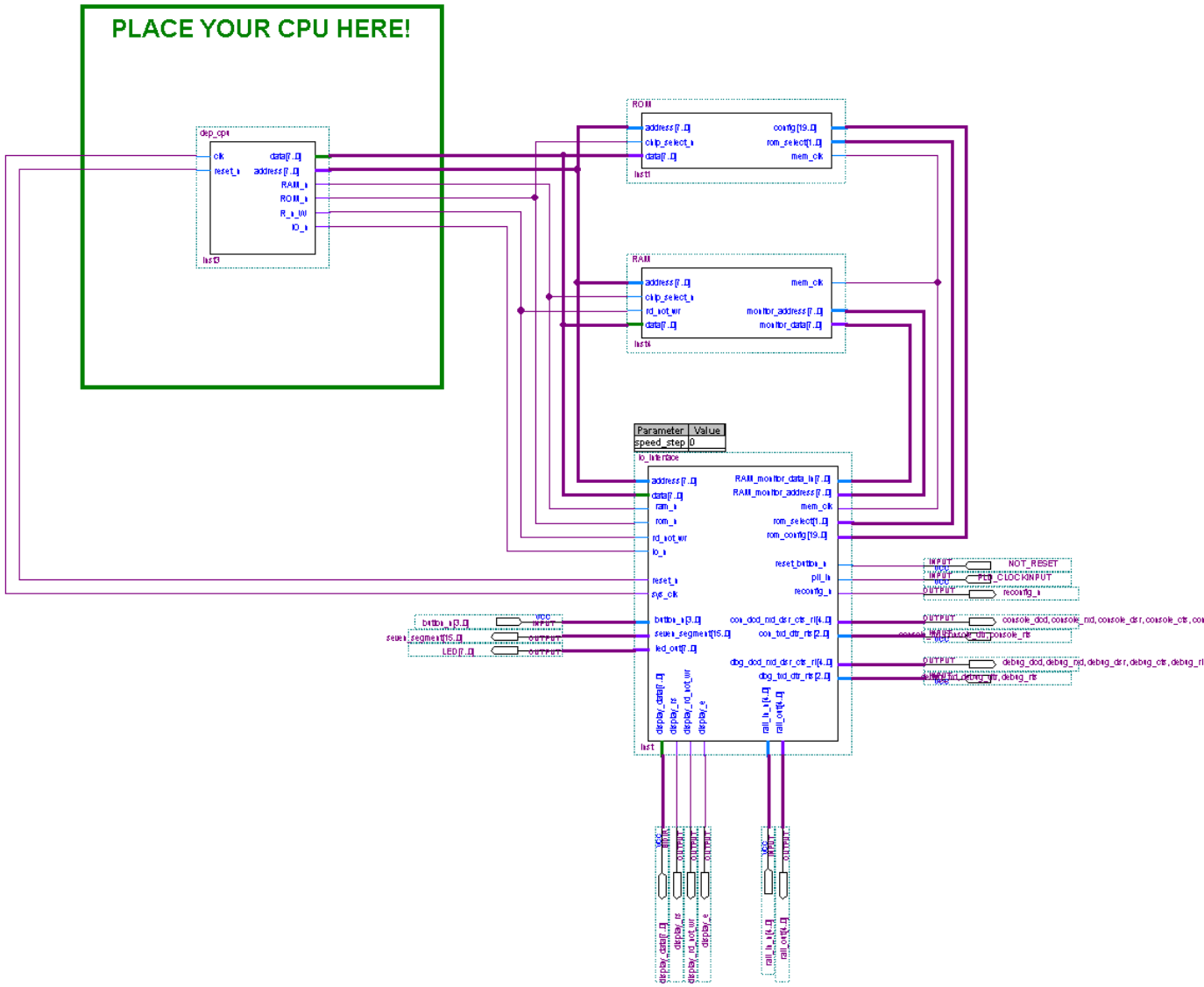

Abbildung 1: DEP-CPU-Vorlage mit bereits eingebauter CPU

1 RAM (RAM.vhd)

Der RAM-Baustein enthält 256 8bit-Zellen, die von der CPU les- und schreibbar sind. Die Realisierung erfolgt als *dual port ram*, so dass Schreib- und Lesezugriffe der CPU parallel zu der *Monitorfunktion* der Siebensegmentanzeige (s.u.) stattfinden können.
Anmerkung: Der Speicher kann zu Testzwecken (CPU-Debugging) teilweise oder vollständig initialisiert werden. Hierzu wird die Initialisierungsdatei `ram.mif` im Verzeichnis `INIT` benötigt.

