restant dans le même état d'esprit en proposant un outil de simulation de portes logiques et d'autres composants informatiques de base. Nous montrons durant l'atelier comment ce simulateur se distingue d'autres solutions par ses possibilités pédagogiques bien appropriées à une approche PRIMM (*Predict-Run-Investigate-Modify-Make*), une extension du modèle bien connu *Use-Modify-Create*. Nous proposons dans l'atelier quelques scénarios pédagogiques adaptés à des élèves de début de lycée.

Keywords: informatique · systèmes logiques · simulation · PRIMM

1 Introduction

Nous semblons sortis de la période où l'informatique n'était enseignée à l'école que dans une perspective d'utilisation de logiciels [1]. Même s'il reste beaucoup de travail sur l'adaptation des programmes scolaires et sur la formation des enseignants, l'informatique est maintenant vue comme une science à part, bien plus riche que ses applications bureautiques.

Un pan de l'informatique encore souvent absent dans la plupart des cours qui nous sont connus traite de l'architecture de base des ordinateurs, ou plus précisément des systèmes logiques et de composants de base comme une unité arithmétique et logique (ALU) ou une bascule. Nous pensons qu'il y a quelques moments pédagogiques très intéressants à faire vivre au sein de cette thématique dès le lycée. En particulier, la réalisation de petits circuits montrant « *comment calculer avec de l'électricité* » ou « *comment stocker des données avec de l'électricité* », pour vulgariser la formulation, semble abordable avec des supports pédagogiques appropriés.

Nous pensons que l'inspection de telles situations problèmes est un moyen pour les apprenants de construire une meilleure machine notionnelle de comment fonctionne un ordinateur, d'expérimenter concrètement de petits systèmes « entrée–traitement–sortie », et de réaliser comment une abstraction matérielle progressive nous permet de finalement construire un minisystème complet.

2 Programme de l'atelier

Les participants à l'atelier viennent de préférence avec un laptop pour pouvoir essayer le simulateur. La prise en charge des interfaces tactiles (en particulier les tablettes) est encore partielle. Les petits écrans de type smartphone se prêtent mal à l'expérimentation.

Voici un plan grossier pour les 45 minutes de l'atelier :

Durée	Contenu
5 min	Introduction et approche pédagogique
5 min	Démo de l'interface
15 min	Activité 1 : développement d'un additionneur
15 min	Activité 2 : utilisation de bascules avec une ALU
5 min	Discussion et expérimentation libre

3 Contexte & approche pédagogique

On retrouve souvent les thématiques d'architecture des ordinateurs, des processeurs ou les systèmes logiques dans les études tertiaires. Obtenir le matériel nécessaire à grande échelle pour aborder ces thématiques est un problème en termes de coûts, de fiabilité et de flexibilité du matériel pédagogique. Nous avons exploré des options comme les logidules [5], qui semblent particulièrement adaptés, mais ne sont malheureusement plus en production. Nous avons ainsi cherché à utiliser, dans un premier temps en tout cas, un simulateur afin de faciliter l'utilisation en classe.

Plusieurs solutions existent ; nous en mentionnons deux en particulier, qui nous semblaient initialement prometteuses :

— Logisim [2] est un logiciel open source de simulation de circuits logiques régulièrement utilisé [4]. S'il est open source, Logisim a été créé il y plus de 20 ans, repose sur des technologies de son époque et n'est plus développé depuis 2014.

— Logicly [1], plus moderne et basé sur une solution web, a l'avantage de ne requérir aucune installation, mais fonctionne selon un modèle *freemium*, que nous ne jugions pas désirable en classe, même s'il a été mis en œuvre par d'autres dans des contextes comparables [6].

Nous n'avons pas été en mesure, pendant nos essais avec des logiciels existants, de développer une approche pédagogique à laquelle ces logiciels se prêteraient bien. Nous nous sommes donc lancés dans le développement de notre propre solution, basée sur un simulateur existant open source [2], que nous avons fini par presque intégralement réécrire et modifier. Le résultat est (pour l'instant) disponible à cette adresse : `https://logic.modulo-info.ch`.

1. `https://logic.ly`
2. `https://github.com/drendog/Logic-Circuit-Simulator`

PRIMM

Nous voulions explicitement que notre solution soit simple à mettre en œuvre via plusieurs supports ou plateformes que les enseignants seraient susceptibles d'utiliser, et puisse activement prendre en charge une approche de type *Predict-Run-Investigate-Modify-Make* (PRIMM) [7].

PRIMM peut être vu comme une extension du modèle bien connu *Use-Modify-Create* (UMC) [3]. Sentance et Waite expliquent que ces modèles découpent en plusieurs étapes bien définies l'approche pédagogique de ce que nous appellerons un *processus de synthèse structurée*. Les apprenants sont ainsi invités à d'abord (a) « faire tourner » des systèmes déjà réalisés ; (b) effectuer des modifications sur un système existant ; avant de (c) créer un système soi-même. Pour faire simple, la différence entre PRIMM et UMC est une spécification plus détaillée de la première étape (a) [7]. Là où UMC la considère comme une étape seule, qui représente l'U de UMC, PRIMM la décompose en les trois étapes *Predict*, *Run* et *Investigate*. Les étapes (b) et (c) peuvent être considérées comme identiques dans les deux approches.

