Cours VHDL - IV

L3-S6 - Université de Cergy-Pontoise

Laurent Rodriguez – Benoît Miramond

Plan du cours

I – Historique de conception des circuits intégrés
- **HDL**
  - Modèles de conceptions
- **VHDL**
  - Les modèles de conceptions en VHDL
  - Les 5 briques de base

II – VHDL et FPGA
- **VHDL**
  - **Syntaxe et typage**
  - **Retour sur les briques de bases**
  - **Retour sur la conception structurelle et comportementale en VHDL**
    Port map, Équations logiques, Tables de vérités (With ... Select)
- **FPGA**
  - **Qu'est ce qu'un FPGA**
  - **Flot de conception**
  - **Carte de développement et environnement de TP**

Plan du cours

III – Modélisation

- Compléments sur la description Structurelle

- Description comportementale approfondie

Circuits combinatoires

- Styles de description

Flots de données

Instructions concurrentes

Table de vérité

Fonctions

- Circuits standards

- Comparateur
- Multiplexeurs
- Encodeurs

Plan du cours

IV – Processus et gestion du temps
- **Syntaxe et sémantique**
- **Portée - signaux & variables**
- **Annotations temporelles et délais**

V – Simulation
- **TestBench**
- **GHDL & GTK-Waves**
- **Quartus & ModelSIM**

VI – Modèles génériques
- **Paramètres génériques**
- **« Generate »**
- **Boucles**
- **Exemples**

Plan du cours

IV – Processus et gestion du temps
- **Syntaxe et sémantique**
- **Portée - signaux & variables**
- **Annotations temporelles et délais**

V – Simulation
- **TestBench**
- **GHDL & GTK-Waves**
- **Quartus & ModelSIM**

VI – Modèles génériques
- **Paramètres génériques**
- **« Generate »**
- **Boucles**
- **Exemples**

Plan du cours

VIII – Machines à états finis (FSM)

- **Définition**
- **Représentation**
  - **Formelle**
  - **Table de transition**
  - **Diagramme**
- **Type de machines**
- **Exemples**
  - **Multiplieur à machine à états**
- **Notions importantes**
  - **Reset**
  - **Représentation des états**

Plan du cours

VIII – Processeur 16bits en VHDL (projet)
- Présentation du projet et consignes (avancées TP)
- Retour sur les entités de base – Le registre
- Description à haut niveau
  - Interface du processeur (ENTITY)
  - Structure parties Contrôle/Opérative (retour sur FSMs)
  - Jeu d'instructions
- Description de l'architecture
  - Le chemin de données
  - Signaux de contrôles (MUX, R_SET, ALU_CMD, ...)
  - Contrôleur (FSM et gestion des signaux de contrôles)
- Séquencement des instructions
  - Exemple du séquencement par l'architecture
    - MVI
    - Add
- Utilisation de la mémoire
  - Mémoire instructions

IX – Révisions/Contrôle des connaissances

VIII - Processeur 16bits en VHDL

(projet)

VIII - Processeur 16bits : Présentation et consignes

4 Séances de Tps restantes

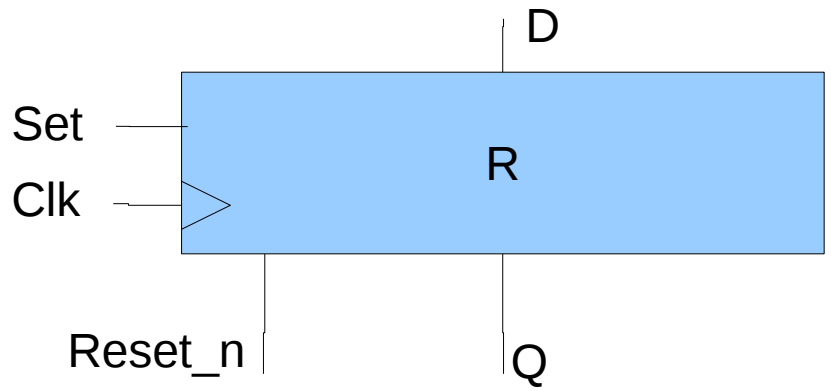
- 0.5 => Additionneurs + Multiplieur génériques (A rattraper hors séances sinon!)
- 1,5 => FSM
- 2 => Projet

Composants

- TP1 : Multiplexeurs, Décodeurs, Afficheurs (7 segments)
- TP2 : Registres, Additionneur, /a², Soustraction, ALU, Accumulateur complet
- TP3 : Multiplicateur + Maj de l'ALU (ajout Mult) + maj accumulateur
- TP4 : Module FSM + génération de signaux de contrôle (sortie FSM)
- Projet : Processeur : assemblage structurel des composants développés

VIII - Processeur 16bits : Retour sur les entité de base : Le registre

Modèle du registre utilisé pour le projet :



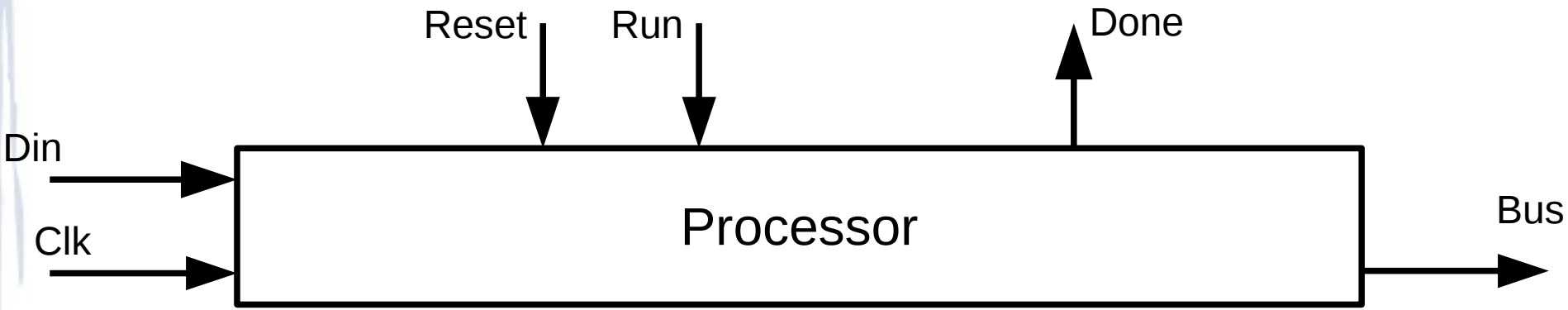
Propriétés :

- Reset asynchrone
- Set synchrone
  - Mise à jour de la valeur de Q seulement sur front montant et si Set est à 1

VIII - Processeur 16bits : Description haut niveau / Entity

IO :

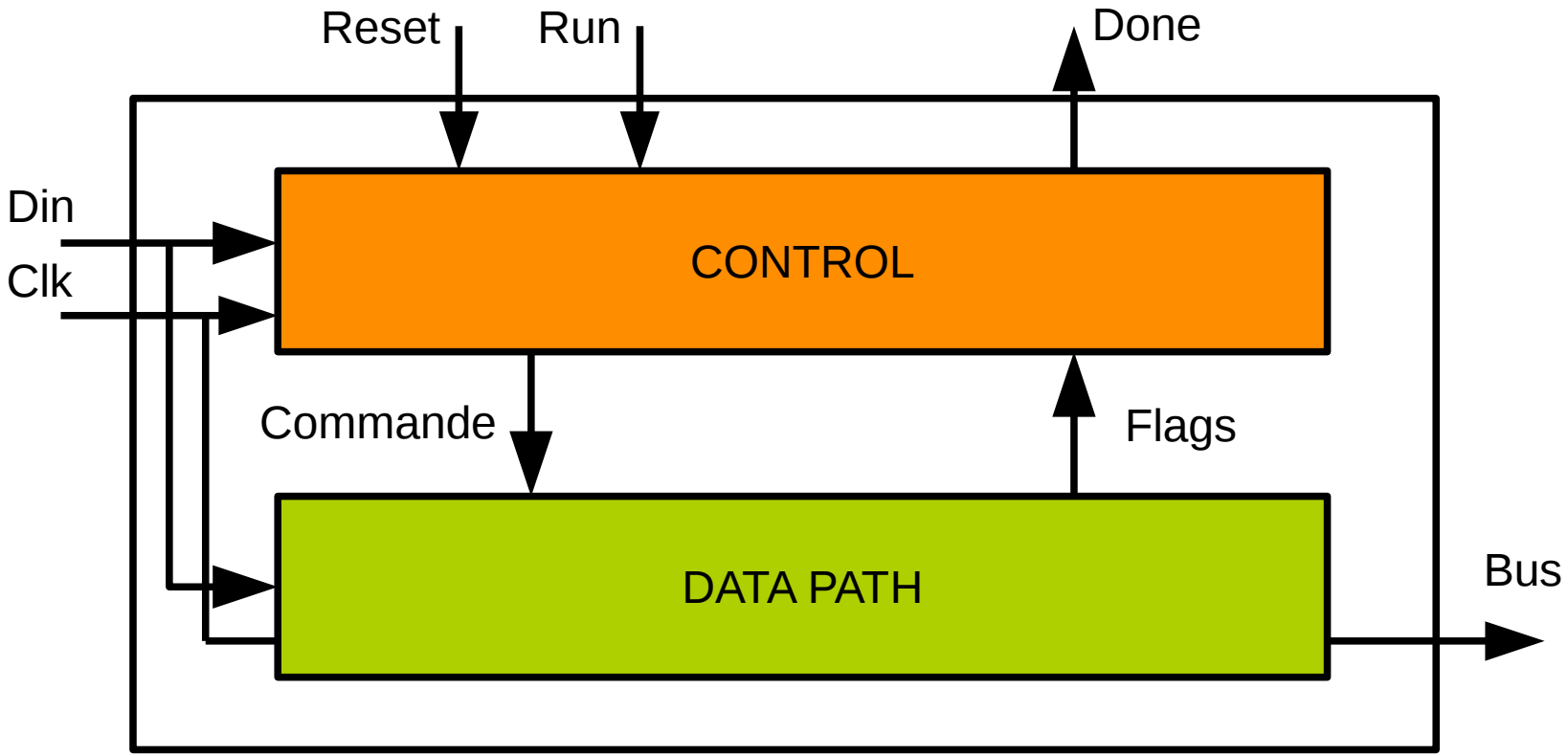
- **Din** : port de 16 bits dont les valeurs seront dans un premier temps positionnées par le programme (puis par la mémoire). Il permet d'acheminer dans le processeur :
  - Soit une instruction vers le contrôleur
  - Soit un entier vers le data path
- **Run** : Signal de synchronisation activé par le programmeur pour indiquer qu'une nouvelle valeur Din est disponible
- **Reset_n** : est le signal permettant la mise à zéro (ou valeurs par défaut) de toutes les bascules (registres) du processeur
- **Clock** : Horloge sur laquelle est cadencé le processeur
- **Bus** : port de 16 bits indiquant la donnée présente sur le Data Path (sur Done : Résultat ou Sortie de l'ALU)
- **Done** : Signal de synchronisation activé par le processeur pour indiqué que la dernière instruction a été exécutée.