2 ROM (ROM.vhd)

Der ROM-Baustein enthält 4 ROMs à 256 8bit-Zellen, die über den CPU-Bus nur lesbar sind. Alle vier ROMs können mit unterschiedlichen Programmen initialisiert werden. Während nach jedem Reset standardmäßig ROM 0 benutzt wird, können durch den *ROM-Umschalter* (s.u.) abwechselnd alle vier ROMs zum Einsatz kommen. Dadurch erhöht sich der Programmspeicher auf 1kB. Die ROMs enthalten standardmäßig den Inhalt der Initialisierungsdateien `ram0.mif` bis `ram3.mif` im Verzeichnis `INIT`. Diese Dateien enthalten das CPU-Testprogramm.
Das Initialisieren und Umschalten der ROMs ist weiter unten beschrieben.
Anmerkung: Zum Lösen der Standardaufgaben ist die Benutzung mehrerer ROMs nicht erforderlich!

3 E/A-Schnittstelle (io_interface.vhd)

3.1 Taktgenerator (main_pll_configurable.vhd)

Der Taktgenerator ist im E/A-Baustein enthalten. Die Taktfrequenz kann über den Parameter *speed_step* eingestellt werden. Der daraus resultierende Systemtakt berechnet sich gemäß:

$$f_{clk} = 30\text{MHz} \times 2^{-speed\_step}, speed\_step \in \{0, \ldots, 31\}\,.$$

Zusätzlich kann *speed_step* per Software über Port 0xFF eingestellt werden:
Ein Wert zwischen 0 und 31 wird direkt als neuer *speed_step* übernommen.
Der Ausgabewert 0x20 bewirkt, dass *speed_step* von nun an über die Tasten in Binärdarstellung gesteuert wird ($speed\_step[4..0] = button\,[3..0] \times 2$).
Beim Ausgabewert 0x40 wird der Tastenstatus nur einmal abgefragt und *speed_step* dementsprechend gesetzt.
Mit 0x80 wird *speed_step* auf den voreingestellten Parameterwert zurückgesetzt.
Anmerkung: Es ist zu beachten, dass die Bahn (genauer: die Reed-Kontakte) höchstens im kHz-Bereich zuverlässig arbeitet.

3.2 ROM-Lader (rom_loader.vhd)

Eigene Programme müssen mittels des ROM-Laders ins ROM bzw. in die ROMs geladen werden. Hierzu werden Binärdateien (keine MIF-Dateien!) benötigt, die vom DEP-Assembler aus dem Programmquelltext erzeugt werden.
Die Übertragung erfolgt über die serielle Schnittstelle `COM1:` des PCs, die mit der seriellen Schnittstelle `Console` des FPGA-Boards verbunden ist.
Zur Übertragung kann man die entsprechende Option im DEP-Assembler-Editor nutzen oder die BIN-Datei manuell von der Konsole aus übertragen durch folgende Eingabe:

```
> copy <file>.bin com1:
```

Sollen mehrere ROMs beschrieben werden, so müssen alle BIN-Dateien direkt hintereinander übertragen werden:

```
> copy <file0>.bin + <file1>.bin + [[<file2>.bin] + <file3>.bin] com1:
```

Die Übertragung ist beendet, sobald die Siebensegmentanzeige den Hexadezimalcode des ersten Befehls im geladenen Programm anzeigt. Nun kann die CPU durch Drücken der Reset-Taste reaktiviert werden.
Sollte die Siebensegmentanzeige „PE“ oder „FE“ anzeigen, so ist bei der Übertragung ein Fehler (Parity Error bzw. Framing Error) aufgetreten. Nach Drücken der Reset-Taste kann die Übertragung wiederholt werden.
Anmerkung: Sollte die Übertragung trotz mehrerer Versuche nicht gelingen, sollten die Einstellungen der seriellen Schnittstellen kontrolliert werden:
Zur Übertragung muss die `COM1:`-Schnittstelle wie folgt konfiguriert sein:
57600 baud, 8 Datenbits, Parität: even, 1 Stoppbit, Flusskontrolle: Hardware.

