

Unité : SOCF

# Architecture SoC-FPGA

## DMA Controller

heig-**vd**

HAUTE ÉCOLE  
D'INGÉNIERIE ET DE GESTION  
DU CANTON DE VAUD

[www.heig-vd.ch](http://www.heig-vd.ch)

Etienne Messerli

avril 2021

**ReDS**



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License

# Contenu

---

- Principe fonctionnement DMA
- DMA :  $\mu$ C DM3730
- DMA : HPS Cyclone-V
- DMA : IPs Intel-Altera de Qsys

# Goulet ?

---

Dans un système à processeur le goulet d'étranglement est fréquemment :

Transferts des données depuis/vers les IOs

- Amélioration par un DMA:
  - Transfert direct avec la mémoire (DMA)
  - Fonctionne indépendamment du CPU
  - Quittance lorsque transfert terminé (IRQ)

# Transfert DMA ...

---

- DMA: Direct memory access
  - Assure le transfert automatique de données vers/depuis la mémoire
  - Utile pour des volumes conséquent de donnée
  - Configuré par le CPU
  - Indication du transfert terminé
    - via une interruption au CPU
  - Amélioration performance :
    - bus très large (128, 256 bits)
    - transfert en mode "burst"
  - Possible d'assurer des transferts entre les périphériques

# ... transfert DMA

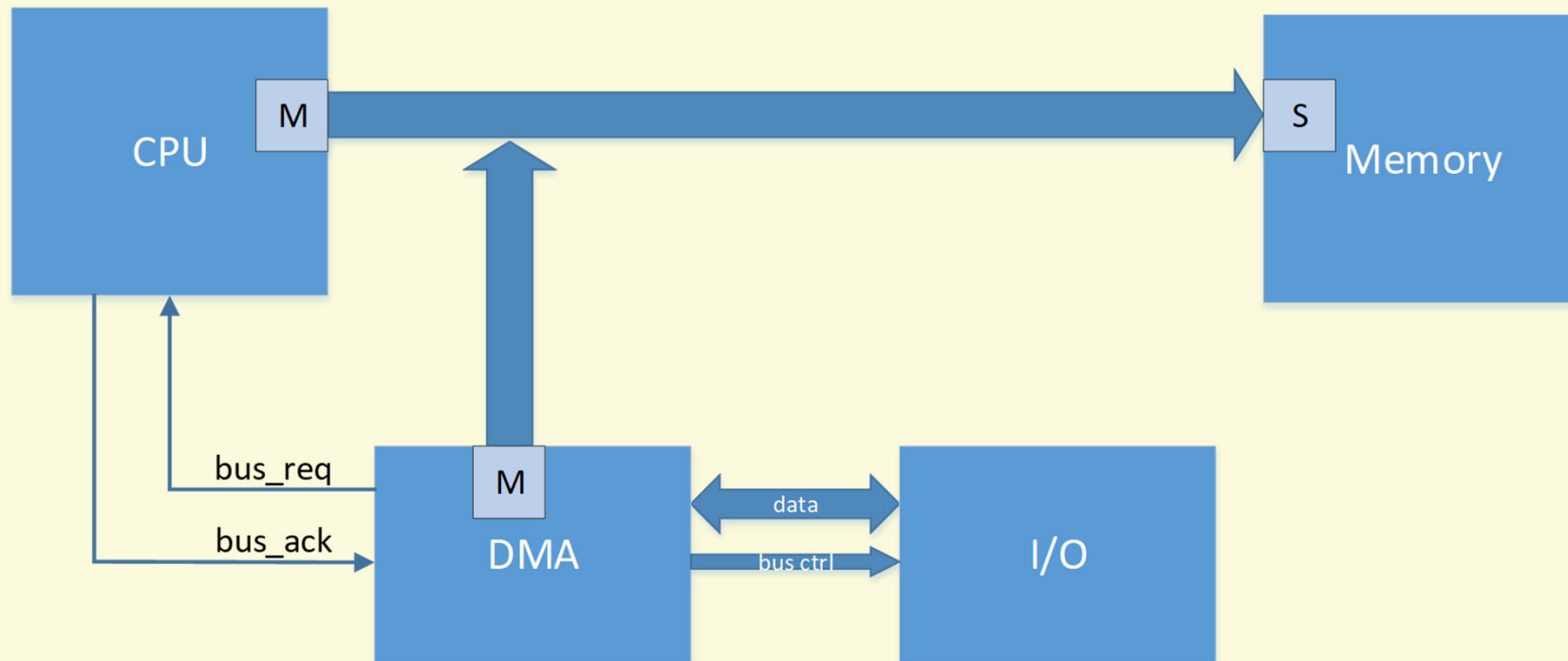
## Utilisation DMA (Direct memory access)

- Déterminer pour chaque critère si : + avantage / - inconvénient

Critères	+	-
Décharge le CPU		
Utilise le bus du système à processeur		
Transfert d'un mot (data) avec un seul accès sur le bus		
Permet de haut débit de données		
Nécessite périphérique avec bus spécifique		
Gestion du DMA (configuration, complexité)		
Gestion d'interruption		

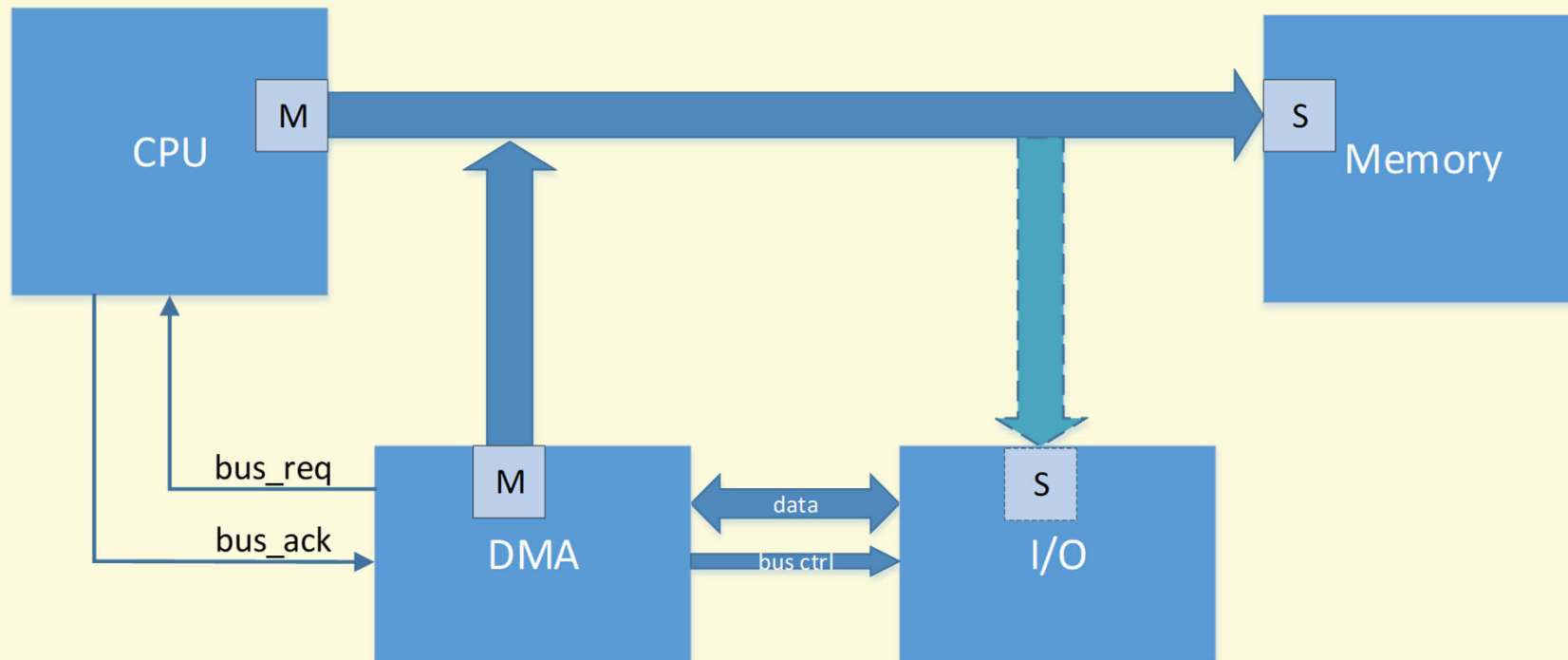
# DMA "full"

- DMA avec port spécifique pour IO (périphérique)



# DMA "full"

- DMA avec port spécifique pour IO (périphérique)



I/O avec bus slave pour  
la configuration, contrôle, ...

