Journée d'échanges de la SSIE

Un simulateur logique pour enseigner l'architecture des ordinateurs avec une approche de type *Use–Modify–Create*

Jean-Philippe Pellet

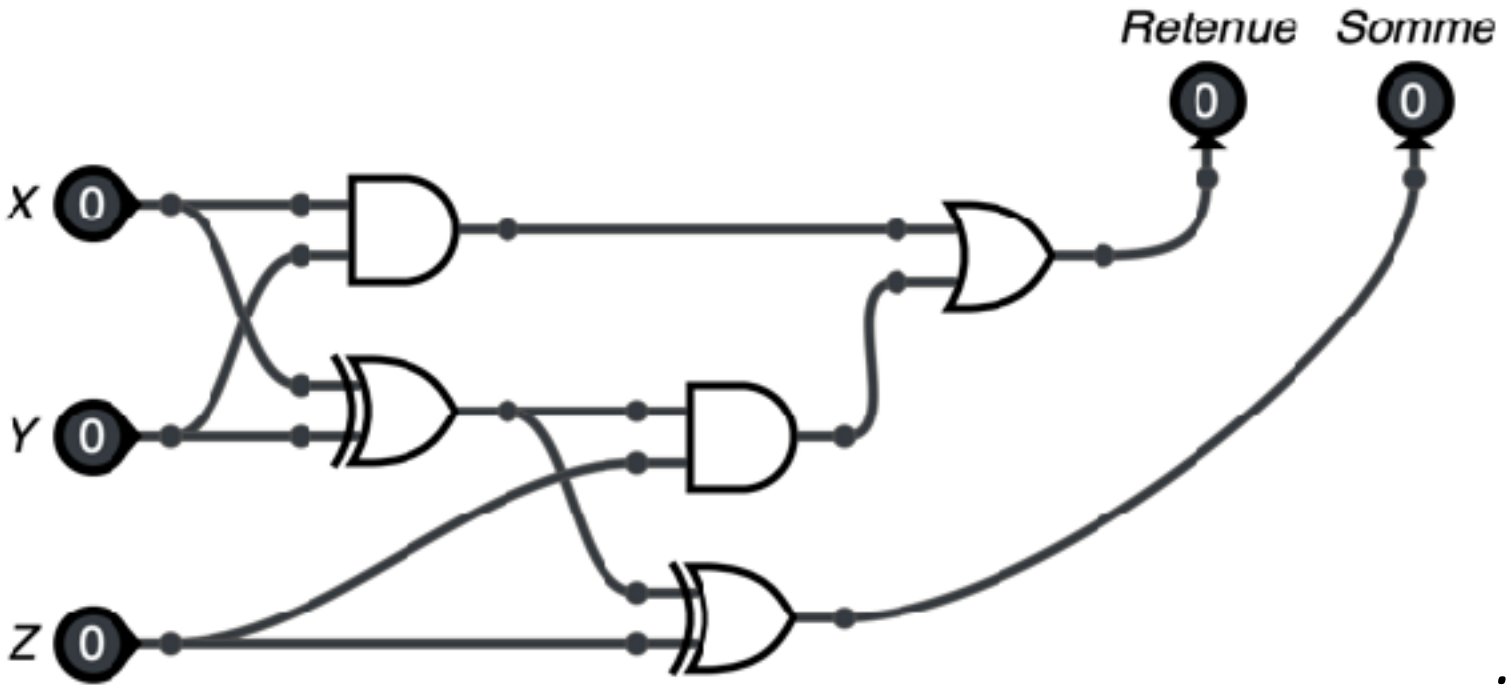
1er avril 2022

Un simulateur de systèmes logiques

- **Architecture des ordinateurs dans plan d'études vaudois**
  - Systèmes logiques, microprocesseurs, composants
  - Pas trop de théorie, exercices appliqués
  - ⇒ Systèmes logiques simples bien adaptés
- **Avantages de s'intéresser aux systèmes logiques**
  - Basé sur principes simples: ET, OU, etc.
  - Aspect «ingénieur» de construction d'un système qui fonctionne
- **Développé dans le cadre du projet Modulo**