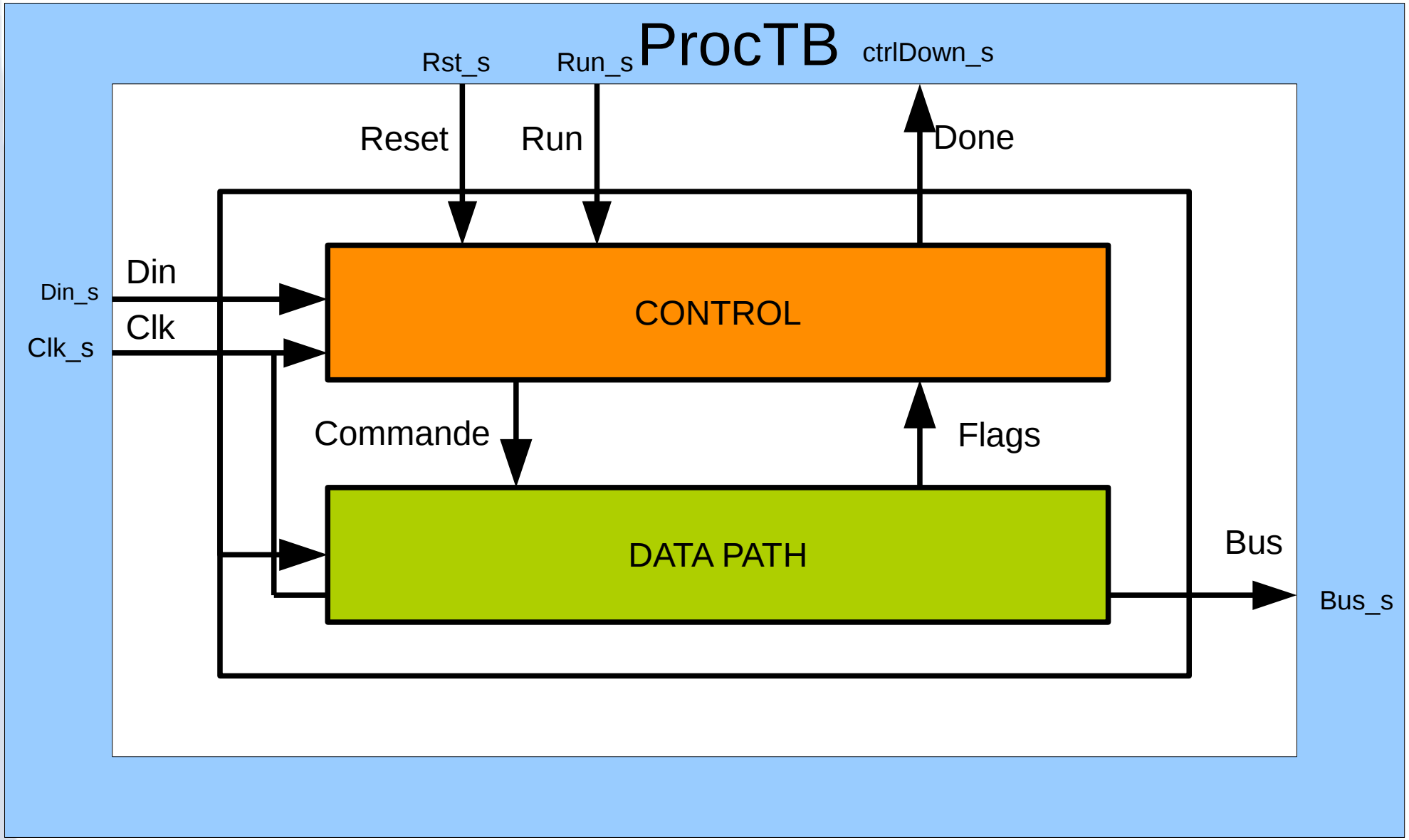
VIII - Processeur 16bits : Description haut niveau

2 parties :

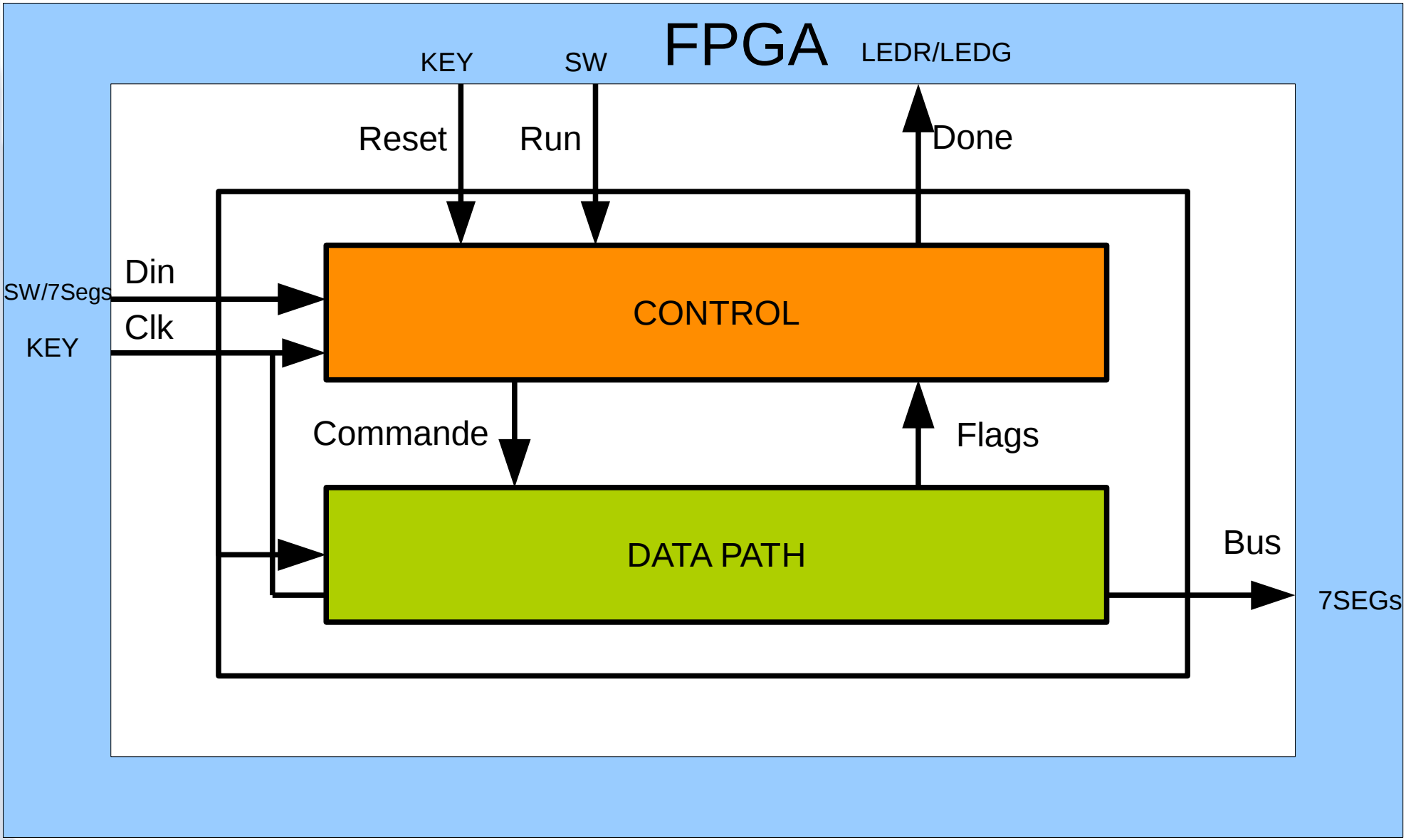
- **Le chemin de données** qui réalise des calculs sur des données de 16bits
  - 0 à 2^(16)-1 (65535) en non signé
  - -2^(16-1) à 2^(16-1)-1 en [-32768, 32767] signé /a²
- **Le contrôleur** qui décode l'instruction courante et pilote le chemin de données pour l'exécuter



VIII - Processeur 16bits : Description haut niveau



VIII - Processeur 16bits : Description haut niveau



VIII - Processeur 16bits : Jeu d'instructions

Le processeur aura un jeu d'instruction extensible, partant d'une première version permettant principalement l'addition/soustraction d'opérandes entières contenue dans des registres de données R0 à R7.

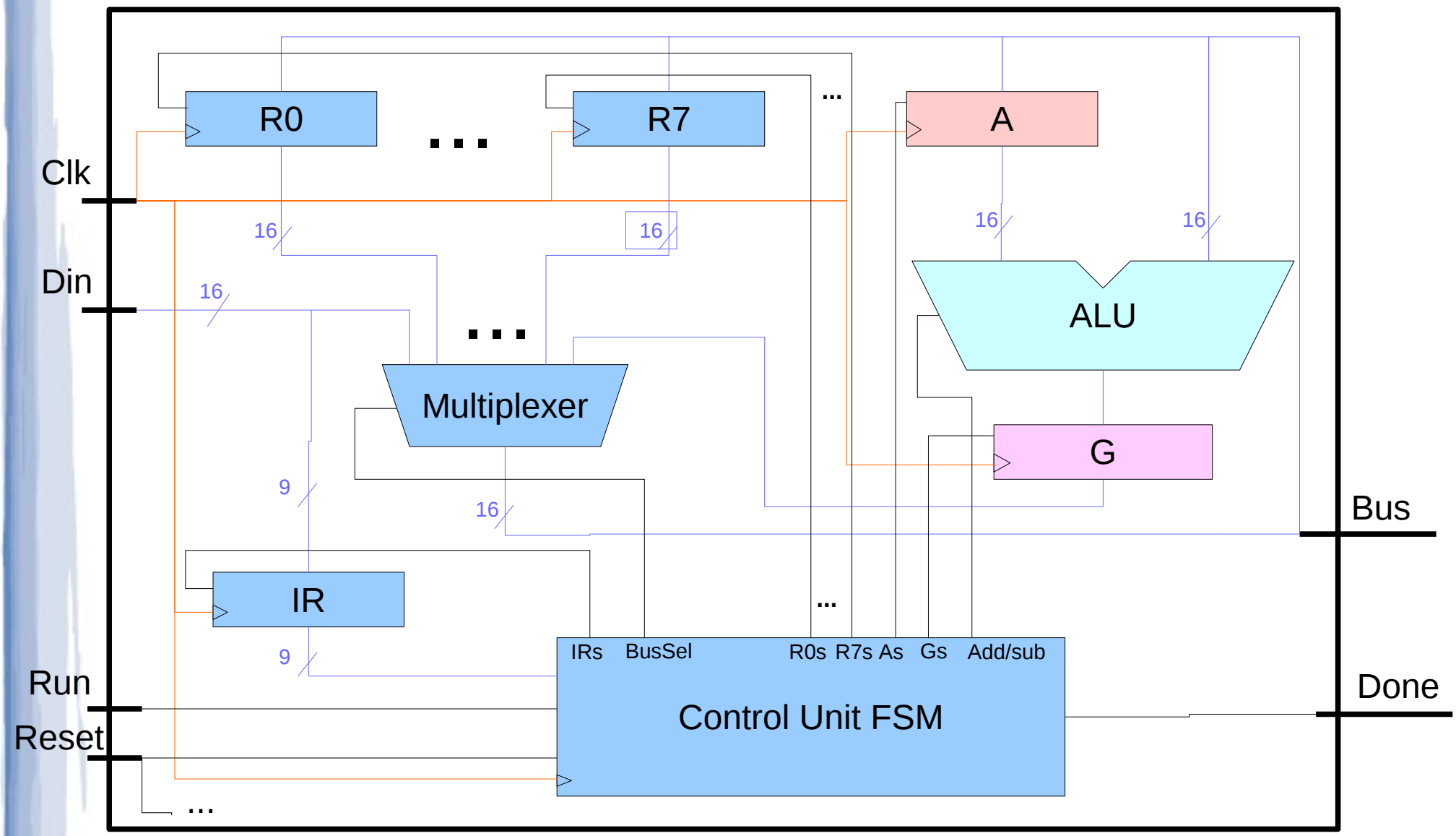
Instructions :