# DMA "full"

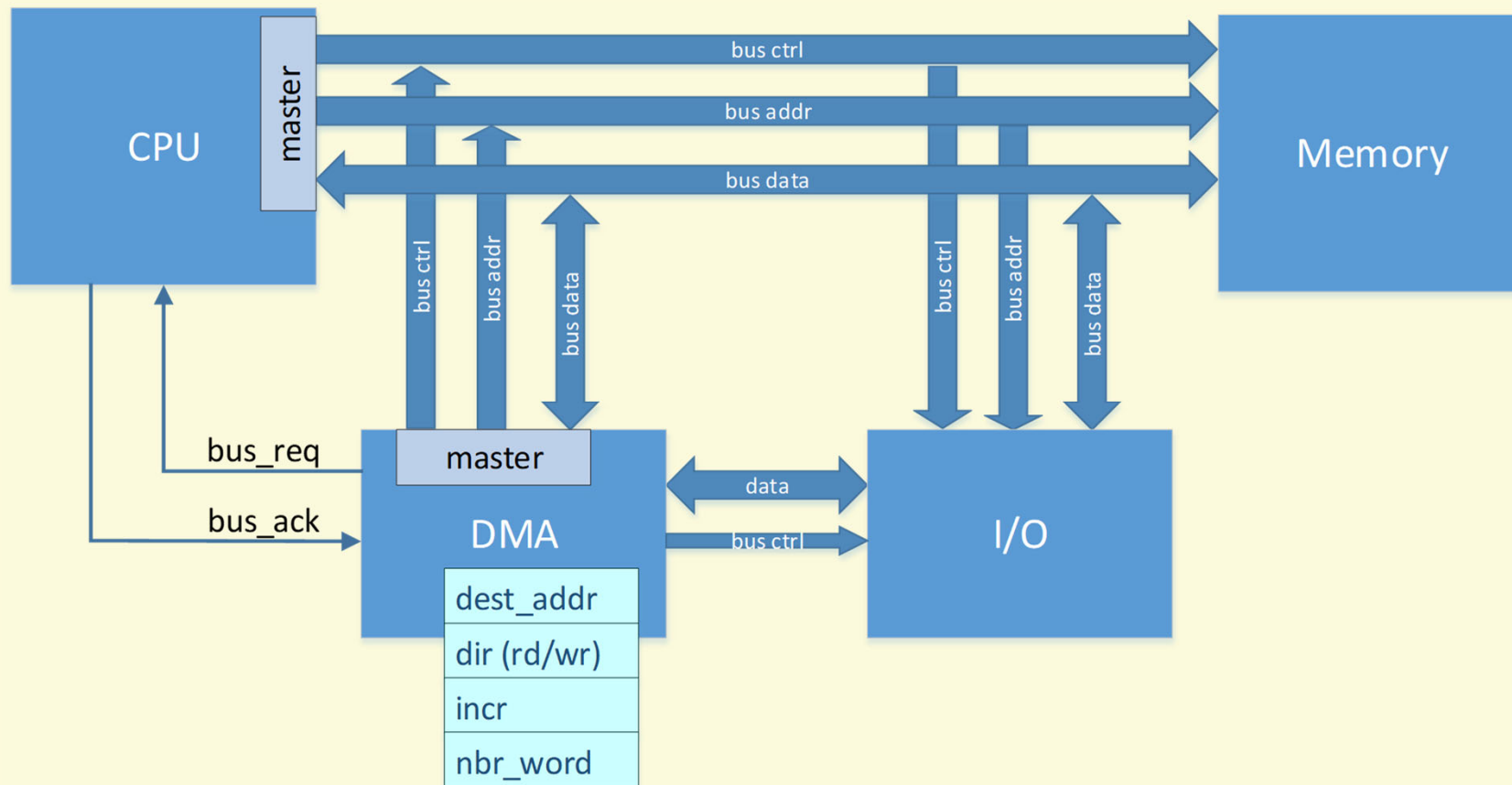
- Applications :
  - Périphérique avec grand volume de donnée
  - Système à haute performance
  - ....

Critères	+	-



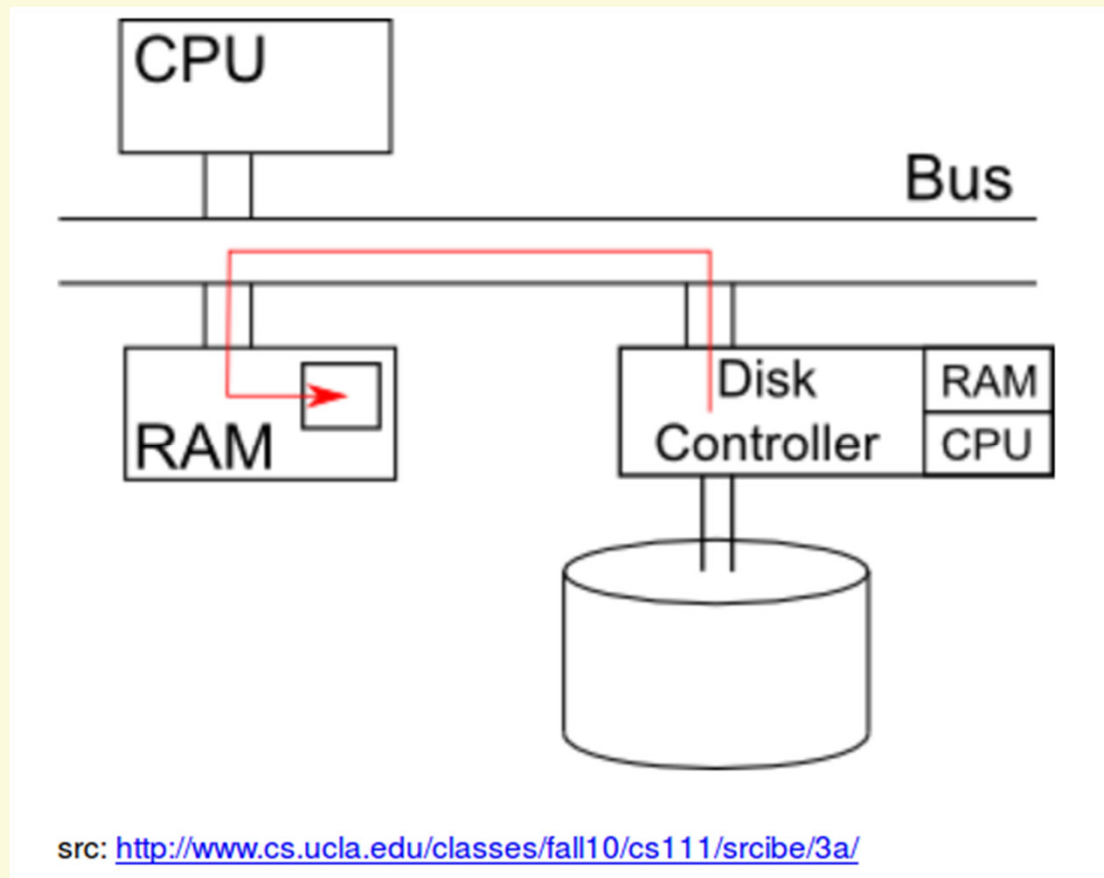
# DMA "full"