Circuits logiques «simples»

- **Exemple de l'additionneur**



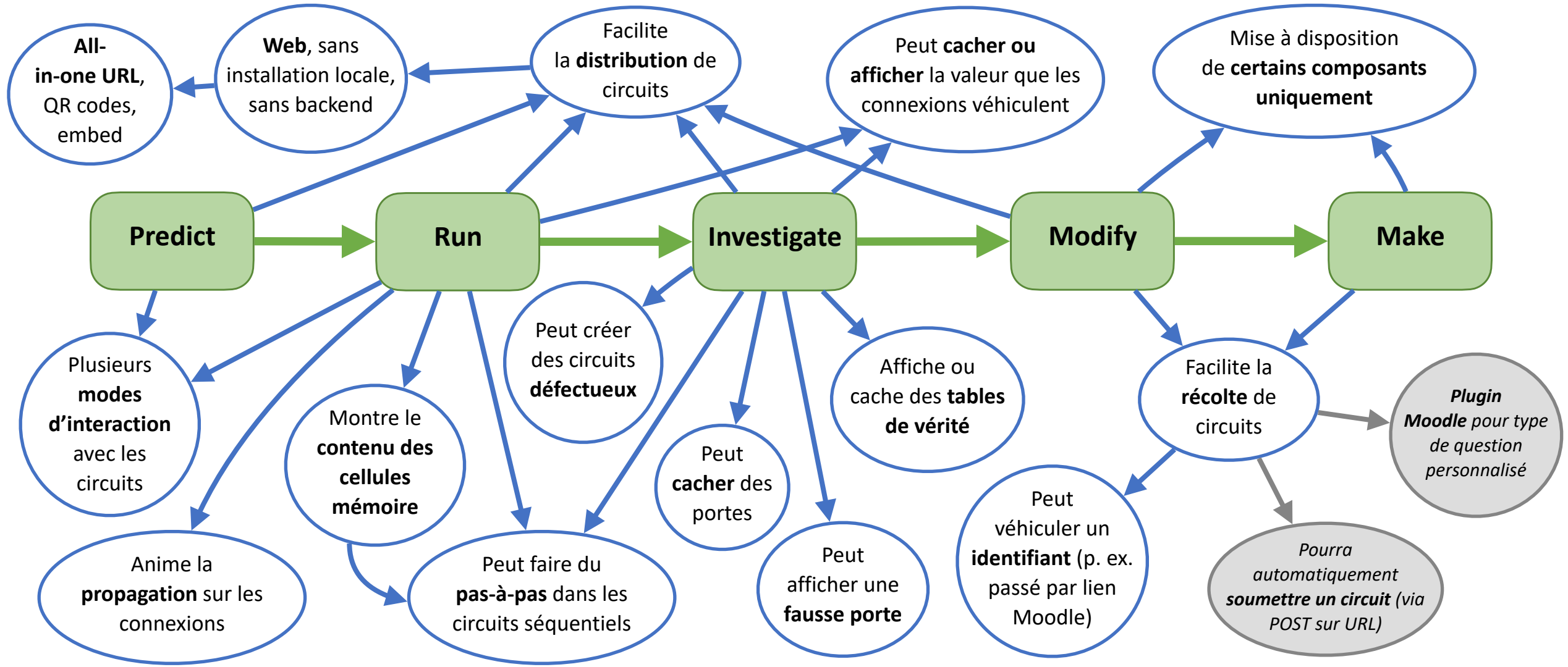
… illisible!