Instruction	Fonction réalisée
MV Rx, Ry	Rx <= [Ry]
MVI Rx, #D	Rx <= D
ADD Rx, Ry	Rx <= [Rx] + [Ry]
SUB Rx, Ry	Rx <= [Rx] - [Ry]

Registre instruction (RI) :

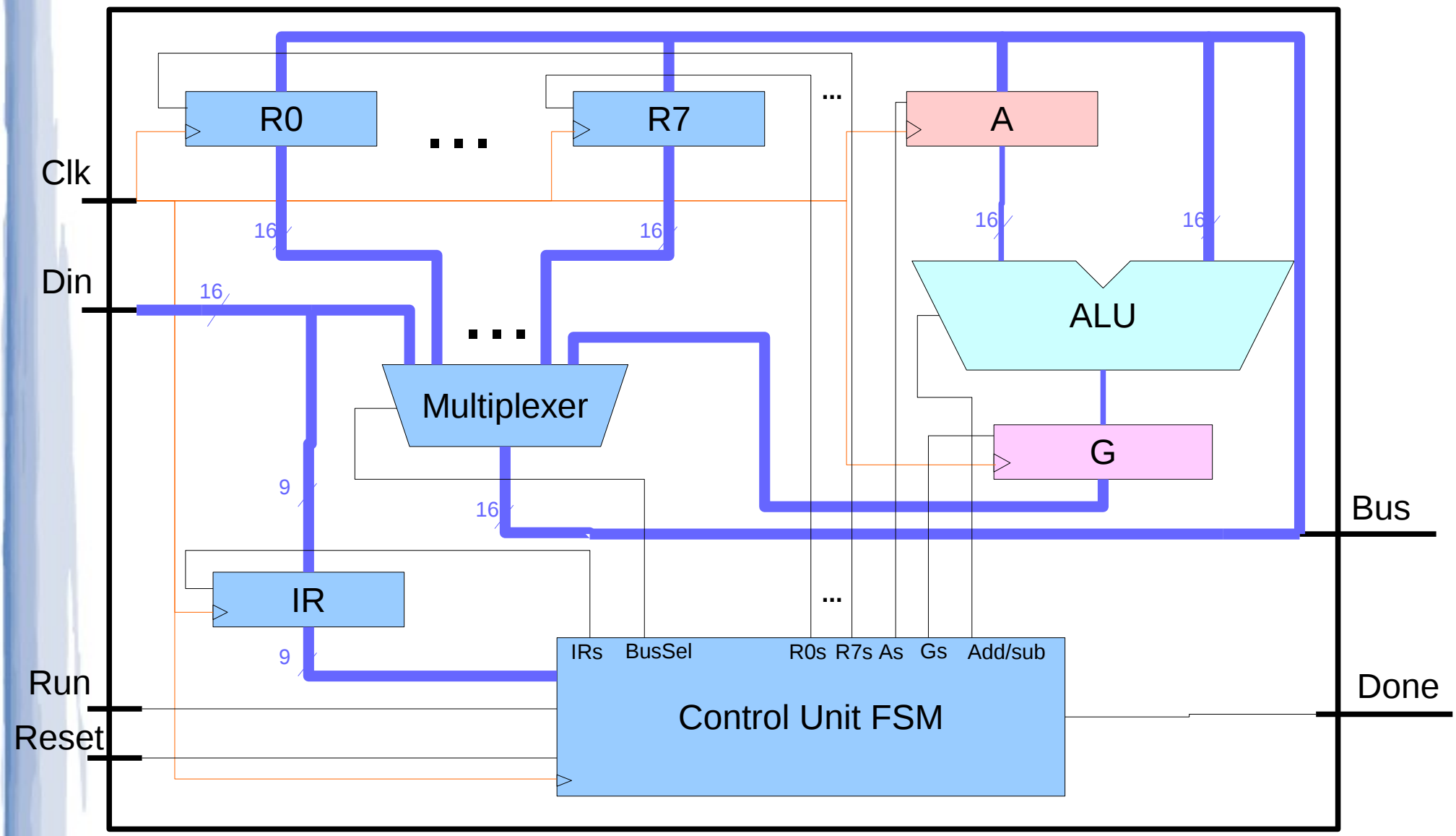
3bits	3bits	3bits
CODOP	Rx	Ry

VIII - Processeur 16bits : Description bas niveau

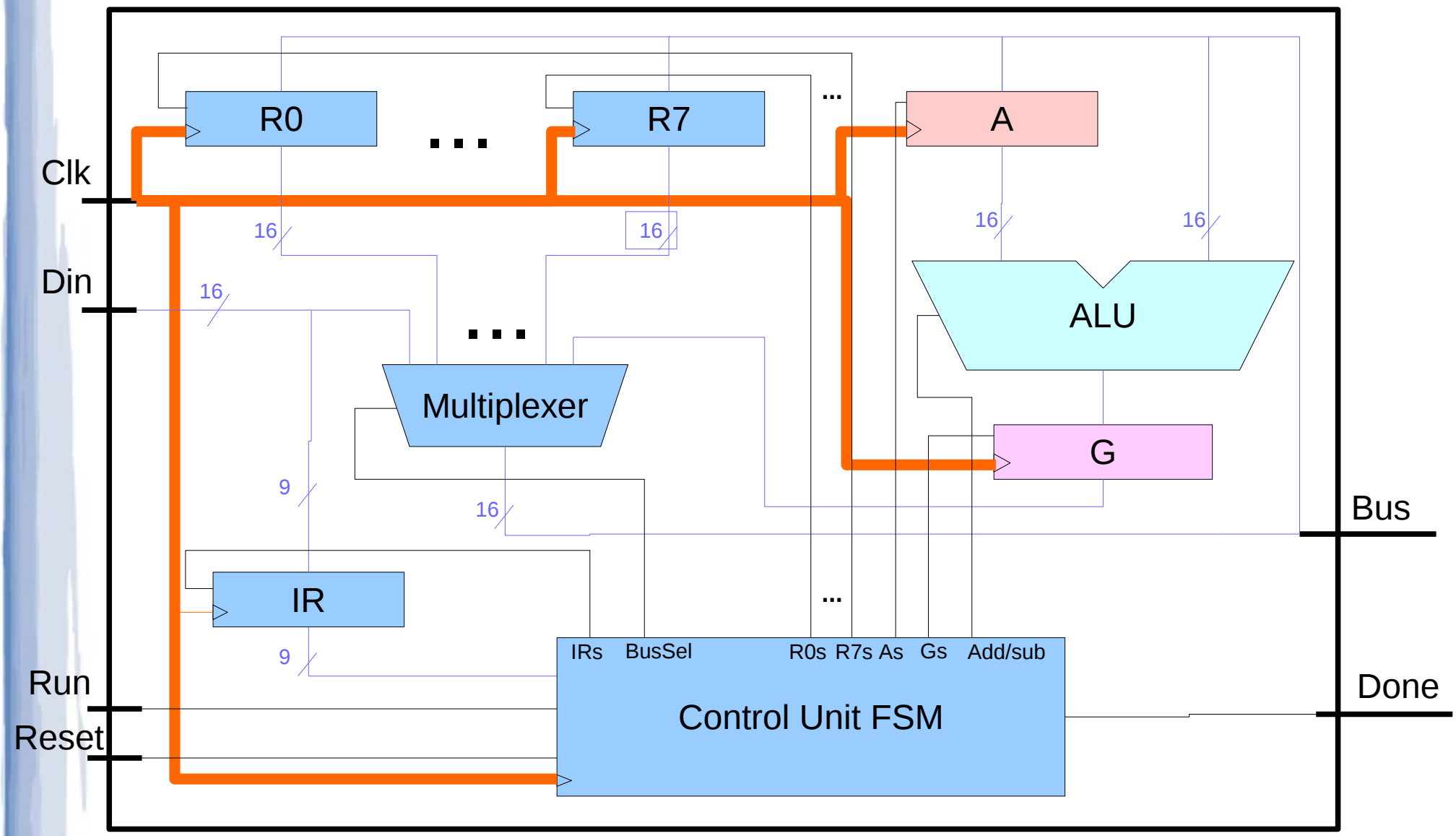


VIII - Processeur 16bits : Description bas niveau

DataPath

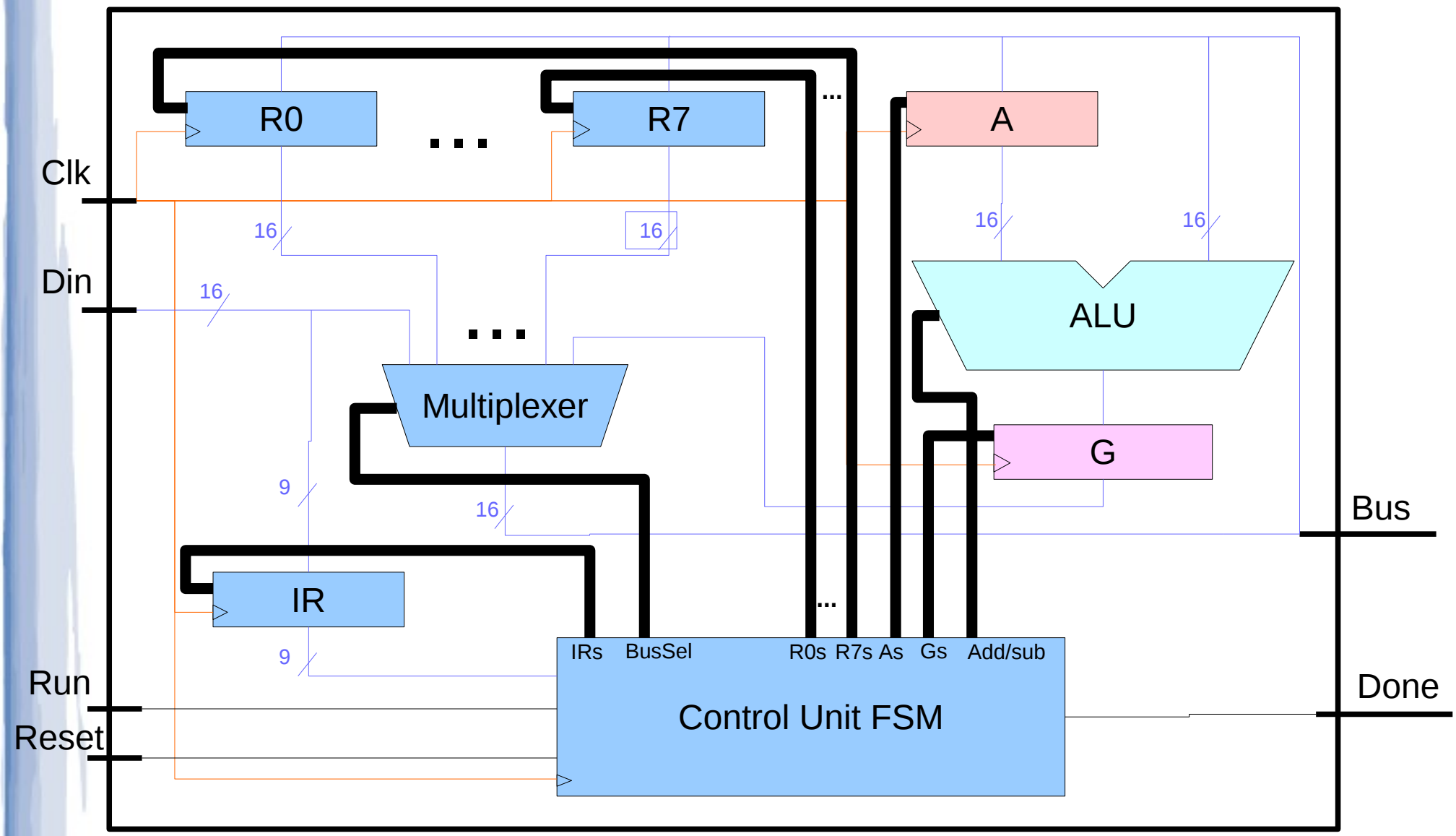


VIII - Processeur 16bits : Description bas niveau



VIII - Processeur 16bits : Description bas niveau

Contrôle



VIII - Processeur 16bits : Jeu d'instructions

Séquencement des instructions (définition de la FSM) :

Instruction	T1	T2	T3
MV Rx, Ry	Lecture de Ry Ecriture sur Rx		
MVI Rx, #D	Lecture de Di Ecriture sur Rx		
ADD Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx
SUB Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx

Certaines instruction prennent plus d'un cycle d'horloge. Plusieurs déplacements de données sont nécessaires pour les réaliser.

VIII - Processeur 16bits : Jeu d'instructions

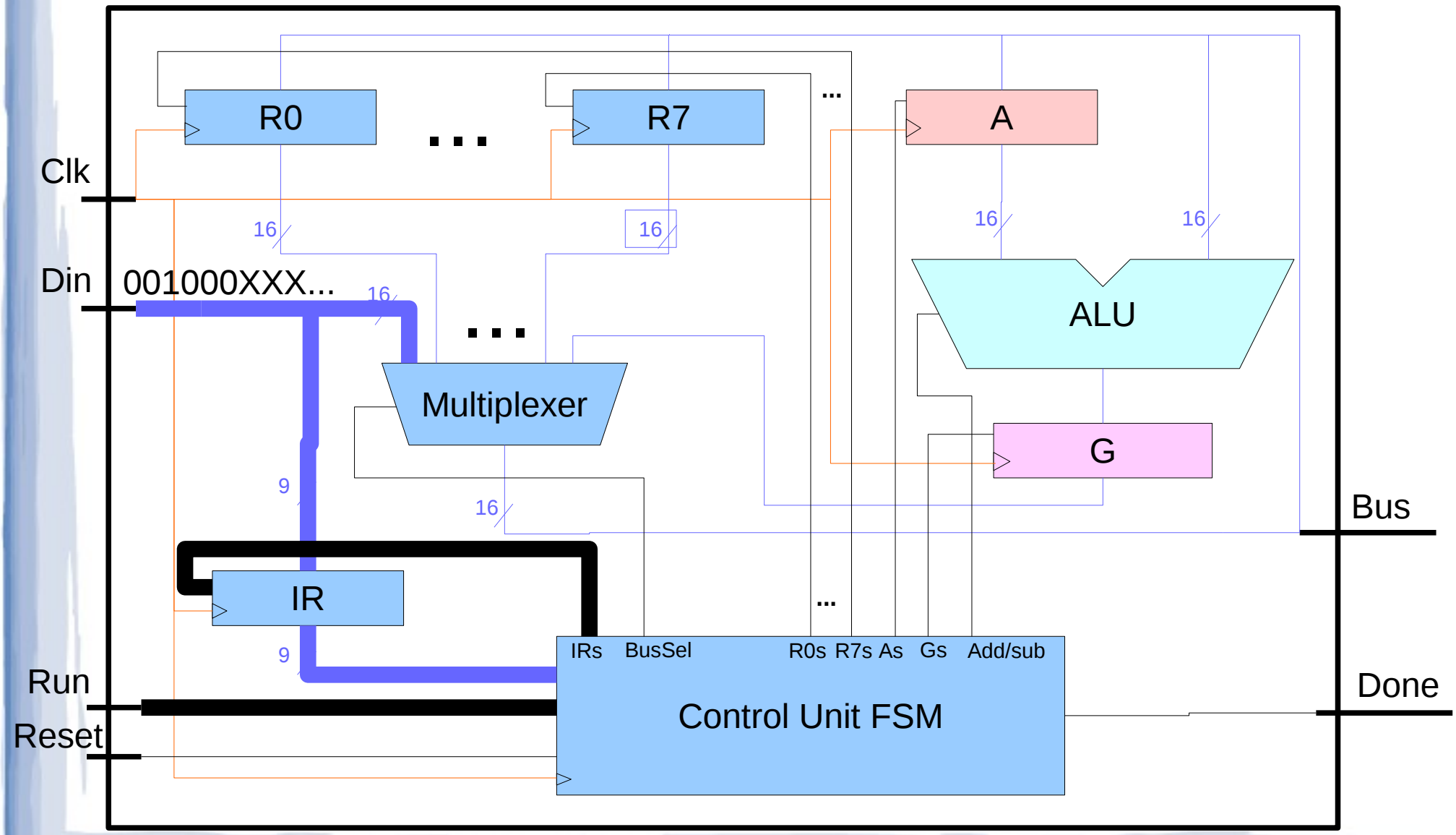
Séquencement des instructions (définition de la FSM) :

Instruction	T1	T2	T3
MV Rx, Ry	Lecture de Ry Ecriture sur Rx		
MVI Rx, #D	Lecture de Di Ecriture sur Rx		
ADD Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx
SUB Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx

Certaines instruction prennent plus d'un cycle d'horloge. Plusieurs déplacements de données sont nécessaires pour les réaliser.

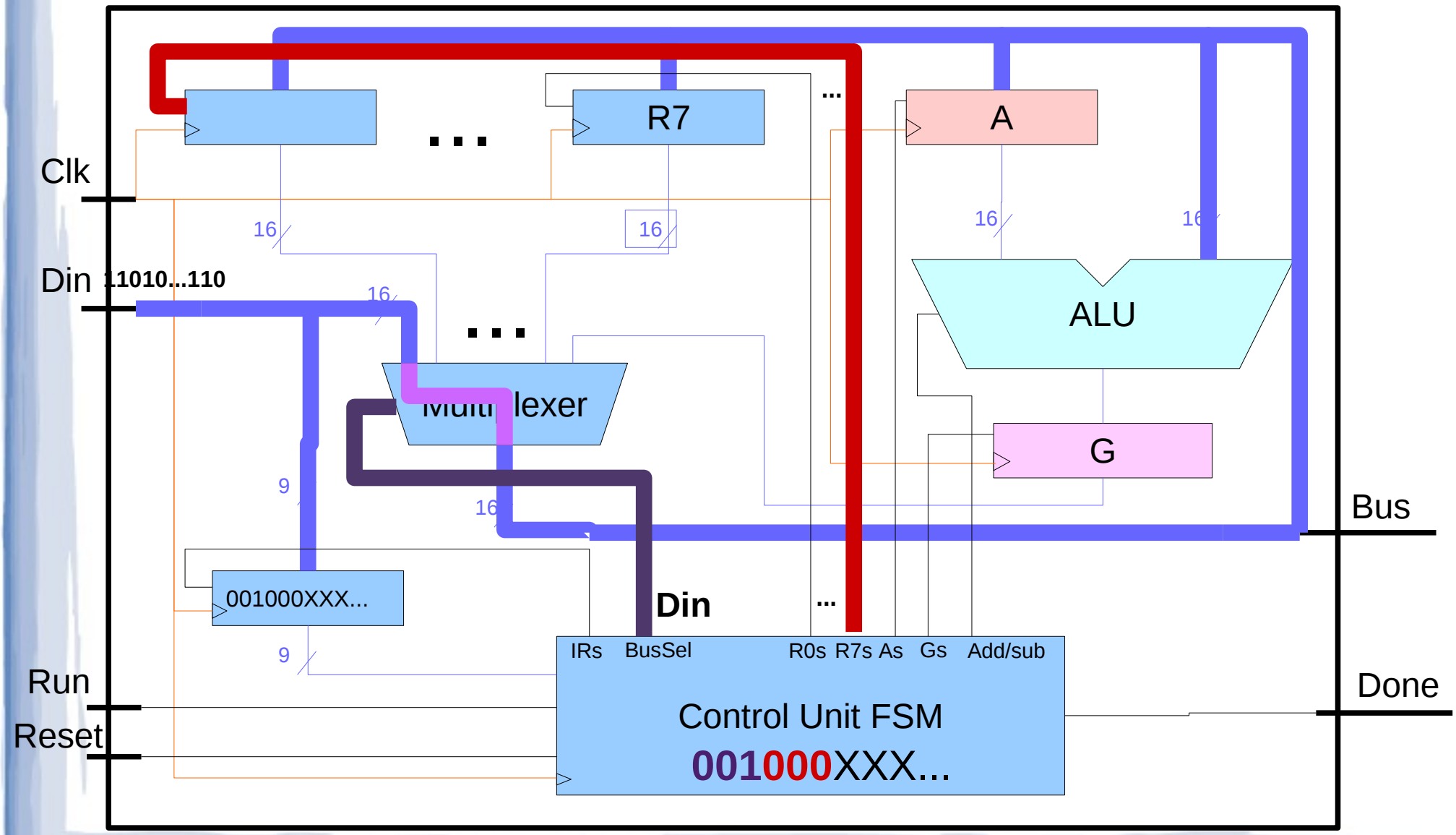
VIII - Processeur 16bits : Exemple d'évolution des signaux MVI