- Contenu du *descriptor*



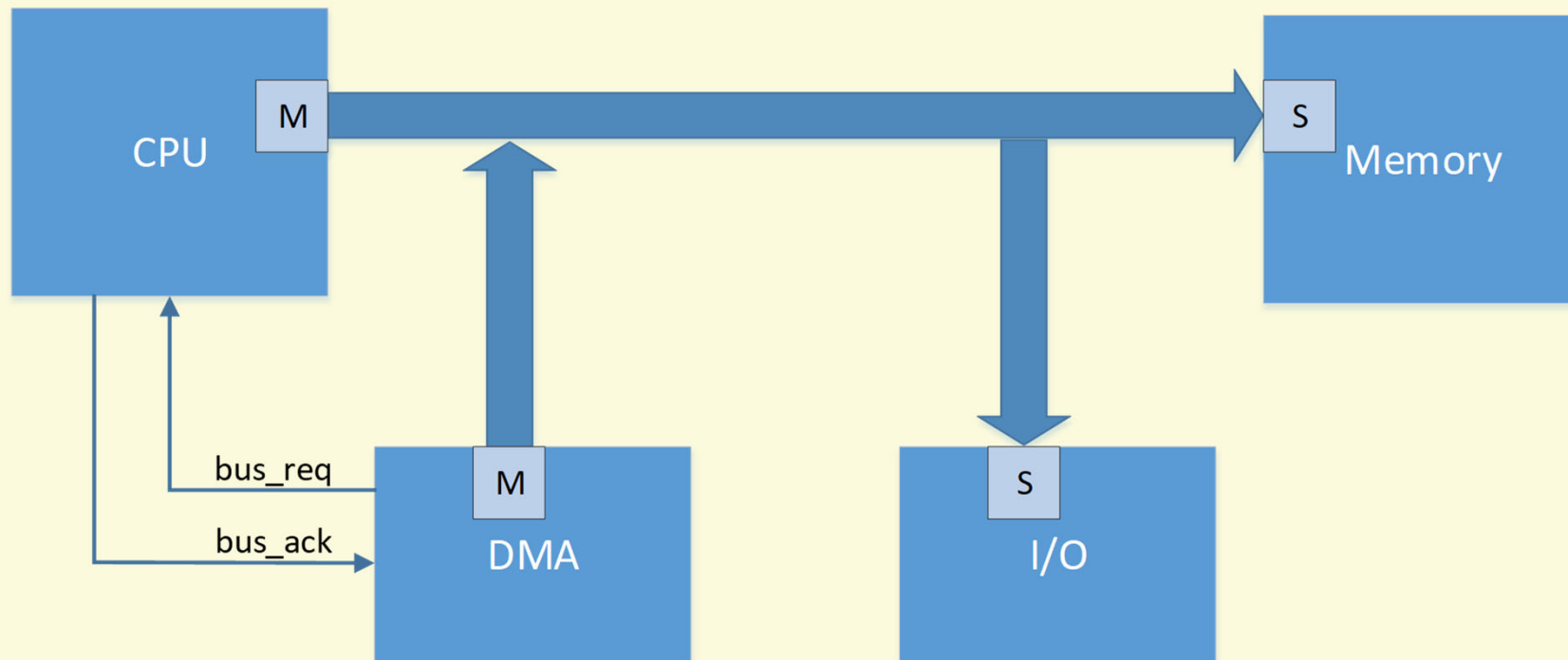
# DMA "full"

- Contrôleur DMA pour disque dur



# Contrôleur DMA, "indirect"

- Contrôleur DMA connecté uniquement sur le bus du système à processeur



# Contrôleur DMA, "indirect"

---

- Il y a deux cas de transferts :
  - périphérique => mémoire
  - mémoire => périphérique
- Le transfert des data va nécessiter 2 accès sur le bus :
  - Transfert périphérique <> contrôleur DMA
  - Transfert data DMA

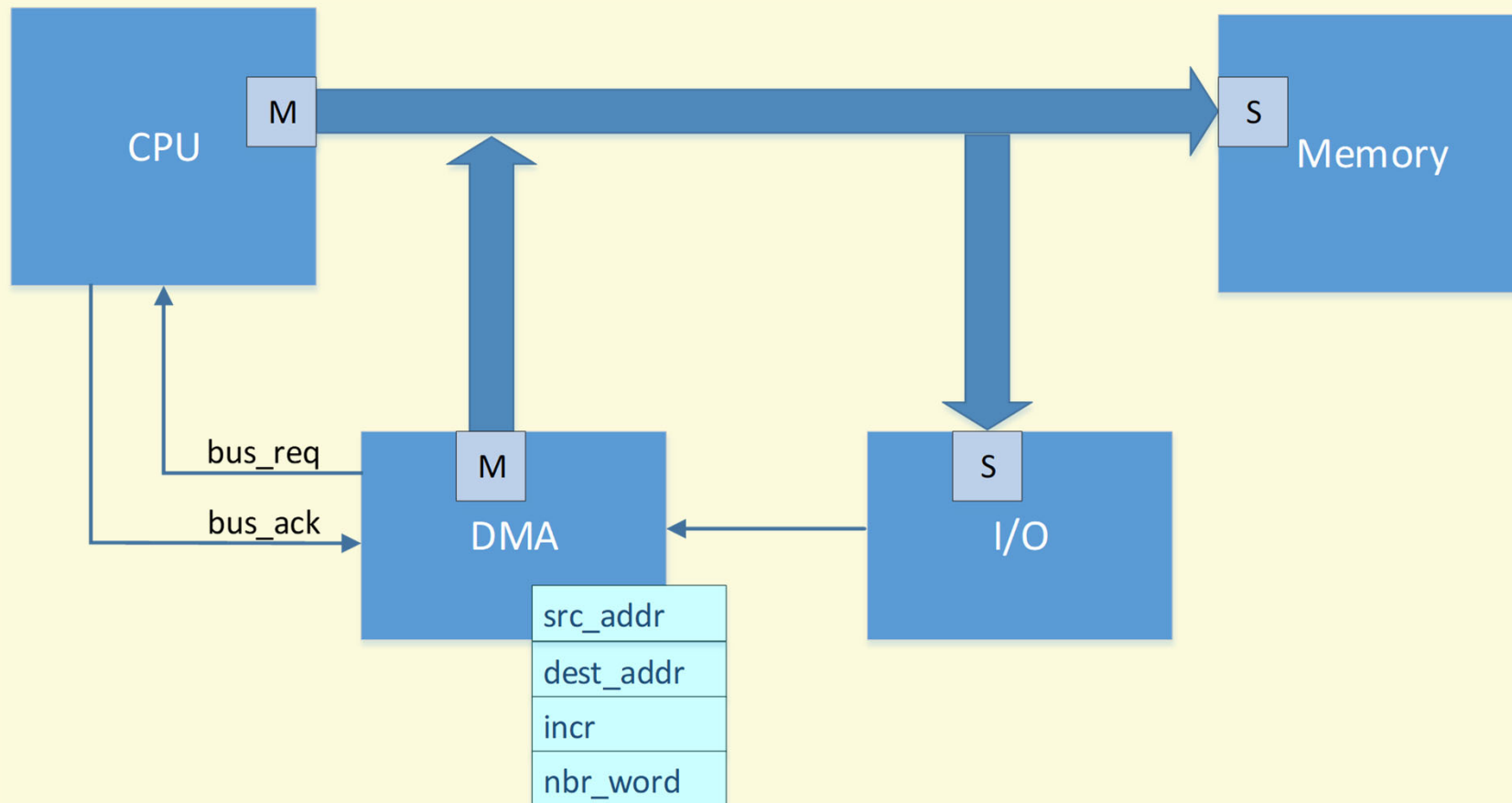
# Contrôleur DMA, "indirect"

- Applications :
  - Système low cost
  - Microcontrôleur gamme moyenne
  - ...

Critères	+	-

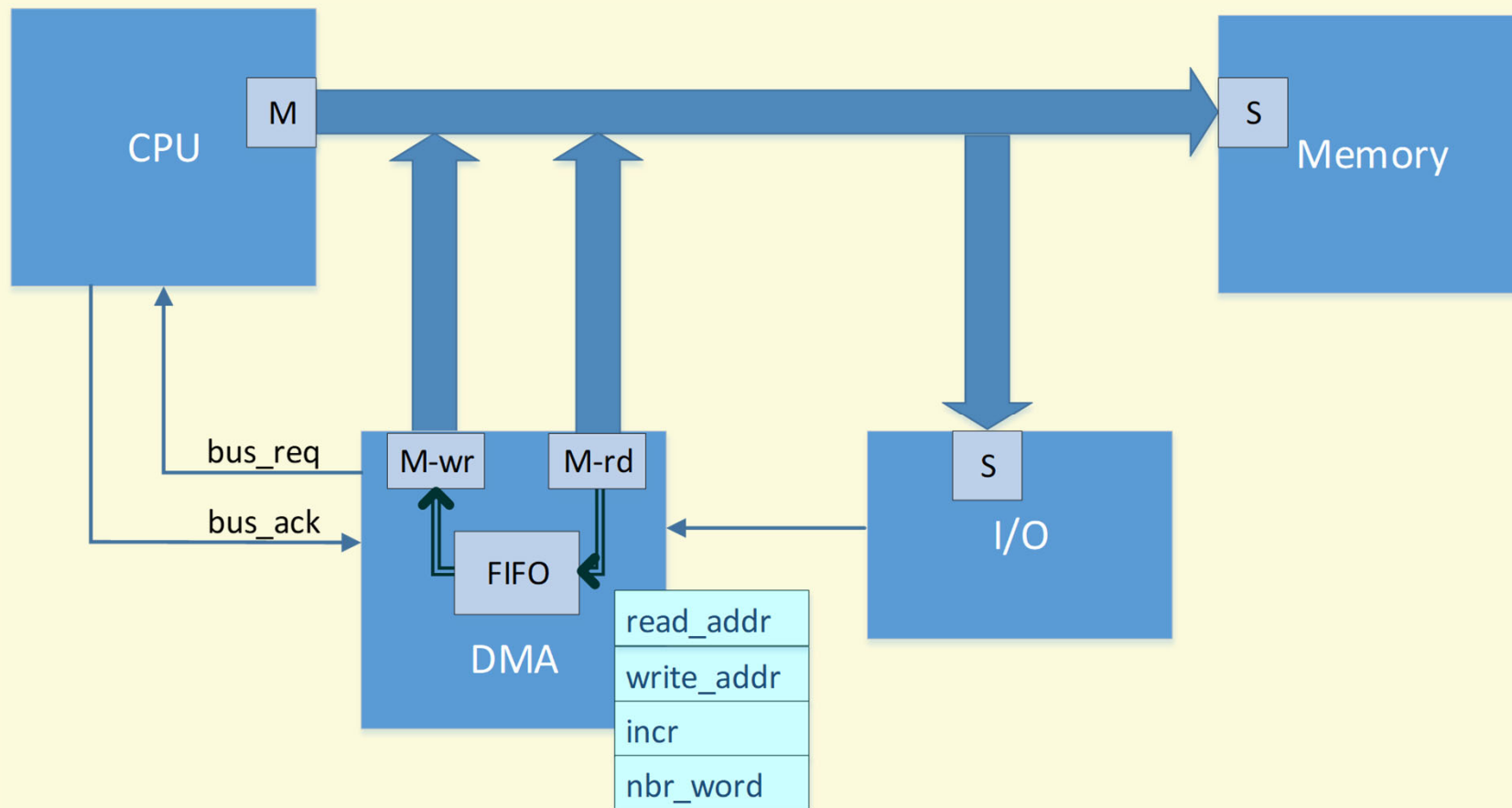
# Contrôleur DMA, "indirect"