Use–Modify–Create et PRIMM

- **Idée de base de U–M–C**
  - Programmer à partir de zéro est difficile, alors on décompose l'approche
  - Pensé pour la création de programmes, mais se prête très bien à notre cas
- **PRIMM:** variante plus précise de U–M–C
  - *Predict:* sans jouer avec le système, dire ce qu'il fait
  - *Run:* faire tourner un système et vérifier les prédictions
  - *Investigate:* annoter, tracer, débugger un peu, nommer les variables, etc.
  - *Modify:* changer/compléter le code avec de petits puis moyens changements
  - *Make:* création d'un petit système complet

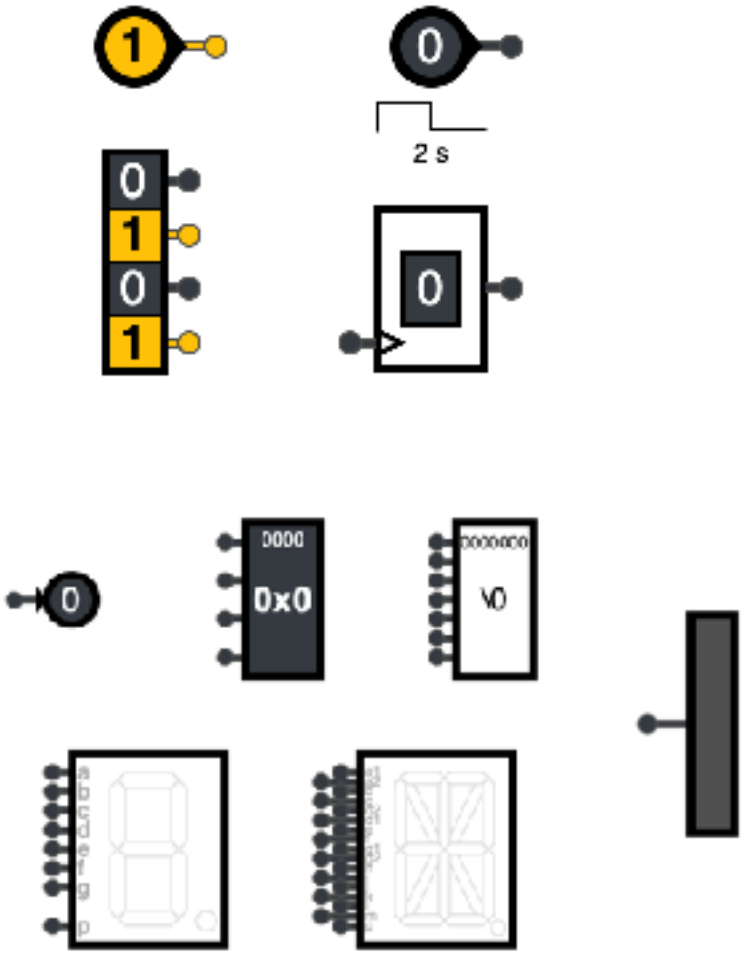
Sentance, S. & Waite, J. (2017). PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In: *Proceedings of the 12th Workshop in Primary and Secondary Computing Education: WIPSCE '17,* Nijmegen

