High-Level-Synthese einer ATM-Switch-Steuerung mit Monet

Walter Lange Wolfgang Rosenstiel

WSI 98-14

11. Januar 1999

Wilhelm-Schickard-Institut
Universität Tübingen
D-72076 Tübingen, Germany
e-mail: wlange@informatik.uni-tuebingen.de

ISSN 0946-3852

High-Level-Synthese einer ATM-Switch-Steuerung mit Monet

Walter Lange Wolfgang Rosenstiel

Dezember 1998

Zusammenfassung

In der vorliegenden Arbeit werden mit Hilfe des handelsüblichen High-Level-Synthese-Werkzeugs Monet^{TM} der Firma Mentor Graphics^{TM} Schaltungen für eine ATM-Switch-Steuerung synthetisiert. Zweck der Untersuchung ist es festzustellen, ob es möglich und sinnvoll ist, Schaltungen für schnelle Datenübertragung mit Hilfe von HLS-Werkzeugen zu erstellen. Die Syntheseergebnisse werden tabellarisch dargestellt.

Inhaltsverzeichnis

1 Einleitung **4**
- 1.1 High-Level-Synthese . . . 4
  - 1.1.1 Grundlagen . . . 4
  - 1.1.2 Ablauf der Synthese . . . 5
- 1.2 Die Anwendung . . . 6
- 1.3 Die HLS-Technologie-Bibliothek . . . 6

2 Das HLS-Werkzeug Monet von Mentor **7**
- 2.1 Regeln für VHDL-Beschreibungen für die Synthese mit Monet . . . 7
  - 2.1.1 Synchrones Warten auf ein Signal . . . 7
  - 2.1.2 Synchronisation der Datenannahme . . . 7
  - 2.1.3 Die Eingabe-Ausgabe-Modi von Monet . . . 8
  - 2.1.4 Zurücksetzen und Initialisieren der Schaltung (Reset) . . . 8
- 2.2 Randbedingungen der Synthese: . . . 8
- 2.3 Simulation der Netzliste nach der Synthese . . . 9
- 2.4 Verwendete Versionen: . . . 9

3 Zuverlässigkeit der Synthese **9**

4 Das AHT-Eingangsmodul AHT_IN **10**
- 4.1 Synthese mit Monet . . . 10
  - 4.1.1 Behandlung der Schleife L1. . . . 10
  - 4.1.2 Gantt-Charts und FSM's: . . . 11
  - 4.1.3 Synthese-Daten . . . 11

5 Das Zellkopfübersetzungsmodul HT **15**
- 5.1 Synthese mit Monet . . . 15
  - 5.1.1 Gantt-Chart und FSM . . . 15
  - 5.1.2 Synthese-Daten für den Prozeß "htproc" . . . 16

6 Die Routingsteuerung RC **18**
- 6.1 Synthese mit Monet . . . 19
  - 6.1.1 Synthese-Daten . . . 19

7 Die Schieberegister-Steuerung SRC **22**
- 7.1 Synthese mit Monet . . . 23
  - 7.1.1 Schleifen-Behandlung . . . 23
  - 7.1.2 Gantt-Charts und FSM's: . . . 24
  - 7.1.3 Synthese-Daten . . . 24

8 Die Verbindungs-Steuerung CC **27**
- 8.1 Synthese mit Monet . . . 27
  - 8.1.1 Schleifen-Behandlung . . . 28
  - 8.1.2 Gantt-Chart . . . 28

8.1.3 Synthese-Daten 28

9 Das Datenannahme-Modul RD **30**
9.1 Synthese mit Monet 30
9.1.1 Synthese-Daten 31

10 Das Ausgangsmodul AHT_OUT **35**
10.1 Synthese mit Monet 35
10.1.1 Behandlung der Schleifen L1 und L2. 36
10.1.2 Gantt-Charts und FSM's: 36
10.1.3 Synthese-Daten 37

11 Zusammenfassung **39**
11.1 Verwendete Bibliotheken und Parametereinstellungen 40
11.2 Ergebnisse der Synthese 40
11.2.1 Fläche, Verzögerung, FSM 40
11.2.2 Rechenzeit, Speicherbedarf und Zuverlässigkeit 42
11.2.3 Dokumentation und Einarbeitungszeit 43