- Contenu du *descriptor*
- Interaction entre DMA et périphérique (resquet)



# Contrôleur DMA, "indirect"

- Détail de l'architecture avec mémoire interne



# Contrôleur DMA REPTAR

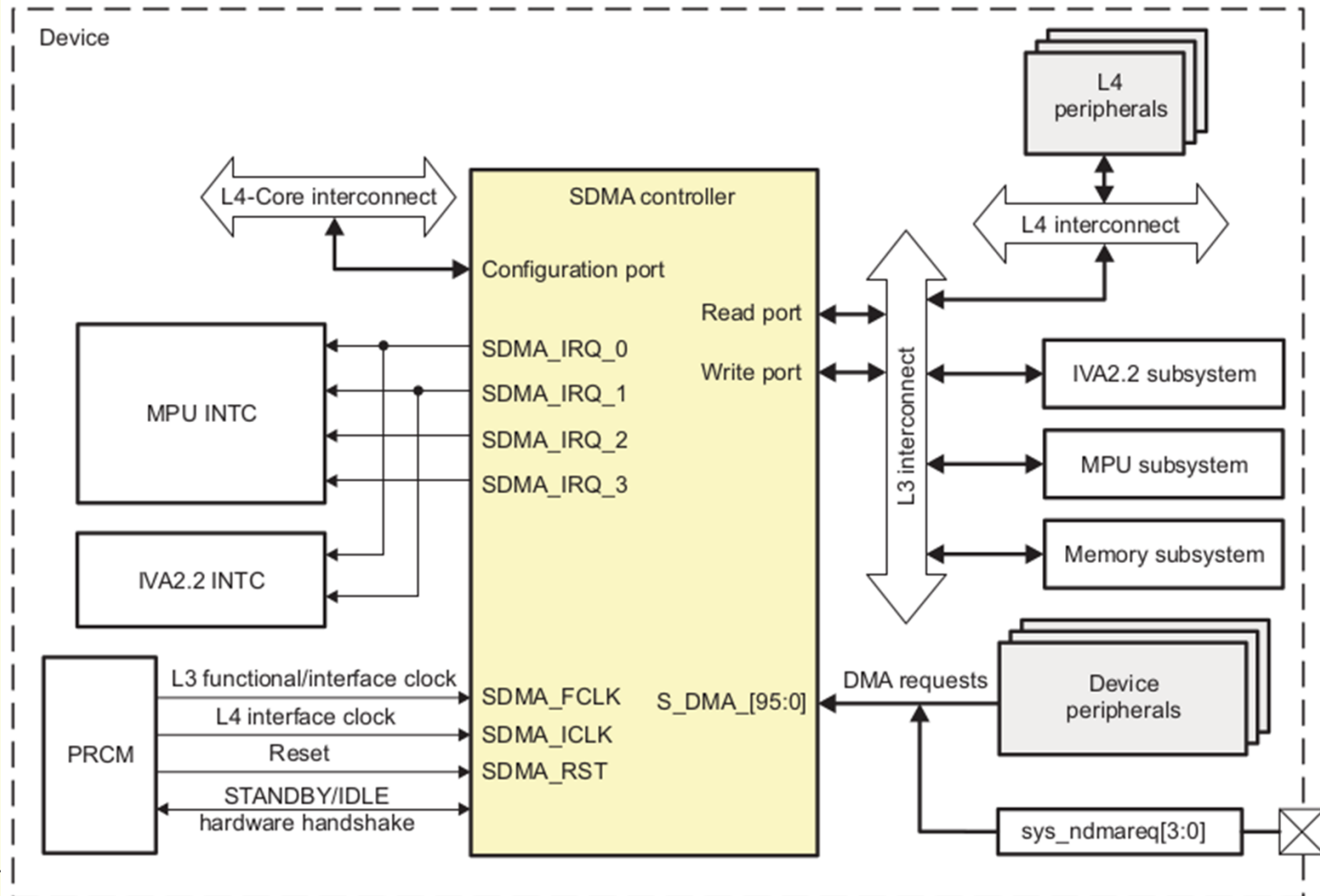
---

- Exemple contrôleur DMA module DM3730
  - SDMA: system direct memory access
  - Caractéristiques :
    - Data transfer support in either direction between:
      - Memory and memory
      - Memory and peripheral device
    - 32 logical DMA channels supporting:
      - 8-bit, 16-bit, or 32-bit data element transfer size
    - First-come, first-serve DMA scheduling with fixed priority
    - Up to 96 DMA requests
    - Four programmable interrupt request output lines
    - FIFO depth: 256 x 32-bits