3.3 ROM-Umschalter (rom_selector.vhd)

Das aktuelle Programm kann in einem der vier ROMs enthalten sein. Nach jedem Reset wird ROM 0 als aktueller Programmspeicher gesetzt. Während des Programmablaufs kann zu einem anderen ROM umgeschaltet werden. Dies geschieht durch Ausgabe der ROM-Nummer (0 bis 3) an I/O-Port 0xFE.
Das Verketten mehrerer ROMs zu einem großen geschieht am einfachsten, wenn sich der ROM-Umschaltbefehl in der letzten Zeile des aktuellen ROMs befindet. Dadurch befindet sich der Programmzähler nach dem Umschalten bei 0 und das Programm des neuen ROMs wird ab der ersten Zeile abgearbeitet.
Anmerkung: Der Programmzähler (PCR) der CPU gilt für alle ROMs und wird beim Umschalten nicht beeinflusst. Dies ist bei der Programmierung mehrerer ROMs zu berücksichtigen!

3.4 Ein-/Ausgabe

3.4.1 Bahn (rail_interface.vhd)

Die Abfrage des Bahnstatus' und die Ausgabe der Steuersignale geschieht über Port 0x80. Das Format der Ein- und Ausgaben ist im DEP-Skript beschrieben.

3.4.2 LC-Display (display_interface.vhd)

Das Display erfordert keine Initialisierung seitens des Benutzers. Es können sofort Zeichen in Form von ASCII-Codes oder spezielle Befehle ausgegeben werden. Die Ausgabe von Zeichen erfolgt über Port 0x40, Befehle werden über Port 0x41 an das Display gesandt.
Die wichtigsten Befehle sind:
0x1 = CLEAR_CMD
0x2 = HOME_CMD
0x80 + ADDR = SET_POS (s. Tabelle 1)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Tabelle 1: Addressierung der Displaypositionen (hexadezimal)

Anmerkung: Da zwischen dem Ende der ersten Zeile und dem Anfang der zweiten Zeile eine Lücke im Adressraum ist, gehen 24 Zeichen verloren, wenn der Cursor nicht explizit an den Anfang der zweiten Zeile gesetzt wird.

3.4.3 Siebensegmentanzeige (seven_segment_interface.vhd)

Die Siebensegmentanzeige verfügt über verschiedene Modi, die über Port 0x21 gesetzt werden können:
0 = Abschalten der Siebensegmentanzeige,

1 = Datenbus-Monitor,
2 = Adressbus-Monitor,
3 = Anzeige eines beliebigen Wertes,
4 = RAM-Monitor einer beliebigen RAM-Adresse,
5 = Bahn-Monitor,
6 = Ansteuern beliebiger Segmente der linken Ziffer,
7 = Ansteuern beliebiger Segmente der rechten Ziffer.
Die anzuzeigenden Werte in den Modi 3, 4, 6 und 7 müssen an Port 0x20 ausgegeben werden.
Die Zuordnung der Segmente in den Modi 5 bis 7 ist in Abbildung 2 zu sehen:

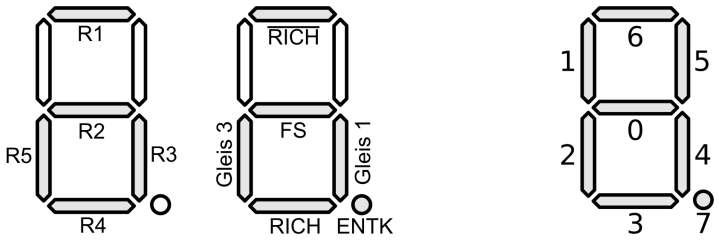

Abbildung 2: links: Zuordnung der Bahnsignale zu den Segmenten im Modus 5
rechts: Zuordnung der Datenbits zu den Segmenten in den Modi 6 und 7

3.4.4 LEDs (led_interface.vhd)

Die LED-Schnittstelle verfügt über verschiedene Modi, die über Port 0x11 gesetzt werden können:
0 = Abschalten der LEDs,
1 = Datenbus-Monitor,
2 = Adressbus-Monitor,
3 = Binärdarstellung eines beliebigen Wertes.
Der anzuzeigende Wert wird an Port 0x10 ausgegeben.