MVI R0, #D	Lecture de Di Ecriture sur Rx



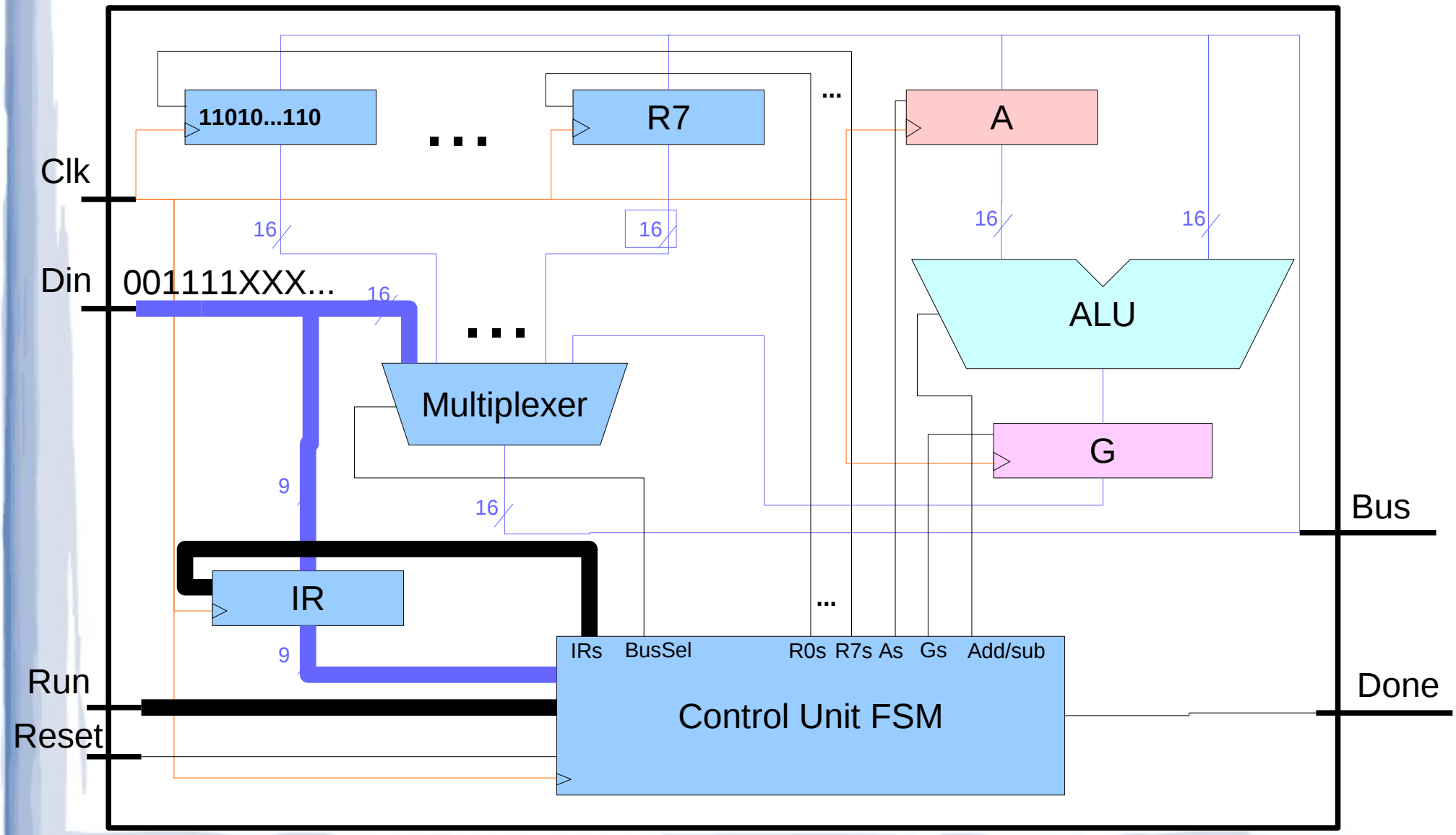
VIII - Processeur 16bits : Exemple d'évolution des signaux MVI

MVI R0, #D	**Lecture de Di** **Ecriture sur Rx**



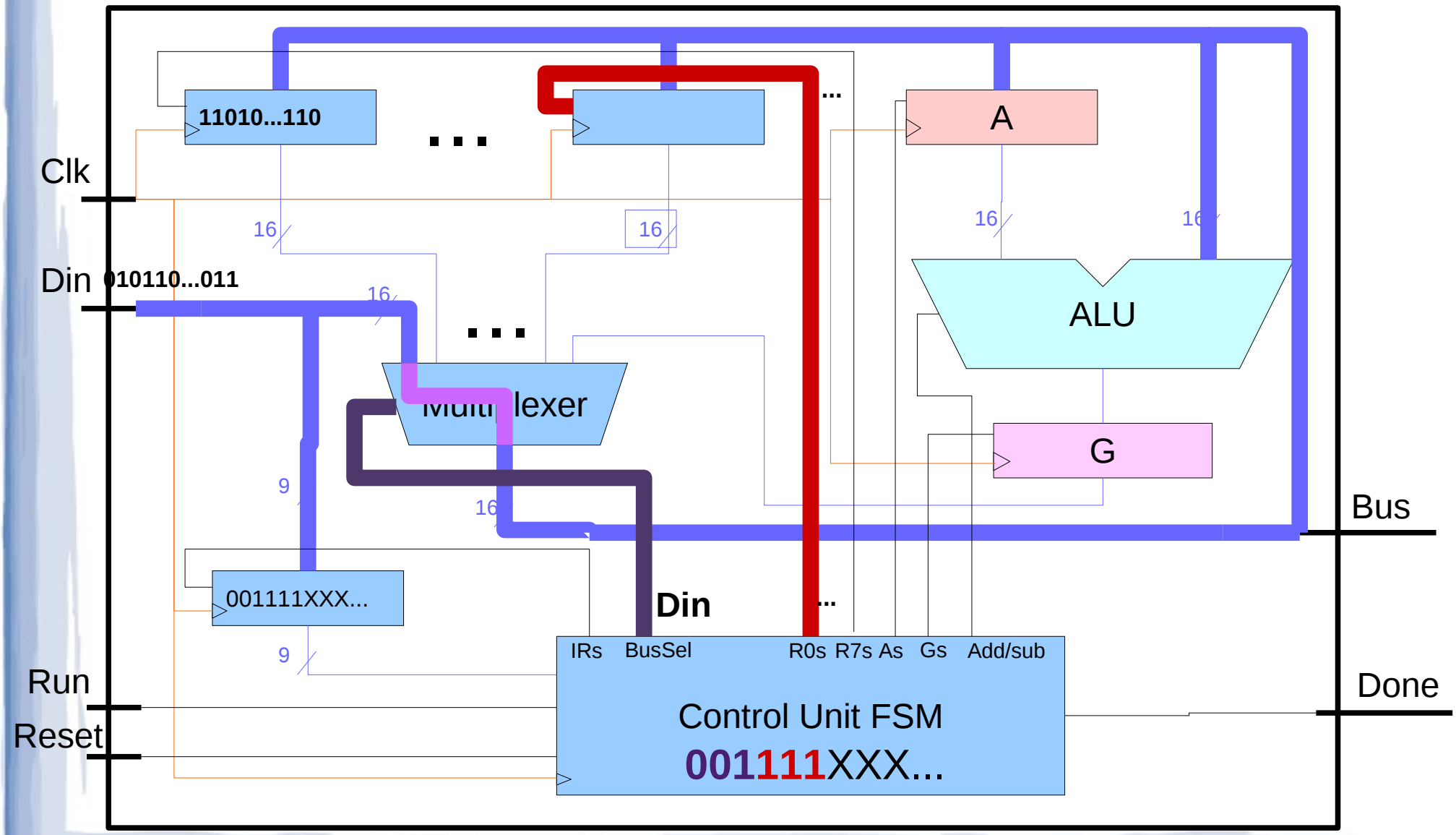
VIII - Processeur 16bits : Exemple d'évolution des signaux MVI

MVI R7, #D	Lecture de Di Ecriture sur Rx



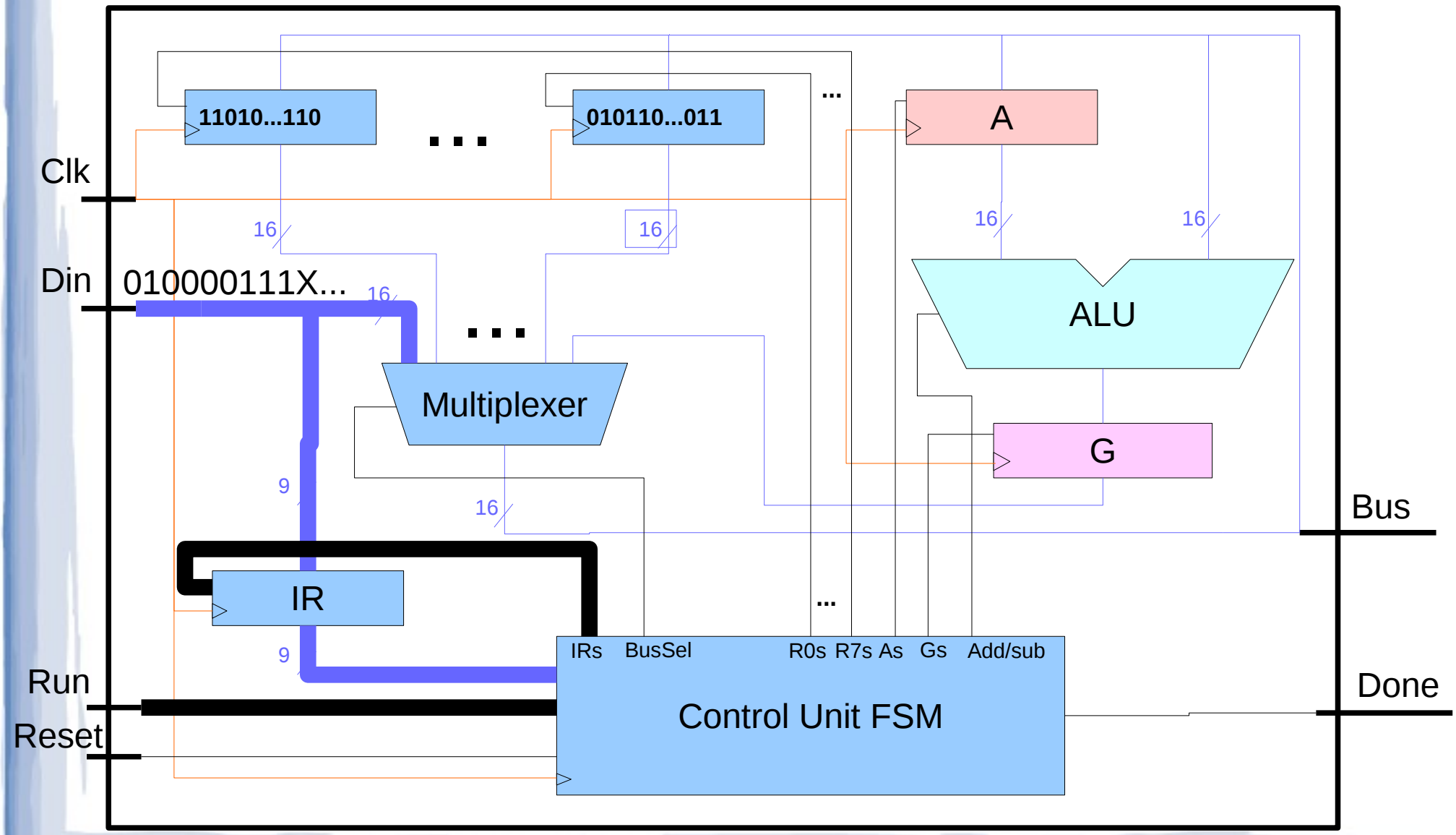
VIII - Processeur 16bits : Exemple d'évolution des signaux MVI