# SDMA module DM3730 ...

Figure 11-1. SDMA Overview

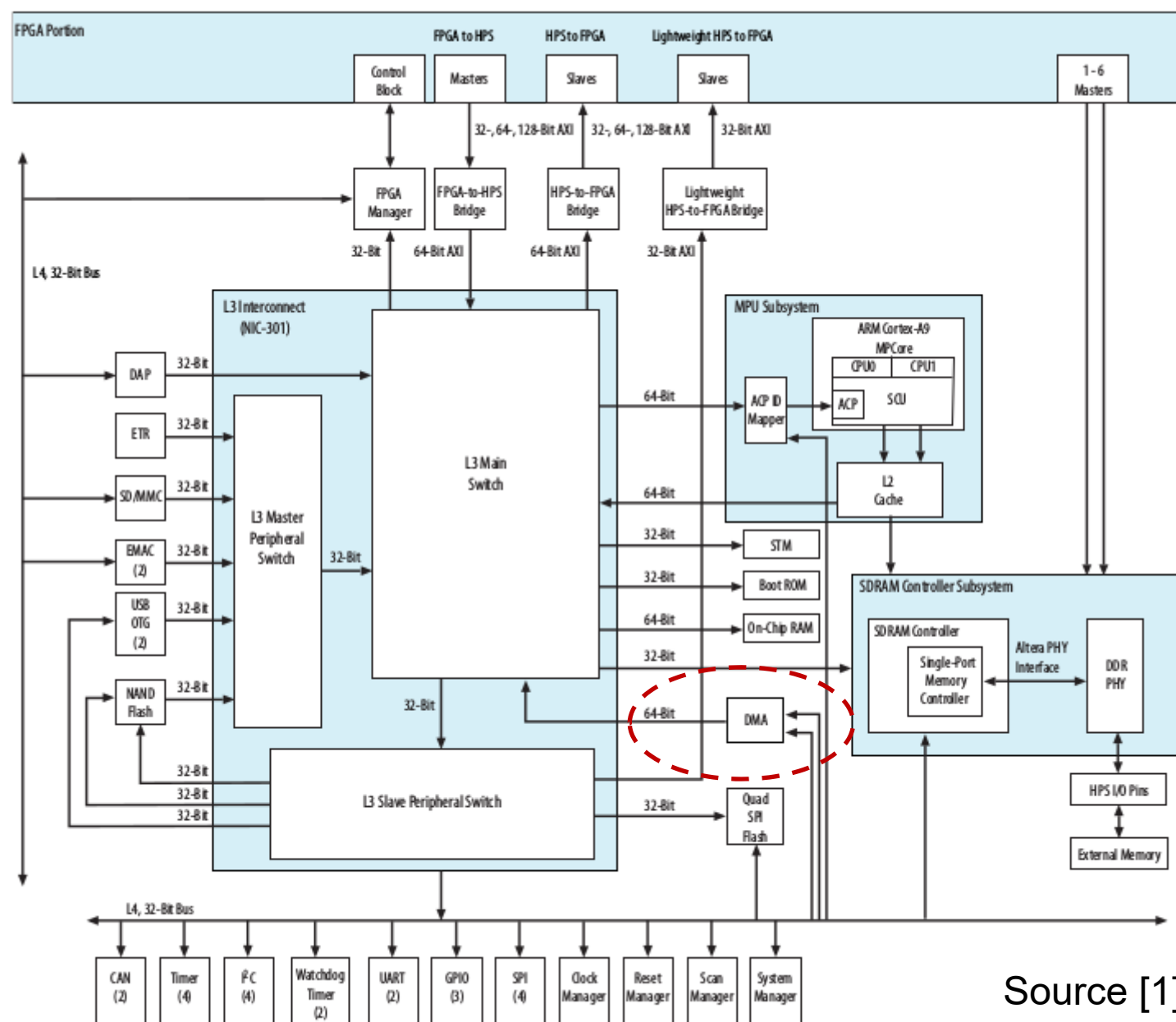


# Controller DMA HPS Cyclone-V

---

- Features of the DMA Controller
  - The DMAC supports multiple transfer types:
    - Memory-to-memory
    - Memory-to-peripheral
    - Peripheral-to-memory
  - The DMAC supports up to:
    - Eight logical channels for different levels of service requirements
    - 31 peripheral handshake interfaces for peripheral hardware flow control
  - The DMAC support scatter-gather transfert

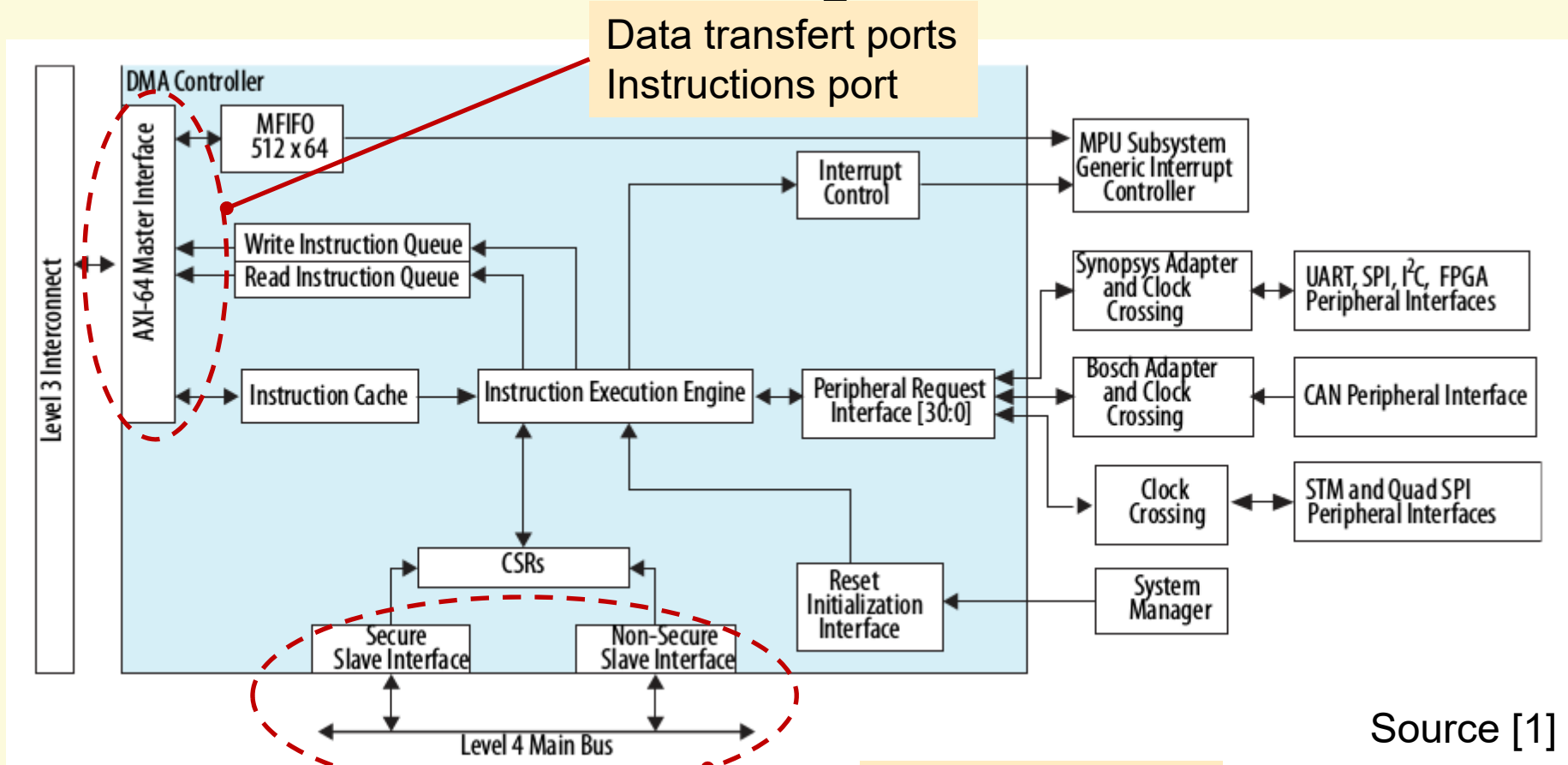
# Cyclone V SoC block diagram



Source [1]

# Controller DMA HPS Cyclone-V

- DMA Controller: Bloc diagram



# Controller DMA HPS Cyclone-V

---

The DMAC provides:

- An instruction processing block that enables it to process program code that controls a DMA transfer
- An AMBA-AXI master interface unit **to fetch the program code from system memory into its instruction cache**

Note: The AXI master interface **also performs DMA data transfer**.

The DMA instruction execution engine executes the program code from its instruction cache and schedules read or write AXI instructions through the respective instruction queues.

- A multi-FIFO (MFIFO) **data buffer** that stores data that it **reads**, or **writes**, during a DMA transfer
- 11 interrupt outputs to enable efficient communication of events to the MPU subsystem
- A slave interface for the configuration (secure/non-secure)

# Controller DMA & FPGA

---

- Intégration de contrôleur DMA dans la FPGA
- Permet un DMA sur mesure
  - choix du type et de la performance du DMA
- Plusieurs type d'IP disponible
- Utilisation de Qsys
- Version de contrôleur DMA avec connexion direct pour un périphérique
  - bus streaming
  - pas d'adresse
- Transfert sans ou avec "burst"
- Contrôleur simple ou full (SGDMA, scatter-gather DMA)
  - scatter – gather => disperser - regrouper

# Type de contrôleur DMA

---

- DMA classique (DMA Controller)
  - Transfert par bloc avec adresses continues
- Scatter-Gather DMA (disperser – assembler, full DMA)
  - Transfert portant sur plusieurs zones mémoires
  - Typologies
    - memory-to-memory, memory-to-streaming, streaming-to-memory
  - Système avec mémoire virtuelle
    - Taille page mémoire virtuelle!
      - fréquemment 4K
    - Zone continue en mémoire virtuelle peut-être discontinue en mémoire physique !
    - Nécessite une succession de transferts
      - Descripteurs chaînés