Ces approches ont été d'abord développées avec en tête un contexte d'apprentissage de la programmation, mais le cas de la création de circuits logiques et de la compréhension du fonctionnement de briques de base à disposition est conceptuellement très proche. Il s'agit dans les deux cas de combiner de façon structurée des « primitives » dont le comportement est bien défini afin de résoudre un problème.

4 Caractéristiques du simulateur

Voici les caractéristiques du simulateur qui soutiennent une approche PRIMM et dont certaines sont exclusives à notre solution.

Tout d'abord, à toutes les étapes du processus PRIMM sauf la dernière (*Make*), il est essentiel que les enseignants puissent diffuser des schémas logiques partiels ou complets avec lesquels les apprenants pourront interagir ou qu'ils seront en mesure de modifier. Nous avons opté pour une solution web, qui ne requiert aucune installation. De plus, des paramètres d'URL dans les liens partagés peuvent contenir la description complète d'un schéma logique. Par exemple, ce lien donne directement un accès complet au schéma exemple représenté sur la Figure 1. Aucune base de données n'est requise et aucun stockage ne doit être effectué côté serveur ; aucun compte ou profil ne doit être créé, ni pour les enseignants, ni pour les apprenants. La solution est donc particulièrement légère à mettre en place en termes de gestion d'une classe numérique.

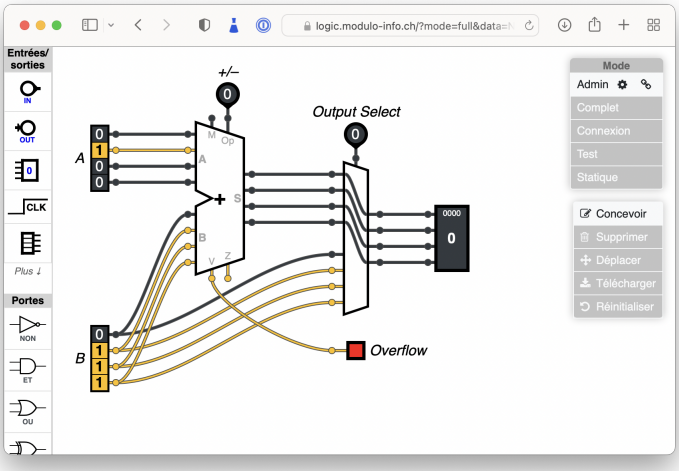

Figure 1. Circuit exemple mettant en œuvre une ALU et un multiplexeur.

Voici comment les étapes de PRIMM sont prises en compte.

— *Predict*. Des circuits peuvent être mis à disposition des apprenants en mode « inerte » pour qu'ils fassent leurs propres prédictions sur leur comportement sans utiliser le simulateur. Les entrées et sorties peuvent être affichées de manière neutre (sans révéler la valeur véhiculée).

— *Run*. Des circuits peuvent être diffusés en mode lecture seule, mais en laissant les apprenants changer les valeurs d'entrées pour observer le comportement (et éventuellement autovalider des hypothèses faites à l'étape *Predict*).

— *Investigate*. Des circuits peuvent être volontairement créés comme défectueux (fausse fonction logique réalisée ; sortie bloquée sur une valeur fixe ; temps de propagation trop élevé ; etc.) pour placer l'apprenant dans une position d'investigation de ce qui ne fonctionne pas correctement.

— *Modify* et *Make*. Les outils d'édition sont complètement disponibles sans rectrictions pour les apprenants, qui peuvent démarrer de zéro ou depuis un début de circuit partagé par l'enseignant en mode édition.

Références

1. Baron, G.L., Drot-Delange, B. : L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. Revue française de pédagogie. Recherches en éducation (195), 51–62 (2016)
2. Burch, C. : Logisim : A graphical system for logic circuit design and simulation. Journal on Educational Resources in Computing (JERIC) **2**(1), 5–16 (2002)
3. Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L. : Computational thinking for youth in practice. ACM Inroads **2**(1), 32–37 (2011)

4. Luković, V., Krneta, R., Vulović, A., Dimopoulos, C., Katzis, K., Meletiou-Mavrotheris, M. : Using Logisim educational software in learning digital circuits design. In : Proceedings of 3rd International Conference on Electrical, Electronic and Computing Engineering IcETRAN. pp. 5–1 (2016)
5. Nicoud, J.D. : Dedicated tools for microprocessor education. IEEE Micro **11**(1), 14–17 (1991)
6. Rueda, R.A.S., Silis, J.A.S. : Logic.ly simulator. Technological tool to facilitate the teaching-learning process about mathematics ? Dilemas Contemporáneos : Educación, Política y Valore (3) (2018)
7. Sentance, S., Waite, J. : Primm : Exploring pedagogical approaches for teaching text-based programming in school. In : Proceedings of the 12th Workshop on Primary and Secondary Computing Education. pp. 113–114 (2017)