MVI R0, #D	**Lecture de Di** **Ecriture sur Rx**



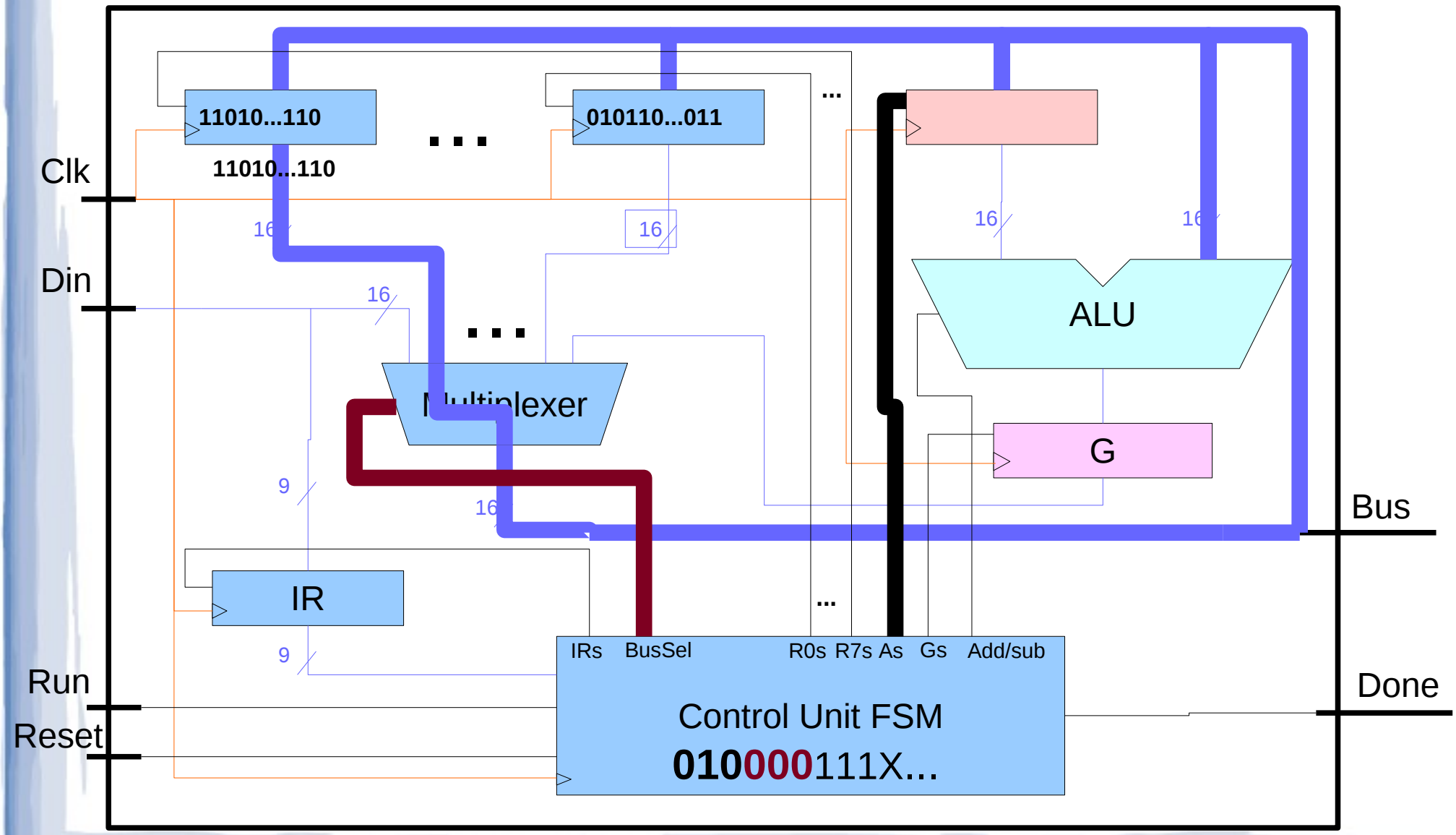
VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx



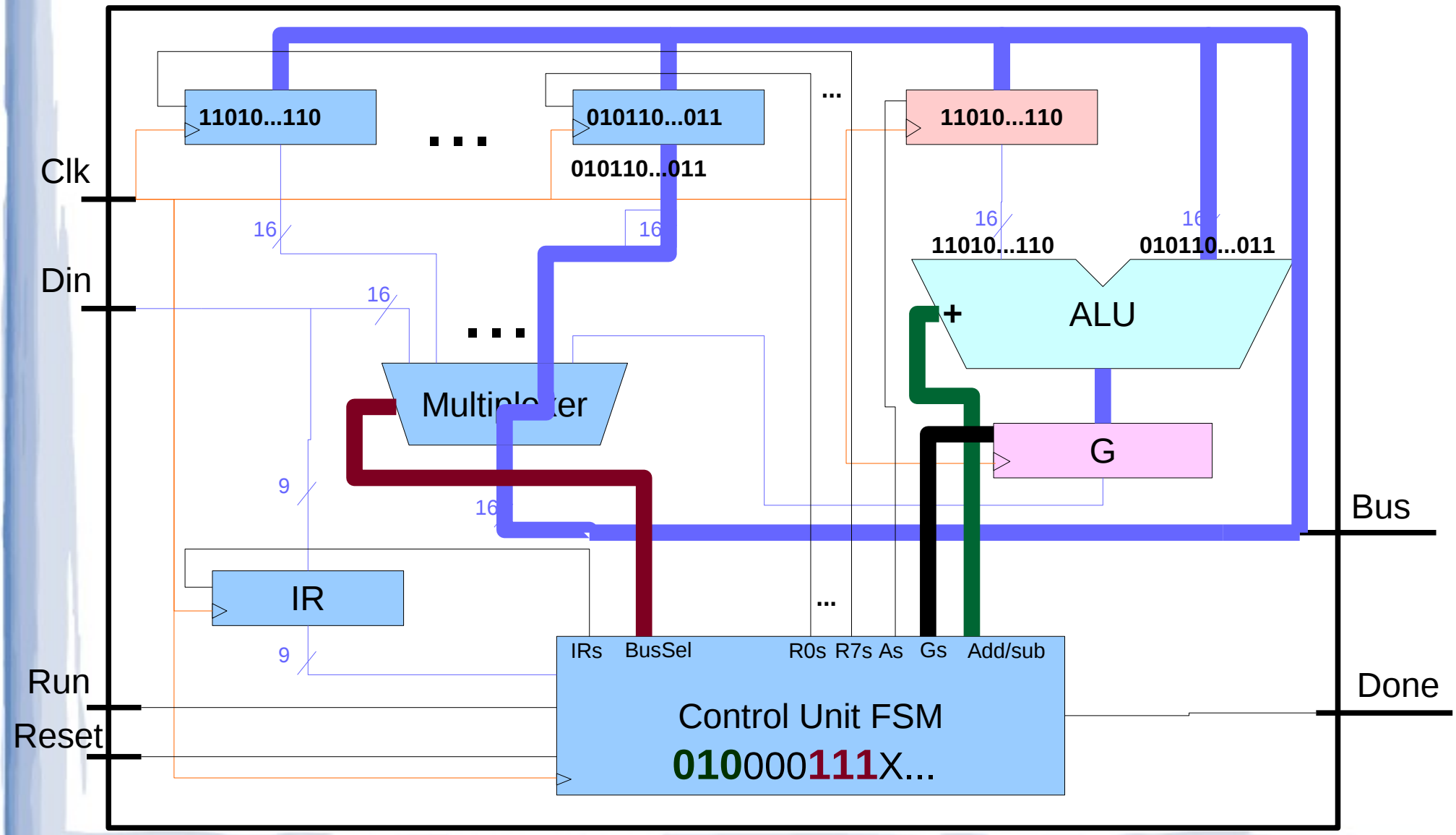
VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	**Lecture de Rx** **Ecriture dans A**	Lecture de Ry Ecriture de G	Lecture du rés (G) Ecriture sur Rx