# IP Intel-Altera: DMA Controller

---

## Embedded Peripherals IP

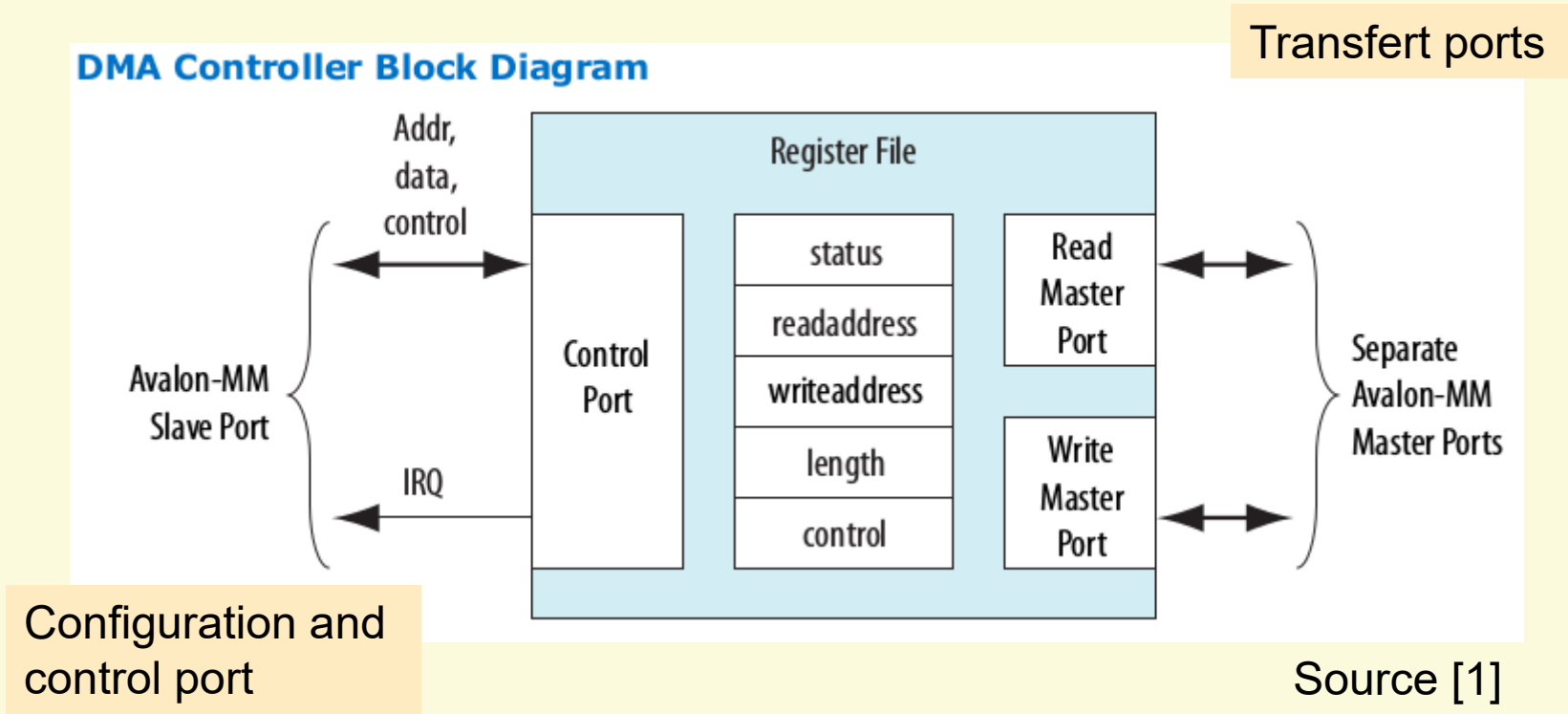
- DMA Controller
  - Interface Avalon slave for configuration
  - 2 Avalons memory-mapped master: read & write



# IP Intel-Altera: DMA Controller

- DMA Controller

DMA Controller Block Diagram



# IP Intel-Altera: DMA Controller

---

Typical DMA transaction proceeds as follows:

1. CPU prepares DMA transaction by writing to the control port
2. CPU enables the DMA controller
  - The DMA controller then begins transferring data without additional intervention from the CPU
  - reads data from the read address/ master write port writes the data to the destination address
3. The DMA transaction ends when a specified number of bytes are transferred. At the end DMA controller generates an interrupt request (IRQ) if configured.
4. During or after the transaction, the CPU can examine the DMA controller's status register (info on transaction progress)

# IP Intel-Altera: DMA Controller

- Explication complète :

Source [1]

A typical DMA transaction proceeds as follows:

1. A CPU prepares the DMA controller for a transaction by writing to the control port.
2. The CPU enables the DMA controller. The DMA controller then begins transferring data without additional intervention from the CPU. The DMA's master read port reads data from the read address, which may be a memory or a peripheral. The master write port writes the data to the destination address, which can also be a memory or peripheral. A shallow FIFO buffers data between the read and write ports.
3. The DMA transaction ends when a specified number of bytes are transferred (a fixed-length transaction) or an end-of-packet signal is asserted by either the sender or receiver (a variable-length transaction). At the end of the transaction, the DMA controller generates an interrupt request (IRQ) if it was configured by the CPU to do so.
4. During or after the transaction, the CPU can determine if a transaction is in progress, or if the transaction ended (and how) by examining the DMA controller's status register.

# IP Intel-Altera: DMA Controller

---

- Configuration:

Source [1]

- Read (source) address location
- Write (destination) address location
- Size of the individual transfers: Byte (8-bit), halfword (16-bit), word (32-bit), doubleword (64-bit) or quadword (128-bit)
- Enable interrupt upon end of transaction
- Enable source or destination to end the DMA transaction with end-of-packet signal
- Specify whether source and destination are memory or peripheral

The master peripheral then sets a bit in the `control` register to initiate the DMA transaction.

# IP Intel-Altera: DMA Controller

- Registres de contrôle, 32 bits:

Offset	Registre	R/W	Commentaires
0	status	RW	Bit de statut du DMA, voir ci-après
1	read_addr	RW	Adresse de départ de l'accès "read master"
2	write_addr	RW	Adresse de départ de l'accès "write master"
3	length	RW	Nombre de <b>bytes</b> à transférer
4	reserved	-	
5	reserved	-	
6	control	RW	Bits de contrôle du DMA, voir ci-après
7	reserved	-	

# IP Intel-Altera: DMA Controller

- Registre : statut

bit	31 ...13	12 ... 5	4	3	2	1	0
	reserved	reserved	len	weon	reop	busy	done

Expliquer les différents bits?

Voir source [1]

# IP Intel-Altera: DMA Controller

- Registre : control

bit	31 ... 13	12	11	10	9	8	7
		Software reset	Quad Word	Double word	WCon	RCon	LeEn

bit	6	5	4	3	2	1	0
	WeEn	ReEn	I_EN	GO	Word	HW	Byte

Expliquer les différents bits?

Voir source [1]

# IP Intel-Altera: Scatter-Gather DMA

---

## Embedded Peripherals IP

- Modular Scatter-Gather DMA Core Interface
  - One Avalon-MM CSR slave port
  - One Descriptor slave port
  - One configurable Avalon-MM Slave or Avalon-ST Source Response port
  - Source and destination data path ports, which can be Avalon-MM or Avalon-ST
  - Interrupt output.



