

Grobkonzept: CRC-DMA

- CRC-IP hat Full-AXI-Master Schnittstelle, um Daten selbständig zu lesen und zu schreiben
 - ↳ hier: AXI3: maximale Burstlänge: 16 Worte à 32 Bit
 - ↳ CRC-IP soll die möglichst größte Burstlänge nutzen
 - CRC-Berechnung parallelisieren? → 1 Byte verrechnen pro Takt
 - **Idee:** CRC-Check durchführen → Demo-Software: CRC-Prüfsumme in Software berechnen und von CRC-IP überprüfen lassen
 - **Idee:** Anstatt, dass die CPU nach jeder berechneten Prüfsumme eines Daten-Pakets einen Interrupt bekommt und das IP fertig ist und wartet, soll es durch zusätzliche Signale möglich sein den Ablauf mehrfach durchlaufen zu lassen → CPU startet IP → IP berechnet für vorgegebene Anzahl gleichgroßer Datenpakete die CRC-Prüfsummen und schreibt sie hintereinander in den Speicher
 - ↳ Zusätzliches Signal für Anzahl Pakete
 - Datenzugriff soll byteweise möglich sein → nicht word-aligned
 - CRC-IP hat AXI-Lite Schnittstelle zur Kommunikation mit Software
- Register:

Name	Beschreibung
Steuerregister	RUN Bit, Interrupt Enable, Finished, Erste Adresse der Daten
Leseadresse	
Anzahl der Datenbytes	
Schreibadresse	Adresse wo IP Daten mit Prüfsumme hinschreibt
eventuell Generatorpolynom	
eventuell CRC-Länge in Bits	→ fix auf 32 Bit
Anzahl Pakete	

- Auf read-pipelining verzichten

Ablauf für CRC-Berechnung eines Pakets

- 1) CPU schreibt über AXI-Lite Schnittstelle die Leseadresse, Anzahl Datenbytes und Schreibadresse, Anzahl Pakete
- 2) CPU setzt RUN Bit und startet somit das IP
- 3) IP macht read burst \rightarrow wird in einem Puffer gespeichert \leftarrow
Leseadresse wird erhöht
- 4) byteweise Berechnung der CRC-Prüfsumme beginnt
- 5) parallel macht IP write burst auf die Schreibadresse und inkrementiert Schreibadresse

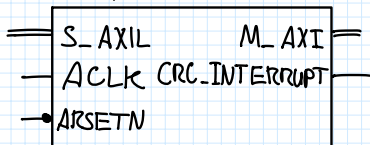
Wenn noch nicht alle Datenpakete gelesen wurden, springe zu Schritt 3)

- 6) IP schreibt berechnete Prüfsumme an Schreibadresse
- 7) Wenn Interrupt Enable Bit gesetzt ist, Interrupt setzen

- Wenn Bit auf 0 gesetzt wird, soll der aktuelle Auftrag zu Ende ausgeführt werden

IP-Konzept: axi_crc

Entity



Registerübersicht

Register	Offset	Beschreibung	Bits
Control	0x00	Steuerregister	[0] - Run; 0: Stop IP; 1: Start IP R0 [1] - Status; 0: Not Running; 1: Running W0 [2] - Interrupt Enable [31:3] - Reserved
Interrupt Status	0x04	Zurücksetzen des Interrupts	[0] - 1: Reset Interrupt; 0: nothing [31:1] - Reserved
Read Address	0x08	Adresse der zu lesenden Daten	[31:0] - Read address
Write Address	0x0C	Zieladresse für Daten + CRC	[31:0] - Write address
Packet Size	0x10	Größe eines Datenpakets in Bytes	[15:0] - Packet size [31:16] - Reserved
Number Packets	0x14	Anzahl der Pakete	[15:0] - Number Packets [31:16] - Reserved
Polynomial	0x18	Generatorpolynom für CRC	[31:0] - Polynomial
Initial Value	0x1C	Initialwert für CRC Berechnung	[31:0] - Initial Value