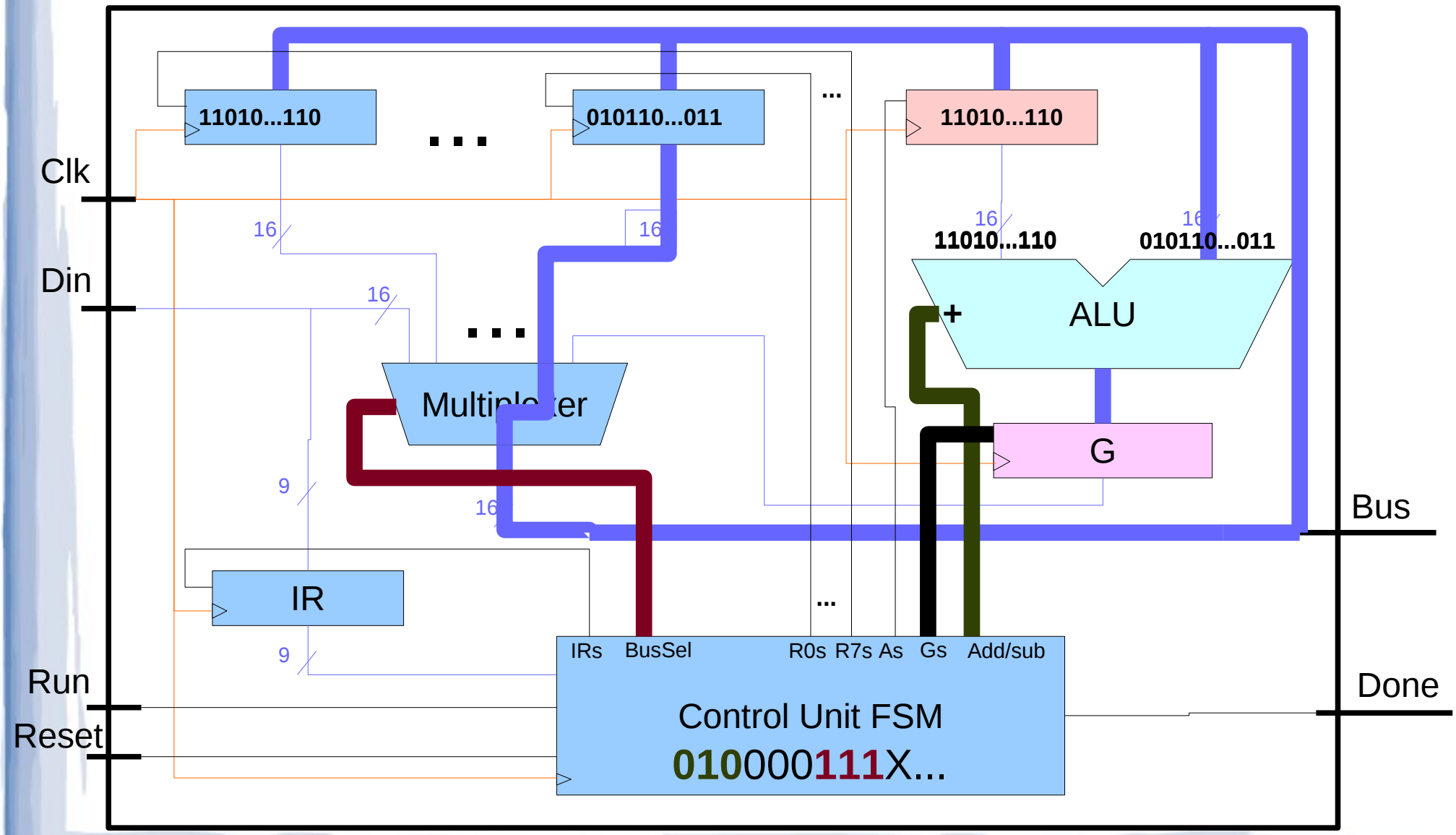
VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	Lecture de Rx Ecriture dans A	**Lecture de Ry Ecriture de G**	Lecture du rés (G) Ecriture sur Rx



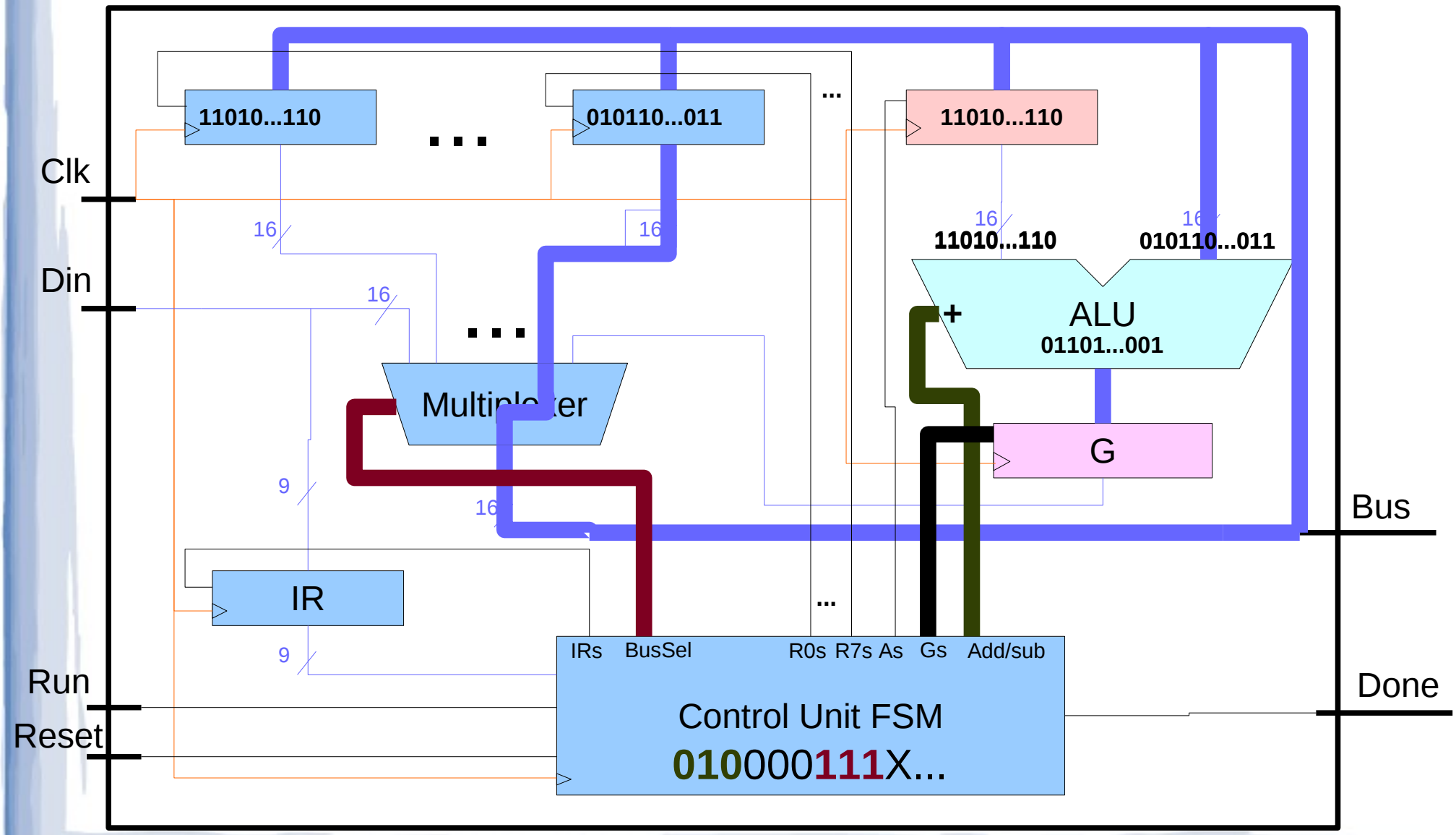
VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	Lecture de Rx Ecriture dans A	**Lecture de Ry Ecriture de G**	Lecture du rés (G) Ecriture sur Rx



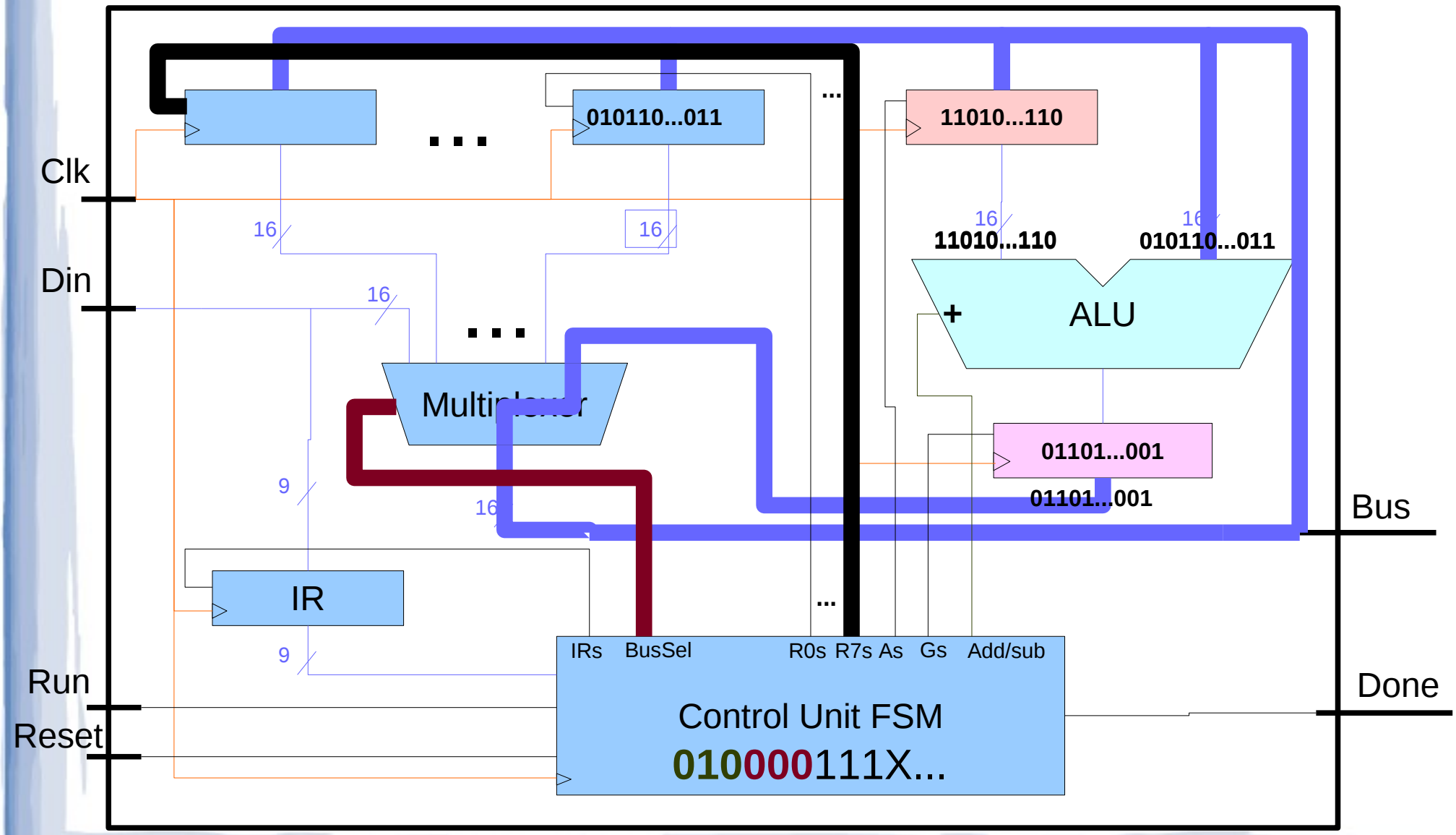
VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	Lecture de Rx Ecriture dans A	**Lecture de Ry Ecriture de G**	Lecture du rés (G) Ecriture sur Rx



VIII - Processeur 16bits : Exemple d'évolution des signaux ADD

ADD Rx, Ry	Lecture de Rx Ecriture dans A	Lecture de Ry Ecriture de G	**Lecture du rés (G)** **Ecriture sur Rx**



VIII - Processeur 16bits : Le Gros du travail !

Une fois l'ensemble des briques de bases réalisée durant les séances de TP, il s'agit principalement d'assemblage structurel !