PRIMM sur un simulateur logique — *démos*



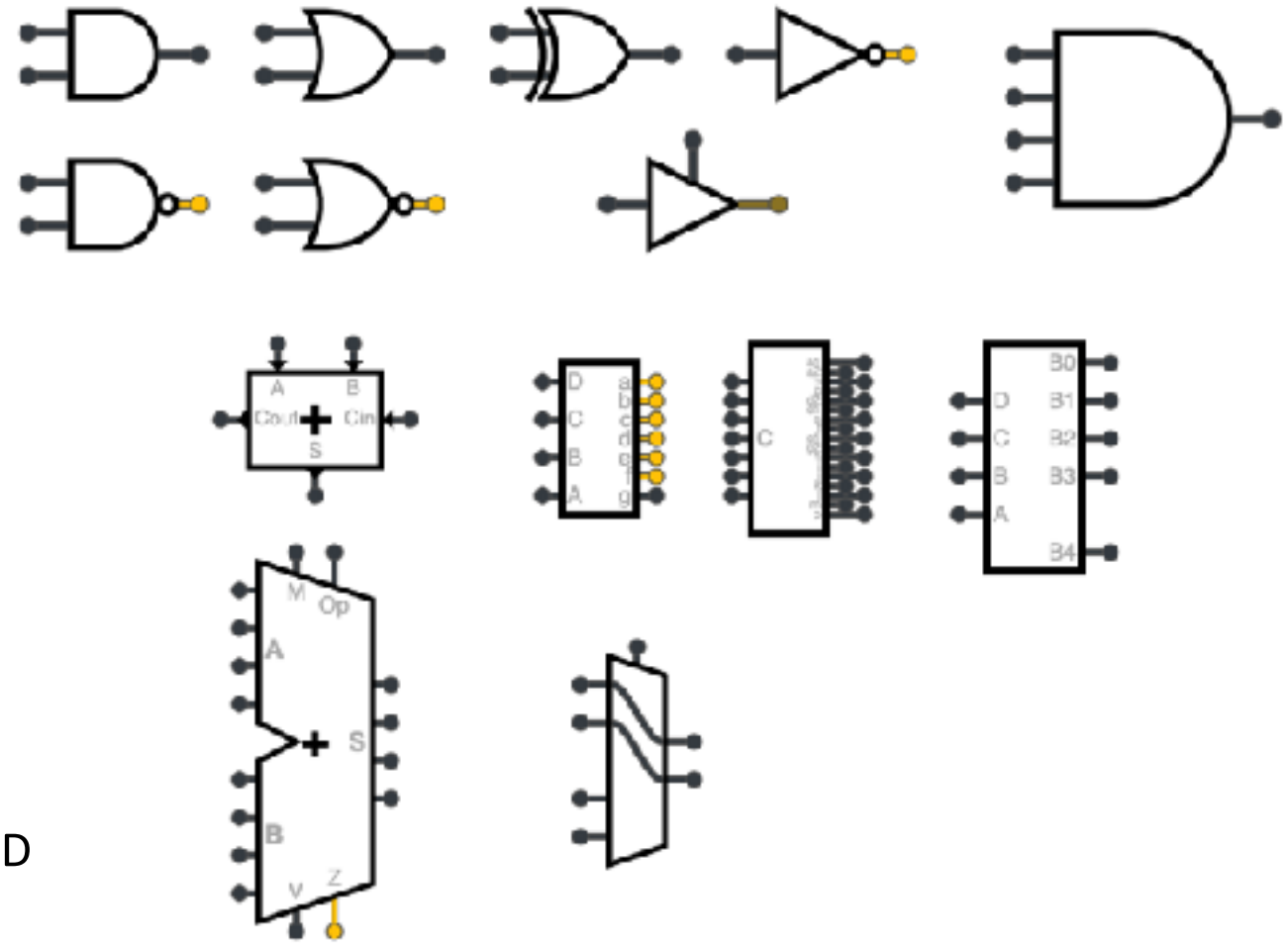
Composants: entrées/sorties

- **Entrées**:
  - 1 bit (poussoir, commutateur),
  - 4 bits (dip switches)
  - Horloge
  - Aléatoire (sur coup d'horloge)
- **Sorties:**
  - 1 bit (standard ou «bande lumineuse»)
  - 4 bits (hexa, décimal signé ou non)
  - 7 bits (ASCII)
  - Afficheur 7 segments, 16 segments