Zustandsautomat

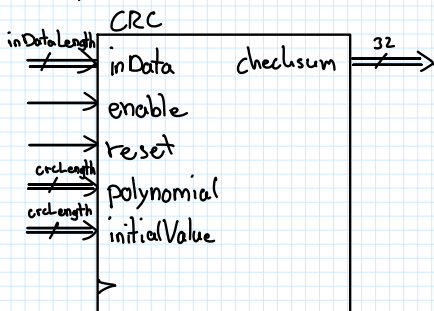
Eingänge	Ausgänge
run	status
interrupt_enable	interrupt_status
interrupt_reset	crc_en
raddr	crc_rst
waddr	byte_sel
packet_size	axi_size
packet_number	axi_addr
axi_idle	axi_write
	axi_start
	ram_sel
	ram_waddr
	ram_we
	ram_raddr
	ram_re

Zustände / Ablauf

1. IDLE - nix passiert

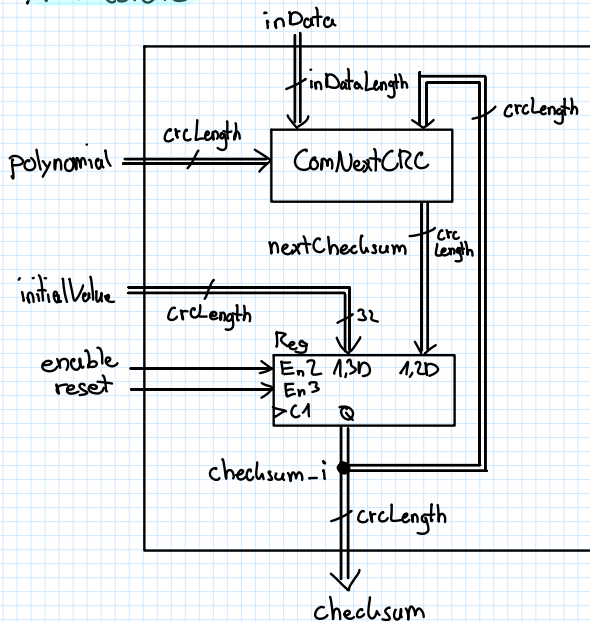
CRC

Entity



Generische Parameter:
 crcLength
 inDataLength

Architecture



crc_axi_master

Entity

Port	In/Out	Type	Beschreibung
CLK	in	std_logic	
RESETN	in	std_logic	
AXI_Interface	in/out		
write	in	std_logic	1: Schreibzugriff; 0: Lesezugriff
start	in	std_logic	Startet Lese-/Schreibvorgang
addr_axi	in	std_logic_vector	Adresse für AXI-Master Zugriff
size	in	std_logic_vector	Anzahl der zu Lesenden/schreibenden Wörter
idle	out	std_logic	1: Komponente führt Lese-/Schreibzugriff aus
raddr	out	std_logic_vector	
rdata	in	std_logic_vector	
re	out	std_logic	
waddr	out	std_logic_vector	
wdata	out	std_logic_vector	
we	out	std_logic	

↳ liefert eine Speicheradresse, fängt an nach Erhalt des Start Signals einen Schreib- oder Leseburst durchzuführen

Crc_axi_ram

↳ Block-RAM

Port	In/Out	Type	Beschreibung
CLK	in	std_logic	
Waddr	in	std_logic_vector	
Wdata	in	std_logic_vector	
WE	in	std_logic	
Raddr	in	std_logic_vector	
Rdata	out	std_logic_vector	
re	in	std_logic	

crc_axi_lite

Port	In/Out	Type	Beschreibung
S_AXI_L	in/out		AXI-Lite Slave Schnittstelle
run	out	std_logic	Startsignal wenn Run Bit auf 1
stop	out	std_logic	Stoppsignal wenn Run Bit auf 0
status	in	std_logic	
interrupt_enable	out	std_logic	
interrupt_status	in	std_logic	
interrupt_reset	out	std_logic	
raddr	out	std_logic_vector	
waddr	out	std_logic_vector	
packet_size	out	std_logic_vector	
packet_number	out	std_logic_vector	
polynomial	out	std_logic_vector	
initial_value	out	std_logic_vector	