Le point chaud : La FSM

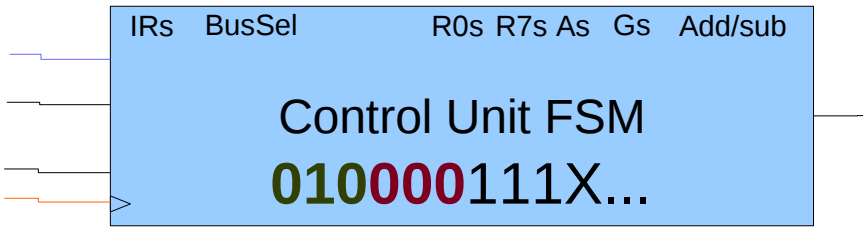
C'est la seule partie qui n'est pas textuellement décrite. Ce qu'il faudra faire avant de se lancer dans le projet !

Décrire sous forme de diagramme de moore la FSM permettant de réaliser les différentes instructions requise et extensible pour de nouvelle instructions.

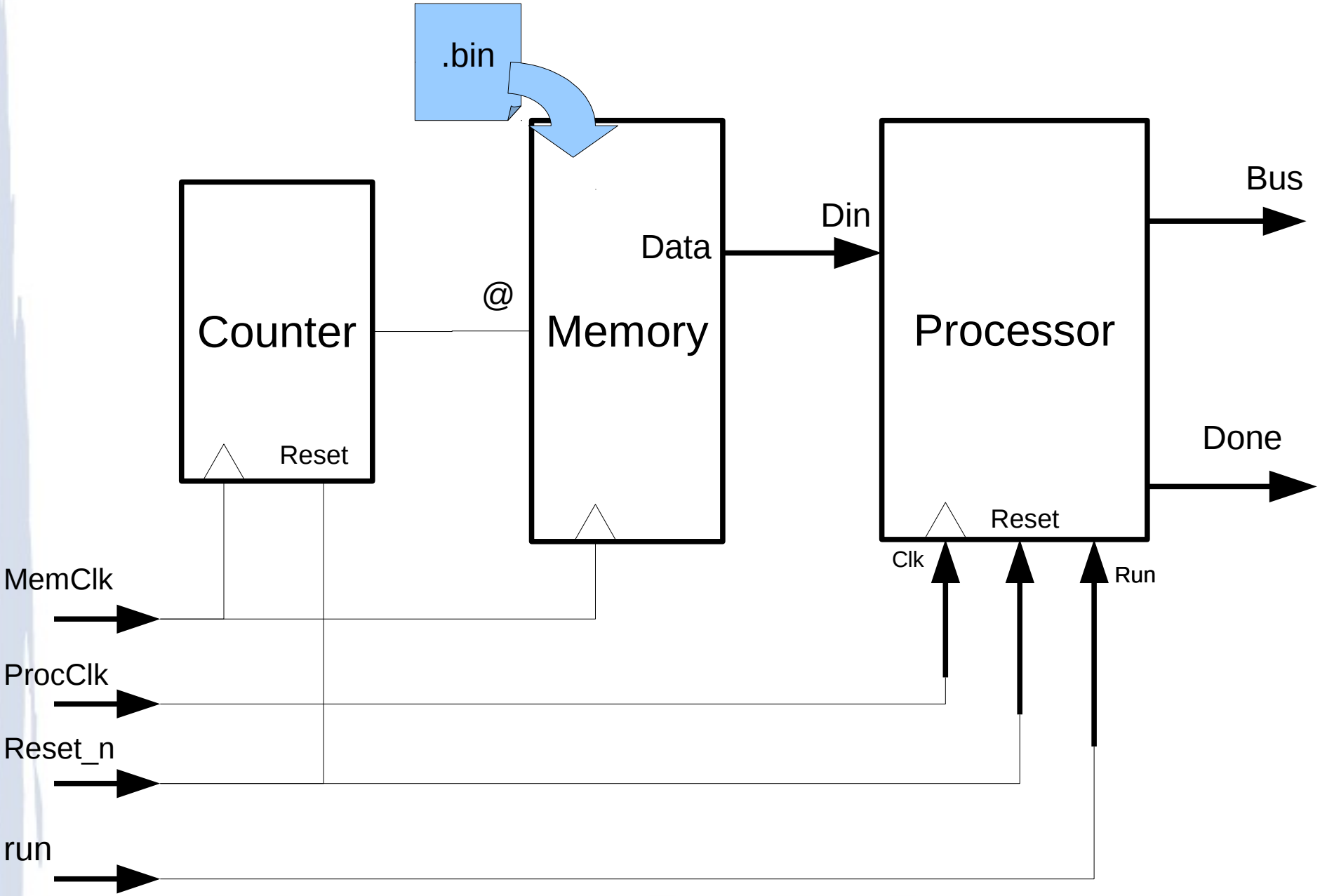
Ce n'est pas très difficile si l'on se base sur les FSM réalisées en TP et que l'on a bien compris les différentes étapes de réalisation d'un instruction (FETCH, DECODE, EXECUTE, ...)

Penser à prendre en compte RUN !

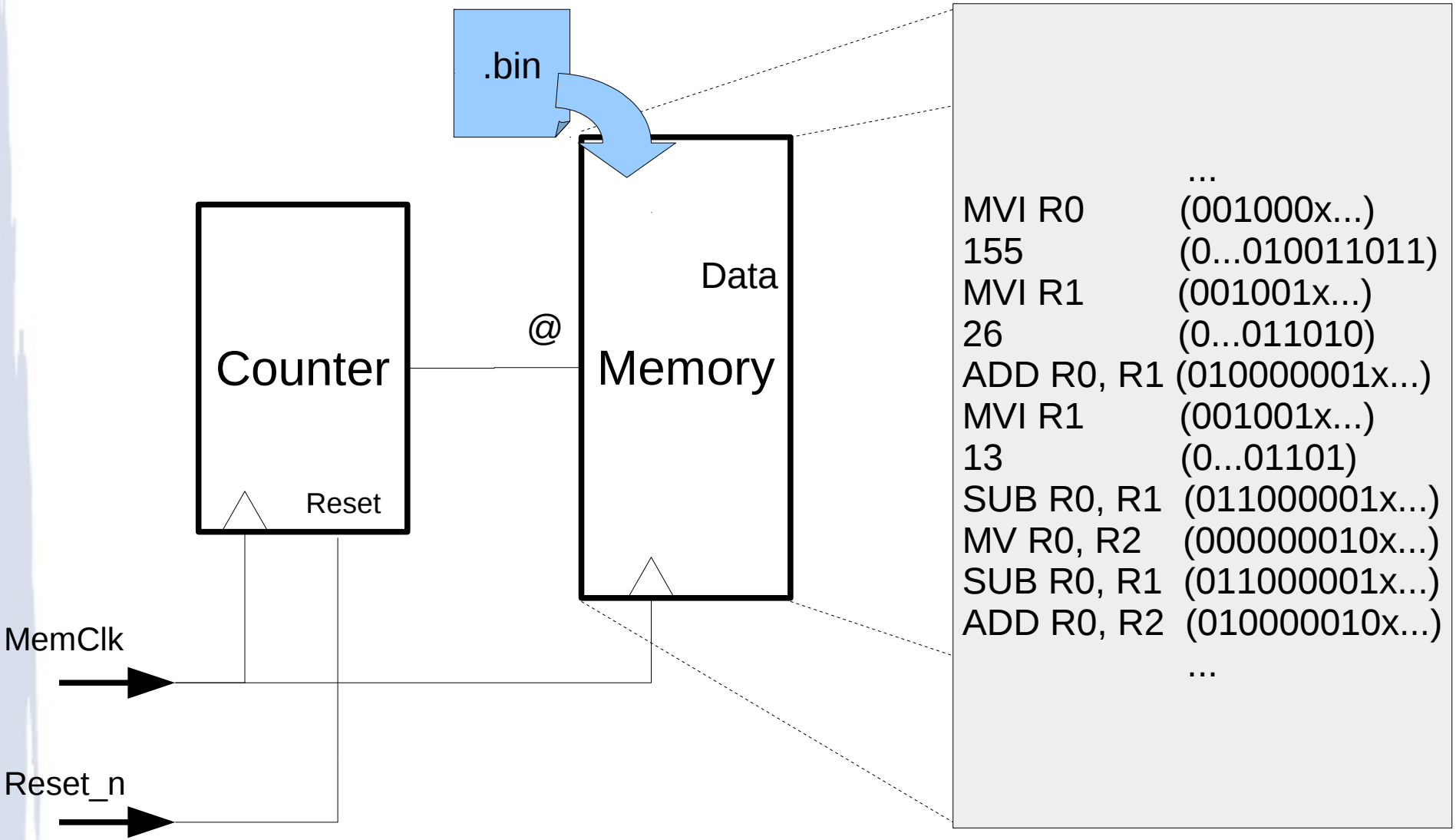
???



VIII - Processeur 16bits : Utilisation de la mémoire



VIII - Processeur 16bits : Utilisation de la mémoire



Bénéfices : On ne positionne plus les instructions et valeurs via les switchs !

VIII - Processeur 16bits : Extensions !

De manière à apporter vos propres modifications au processeur, il vous sera demander de réfléchir et de réaliser l'une, au moins, des extensions suivantes :

- **L/S** : Ajout d'instruction de type 'I' pour le chargement (load) et le rangement (store) en mémoire qui permettent respectivement de charger et de ranger une donnée depuis ou vers la mémoire. Les opérandes de ces instructions sont un registres et une adresse mémoire codée comme un immédiat de 16 bits.
- **Logic** : Ajout d'instruction de type 'R' pour les opérations logiques (AND, OR, NOT).
- **Mult** : Ajout d'instruction de type 'R' pour la multiplication de deux nombres stockés dans des registres. Vous porterez attention à expliquer comment est traité le résultat dans les registres.
- **Jump** : Ajout d'un nouveau type d'instruction de type 'J' pour réaliser des sauts dans l'exécution des instructions. Ce type d'instructions n'a qu'une seule opérande : une adresse de saut sur 16 bits.
- **Compilo** : Écriture d'un assembleur en Java, C ou C++ qui permet de générer directement le fichier MIF à partir d'une fenêtre de saisie du code assembleur.

Biblio

Documents de cours

- Ce cours en ligne à :

Http://perso-etis.ensea.fr/rodriguez/

- Un très bon support de cours

http://hdl.telecom-paristech.fr/index.html

- Le cours de Licence 2 (en particulier pour ceux qui ne l'ont pas suivi):

http://perso-etis.ensea.fr/miramond/Enseignement/L2/circuits_numeriques.html

- Le cours de Licence 3 de 2013 :

http://perso-etis.ensea.fr/miramond/Enseignement/L3/Cours_VHDL_2011.html

Documents de développement

- Quartus

http://quartushelp.altera.com/current/

- Documentation Altera sur les Cyclones II et IV (entre autre...)

http://www.altera.com/literature/lit-index.html