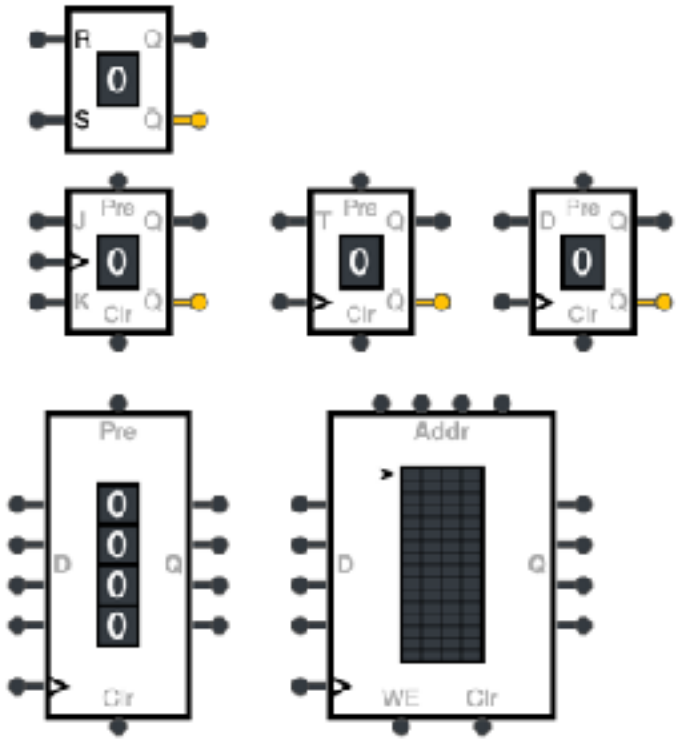
Composants: logique combinatoire

- **Portes**:
  - AND, OR, XOR, NOT
  - NAND, NOR
  - IMPLY, NIMPLY
  - Buffer à trois états (haute impédance)
  - 2, 3 ou 4 entrées
- **Composants combinatoires**
  - Additionneur
  - ALU (addition, soustraction, AND, OR)
  - Multiplexeurs (8/4/2 vers 4/2/1)
  - Décodeurs 7 segments, 16 segments, BCD



Composants: logique séquentielles

- **Composants combinatoires**
  - ✦ Verrou SR
  - ✦ Bascules: JK, T, D
  - ✦ Registre 4 bits
  - ✦ RAM de 16 × 4 bits
- **Pas encore disponible**
  - ✦ Compteur, registre à décalage, bus, ROM, composants personnalisés



À vous de jouer, *si le temps le permet — sinon, vos questions!*

1. Décodeur 2 bits

Depuis deux entrées représentant un nombre entre 0 et 3, créer quatre sorties dont toujours exactement une est active: celle dont l'index correspond au nombre représenté par l'entrée.

3. Mini-ALU

Construire une mini-ALU de 4 bits avec un bit de contrôle pour choisir entre addition et soustraction à l'aide de 4 additionneurs et de portes logiques supplémentaires.

2. Code de Hamming(7, 4)

À partir de 4 entrées de données D1 à D4, générer les 3 bits de parité P1 à P3 d'un code de Hamming(7, 4).

Bit #	1	2	3	4	5	6	7
Transmitted bit	p_1	p_2	d_1	p_3	d_2	d_3	d_4
p_1	Yes	No	Yes	No	Yes	No	Yes
p_2	No	Yes	Yes	No	No	Yes	Yes
p_3	No	No	No	Yes	Yes	Yes	Yes

4. Compteur

À l'aide de 4 bascules D, construire un compteur qui compte de 0 à 15 par incrément de 1 à chaque coup d'horloge.

Remerciements

- **drendog** — https://github.com/drendog
  - ✦ Projet initial forké puis réécrit ⇒ https://github.com/jppellet/Logic-Circuit-Simulator
- **Philippe Rochat**
  - ✦ Corédacteur du chapitre d'architecture des ordinateurs
- **Romain Edelmann, Nabil Dionisio, Ivan Moura, Raphael Holzer**
  - ✦ Tests en classe et suggestions
- **Jean-Daniel Nicoud**
  - ✦ Tests, discussion sur conventions, suggestions