# IP Intel-Altera: Scatter-Gather DMA

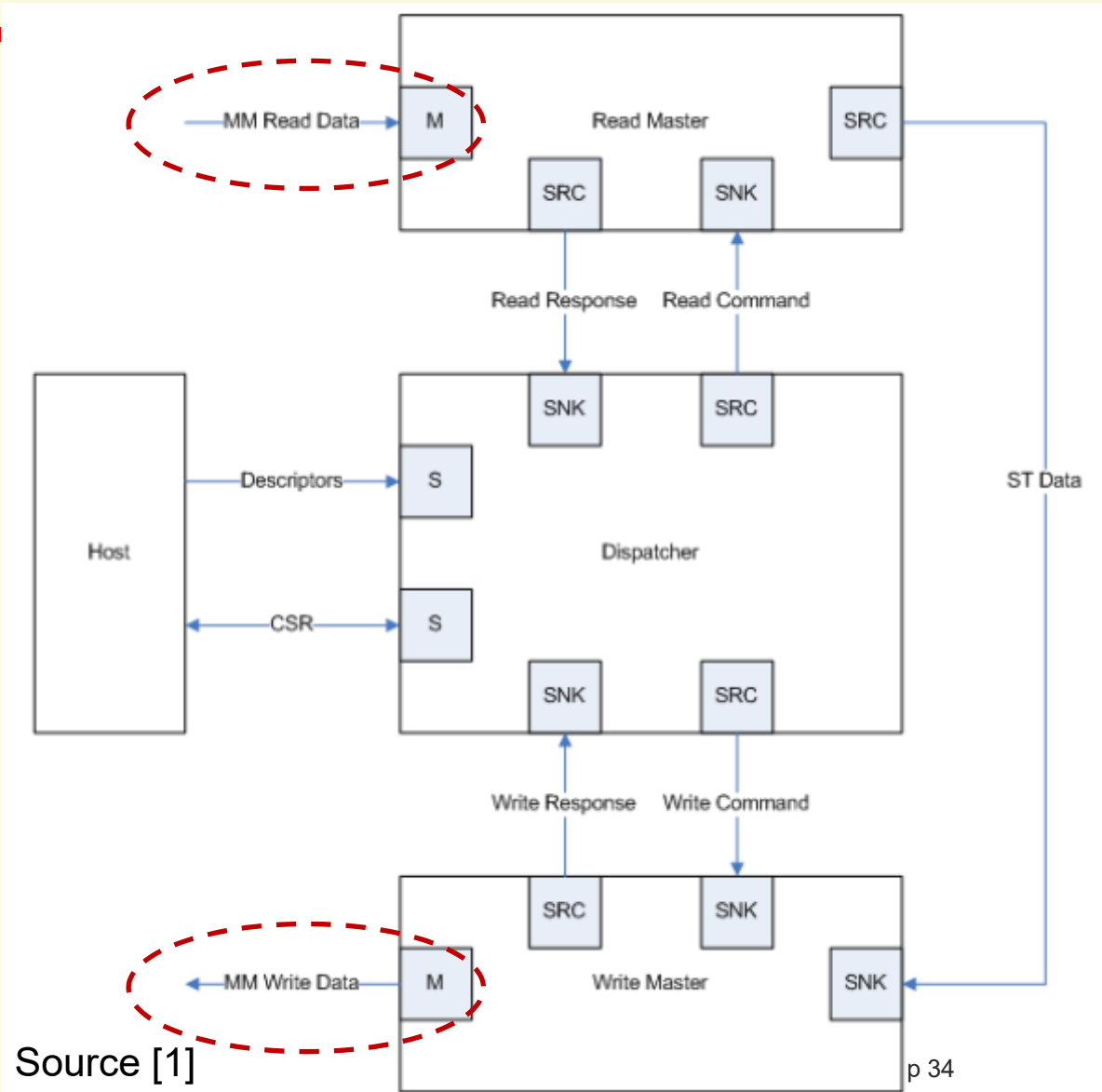
---

## Embedded Peripherals IP

- Modular Scatter-Gather DMA Core
  - operations with preloaded instructions
  - operations with called descriptors
  - multiple descriptors with different transfer sizes
  - source and destination addresses
  - Configurations
    - Memory-Mapped reads and writes
    - Memory-Mapped read to streaming bus out (write)
    - Streaming bus in (read) to the Memory-Mapped write

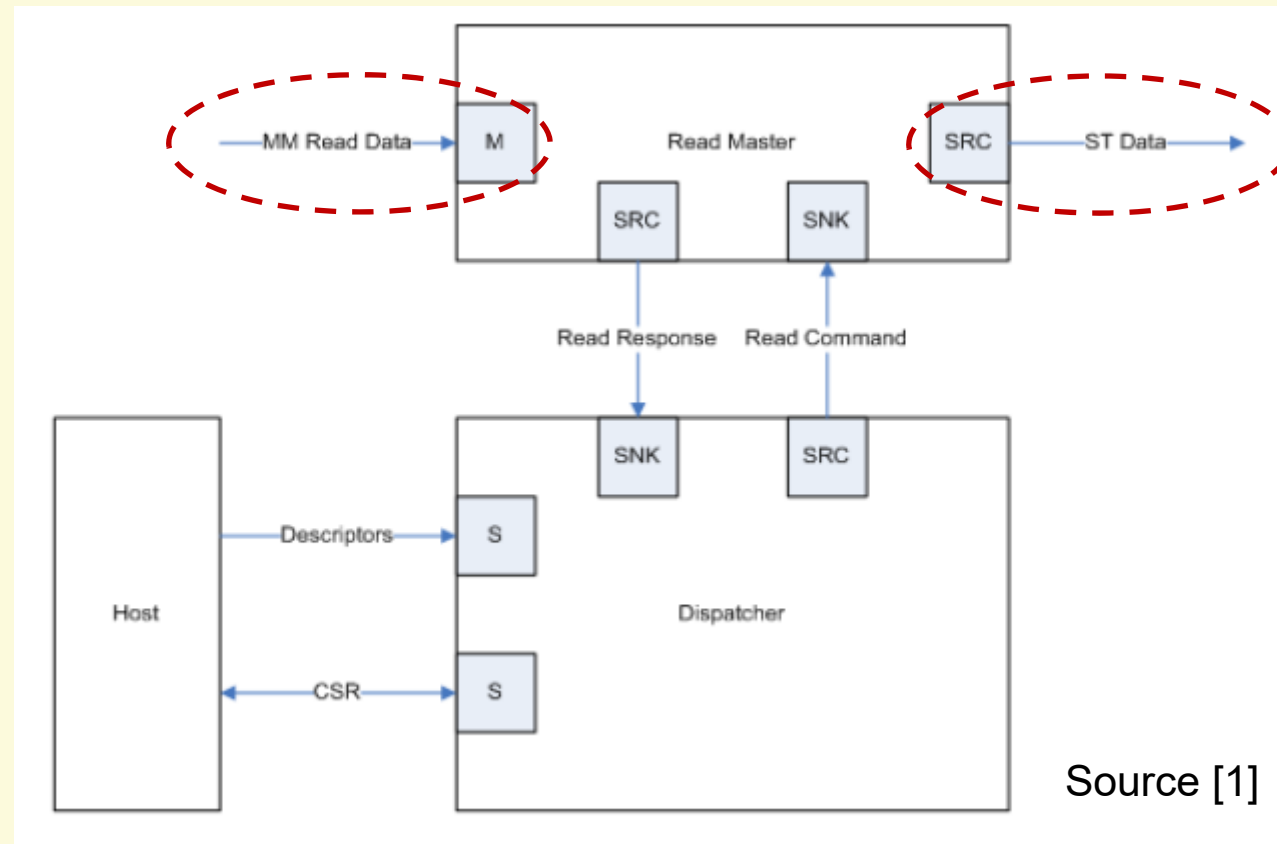
# IP Intel-Altera: Scatter-Gather DMA

- Memory-Mapped reads and writes



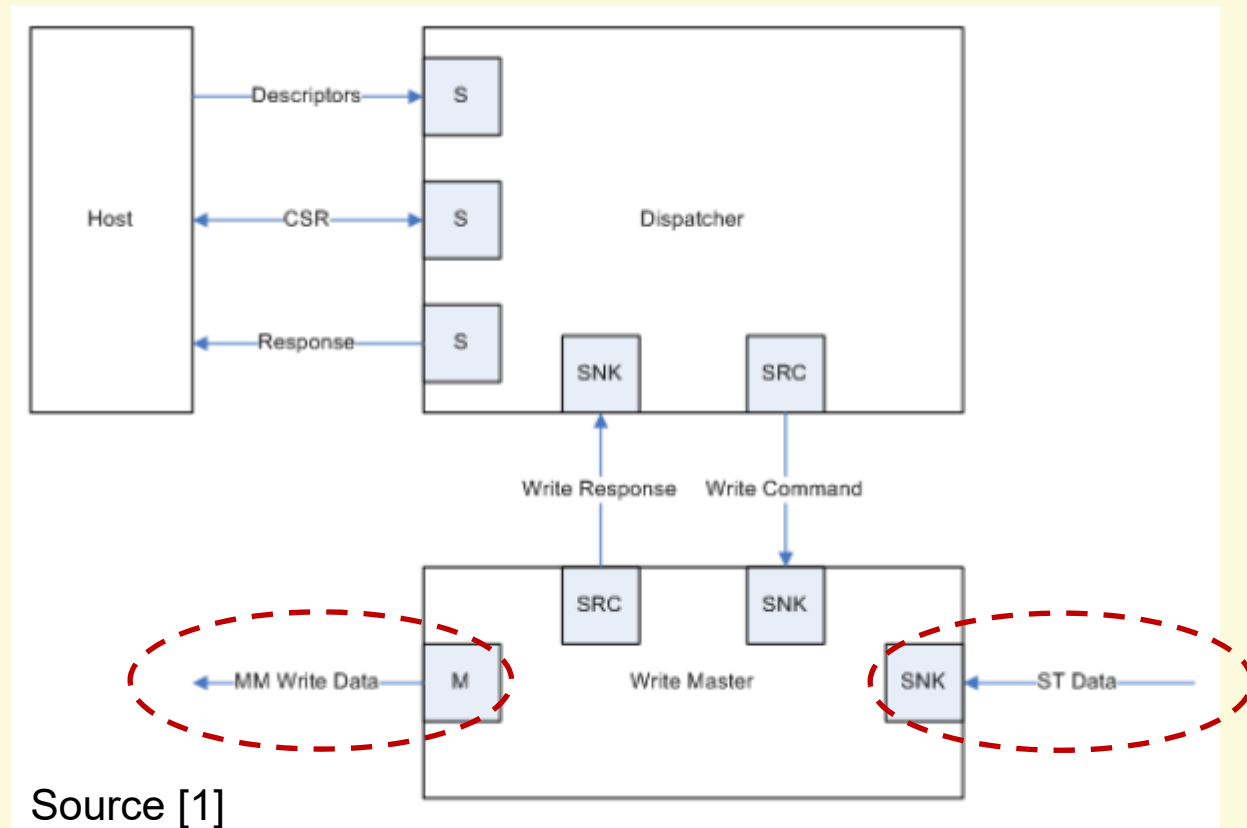
# IP Intel-Altera: Scatter-Gather DMA

- Memory-Mapped read to streaming bus out (write)



# IP Intel-Altera: Scatter-Gather DMA

- Streaming bus in (read) to the Memory-Mapped write



# IP Intel-Altera: Scatter-Gather DMA

- Standard descriptor format : 128 bits

**Table 197. Standard Descriptor Format**

Offset	Byte Lanes			
	3	2	1	0
0x0	Read Address[31:0]			
0x4	Write Address[31:0]			
0x8	Length[31:0]			
0xC	Control[31:0]			