3.4.5 Tasten (button_interface.vhd)

Über Port 0x30 kann der Status der vier Tasten abgefragt werden. Dabei zeigen die vier niederwertigen Bits den aktuellen Zustand an, während die höherwertigen Bits besagen, ob die Tasten seit der letzten Abfrage gedrückt worden sind. Mit jeder Statusabfrage werden diese Bits zurückgesetzt.

3.5 Sonstiges

3.5.1 Stack (lifo.vhd)

Der Stack kann 64 Elemente speichern. Die Elemente werden über Port 0x60 auf den Stapelspeicher gelegt (*push*) und können durch Lesen von demselben Port wieder entfernt werden (*pop*). Lesen ohne Entfernen eines Elements (*peek*) ist über Port 0x61 möglich. Über Port 0x62 kann der Status abgefragt werden:
Bits 7 und 6 sind gesetzt, falls der Stack voll bzw. leer ist. Die Bits 5 bis 0 geben die Anzahl der enthaltenen Datenwörter an.
Durch Ausgabe von 1 an Port 0x62 wird der Stack gelöscht.
Ein voller Stack wird durch Schreiboperationen nicht verändert. Ein leerer Stack liefert beim Lesen einen undefinierten Wert.

3.5.2 Queue (fifo.vhd)

Die Queue kann 64 Elemente speichern. Die Elemente werden über Port 0x50 in die Warteschlange geschoben und können durch Lesen von demselben Port wieder entfernt werden. Lesen ohne Entfernen eines Elements ist über Port 0x51 möglich. Über Port 0x52 kann der Status abgefragt werden:
Bits 7 und 6 sind gesetzt, falls die Warteschlange voll bzw. leer ist. Die Bits 5 bis 0 geben die Anzahl der enthaltenen Datenwörter an.
Durch Ausgabe von 1 an Port 0x52 wird die Warteschlange gelöscht.
Eine volle Warteschlange wird durch Schreiboperationen nicht verändert. Eine leere Warteschlange liefert beim Lesen einen undefinierten Wert.

3.5.3 Release-ID (release_id.vhd)

Dieser Baustein enthält eine Konstante, die zum Testen des IN-Befehls der CPU benutzt werden kann. Sie kann über Port 0xF0 abgefragt werden.

3.5.4 Zähler (counters.vhd)

Diese Komponente stellt vier 32bit-Zähler 0 bis 3 zur Verfügung, deren Werte byteweise über die Ports 0x70 bis 0x7F abgefragt werden können. Dabei ergibt sich der Port aus:

$$port = 0x70 + 4 \times counter\_index + byte\_index$$

Die Zählimpulse richten sich nach den Zählklermasken, die für die einzelnen Zähler über die Ports 0x70, 0x74, 0x78 bzw. 0x7C gesetzt werden.
Die einzelnen Bits der Zählermaske sind in Tabelle 2 definiert:

Datenbit	7	6	5	4	3	2	1	0
Bedeutung	CLR	ALL	INS	ROM	RAM	IO	READ	WRITE

Tabelle 2: Definition der Zählermaske

Bei gesetztem CLR-Bit wird der Zählerstand vor dem Zählvorgang gelöscht.
Bei gesetztem ALL-Bit wird der Zählerstand in jedem Takt hochgezählt.
Bei gesetztem INS-Bit (und nicht gesetztem ALL-Bit) wird beim Holen der Instruktion aus dem ROM hochgezählt, was der Anzahl der abgearbeiteten Programmzeilen entspricht.
Die nachfolgenden Bits sind nur von Bedeutung, wenn ALL- und INS-Bit nicht gesetzt sind:
Die ROM-, RAM- und IO-Bits spezifizieren in Verbindung mit den READ- und WRITE-Bits, welche Zugriffe gezählt werden. Dabei müssen sowohl die Komponente, auf die zugegriffen wird, als auch die Art des Zugriffs stimmen, damit ein Zählimpuls ausgelöst wird.
Beispiel: Zum Zählen aller Lesezugriffe setzt man die Maske auf%00011110, während alle Speicherzugriffe durch %00011011 maskiert werden.