Source [1]

- Address
  - adresse de la mémoire physique
  - pas d'adresse pour transfert "streaming"
- Length
  - indique la longueur du transfert en bytes
- Control
  - Différentes information de control du transfert

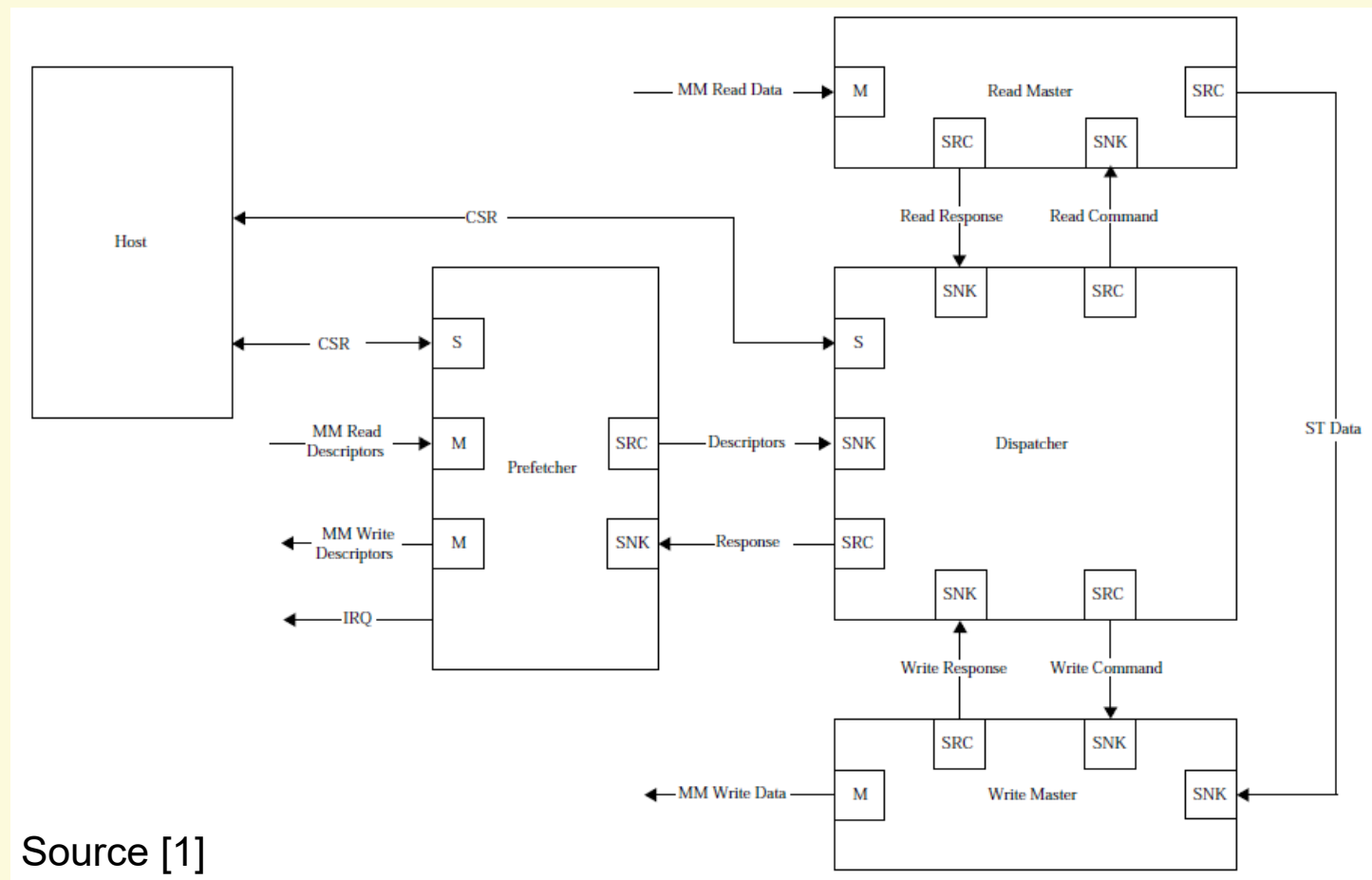
# IP Intel-Altera: Scatter-Gather DMA

---

- Standard descriptor : Control[31..0]
  - voir source [1] : § 26.4.6. Control Field
  - Quelle est la fonction du bit [31] : GO ?
  - Quelle est la fonction du bit [11] : Park Writes?
  - Quelle est la fonction du bit [10] : Park Reads?
  - Pourquoi la génération de l'IRQ est-elle configurable?
    - En principe il faudrait générer une interruption après chaque transfert DMA!

# IP Intel-Altera: Scatter-Gather DMA

- Prefetcher :



Source [1]

# IP Intel-Altera: Scatter-Gather DMA

---

- Quelle est la fonction du bloc Prefetcher ?
  - Etudier son rôle?
  - Indiquer sa fonction en lien avec le descriptor ?

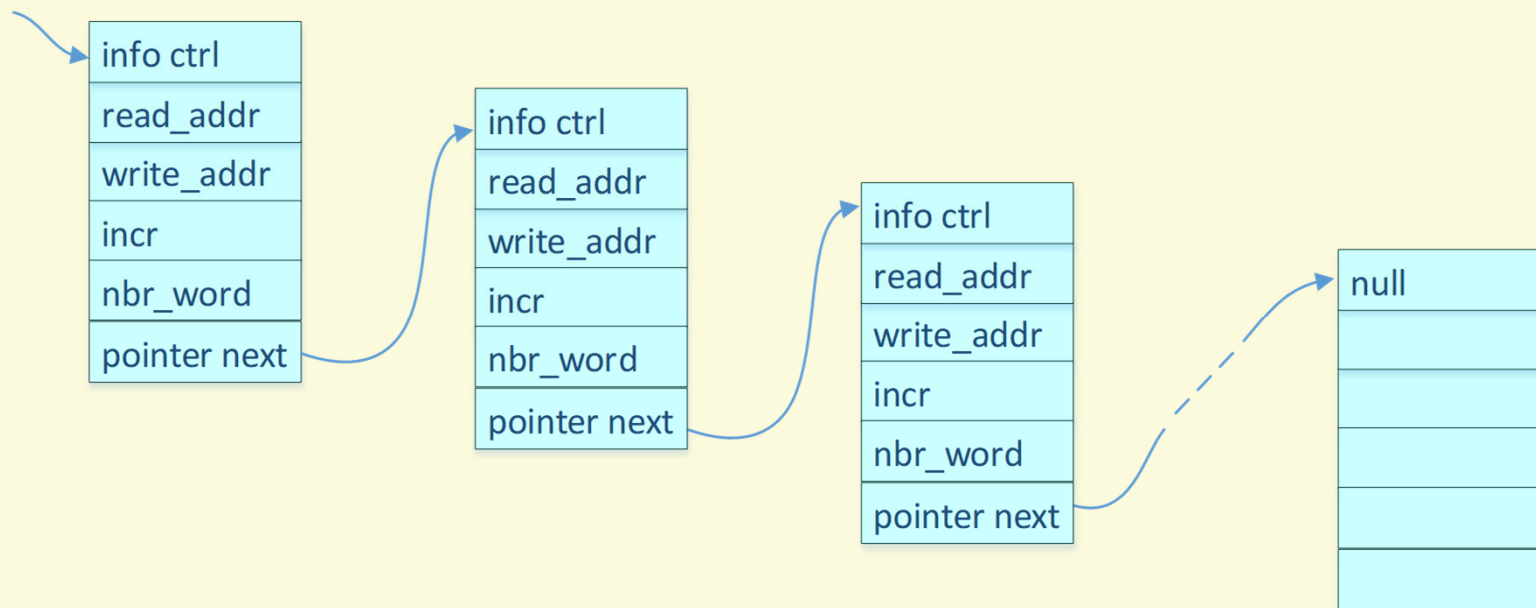
voir source [1] :

§ 26.7. Modular Scatter-Gather DMA Prefetcher Core



# IP Intel-Altera: Scatter-Gather DMA

- Un transfert continu en mémoire virtuelle
  - implique plusieurs transferts dans la mémoire physique  
⇒ nécessite une chaîne de "descriptor"



# Liens sur les sources

---

## [1] Embedded Peripherals IP User Guide

- Intel-Altera, user guide : UG-01085, 2018.05.07

## [2] Wiki Intel FPGA

- [https://fpgawiki.intel.com/wiki/Modular\\_SGDMA](https://fpgawiki.intel.com/wiki/Modular_SGDMA)

# FIN présentation

---

## Questions