12 Anhang: Quellcodes und Berichte (reports) der High-Level-Synthese **44**
12.1 Eingangsmodul AHT_IN 44
12.2 Zellkopfübersetzungsmodul HT 50
12.3 Routing-Steuerung (RC) 54
12.4 Schieberegister-Steuerung (SRC) 62
12.5 Verbindungs-Steuerung (CC) 68
12.6 Datenaufnahme-Modul RD 74
12.7 AHT-Ausgangs-Modul 82

13 Literatur **89**

1 Einleitung

High-Level-Synthese-Werkzeuge sind erst seit wenigen Jahren auf dem Markt. Nachdem 1995 der "Behavioral Compiler^{TM}" der Fa. Synopsys^{TM} angeboten wurde, erschien 1998 "Monet^{TM} " der Fa. Mentor Graphics. Die Aufnahme der HLS-Werkzeuge von den Hardware-Entwicklern geschieht nur zögerlich, man scheut die Einarbeitungszeit und kann auch nicht abschätzen, ob der Einsatz der Werkzeuge zu einem Erfolg führt. Es gibt Beispiele für gute Synthesen von Verhaltensbeschreibungen von Matritzenberechnungen (MPEG-Komprimierung) oder Differentialgleichungen. Untersuchungen über den effektiven Einsatz von General-Purpose-HLS-Werkzeugen für die Synthese von Steuerschaltungen für Hochgeschwindigkeitsnetze sind noch nicht bekannt.

In der vorliegenden Arbeit wird eine Steuerschaltung für einen ATM-Switch (die ATM-Switch-Steuerung ASS) mit einem handelsüblichen HLS-Werkzeug synthetisiert, die Ergebnisse werden tabellarisch dargestellt. Die Entwicklung der ASS ist am Wilhelm Schickard-Institut, am Lehrstuhl für Technische Informatik durchgeführt und im Internen Bericht: "Modellierung einer ATM-Switch-Steuerung" [LaRo97] ausführlich beschrieben worden.

Der in diesem Bericht häufig verwendete Namen "Monet" ist ein Handelsname.

Die Arbeit gliedert sich wie folgt: Im nächsten Abschnitt wird auf die High-Level-Synthese eingegangen, danach wird kurz die Anwendung: die ATM-Switch-Steuerung beschrieben. Im Kapitel 2 wird das verwendeten Synthesewerkzeug Monet vorgestellt. In den Kapiteln 4 bis 10 wird die Synthese der einzelnen ATM-Module beschrieben. In der Zusammenfassung sind die Syntheseergebnisse tabellarisch dargestellt und werden diskutiert. Im Anhang sind die VHDL-Quellcodes der Module, die Testtreiber und die Syntheseberichte aufgelistet.

1.1 High-Level-Synthese

1.1.1 Grundlagen

Die High-Level-Synthese erzeugt aus der Verhaltensbeschreibung einer Schaltung (z.B.: in der Hardware-Beschreibungssprache VHDL) eine RT- (Register-Transfer) Struktur. Die RT-Struktur besteht aus einem Datenfluß-Teil und einem Kontrollfluß-Teil, der als FSM (Finite State Machine) ausgeführt ist [DeMi94] [Gaj92]. Der Datenfluß enthält die Lese-Operationen der Eingangssignale, die Datenverarbeitung derselben und die Schreiboperationen der Ausgangssignale.

Der Kontrollfluß definiert die zeitliche Abfolge in diskreten Taktschritten (Scheduling) der Datenflußoperationen.

Die High-Level-Synthese (HLS) wird von den heutigen HLS-Werkzeugen im wesentlichen in drei Schritten durchgeführt:

1. Allokation: Hier werden die Komponenten für die Schaltung selektiert.
2. Scheduling: Die Kontrollschritte werden festgelegt.
3. Assignment (oder Binding): Die in der Allokation definierten Komponenten werden den Operationen in den einzelnen Kontrollschritten zugeordnet.

Ein großer Vorteil für den Schaltungsentwickler ist, daß bei Einsatz eines HLS-Werkzeugs nicht nur die Selektion der einzelnen Komponenten vom Werkzeug durchgeführt wird, sondern auch die Entwicklung des Controllers (Finite State Machine, FSM) automatisch erfolgt. Der Entwickler kann sich auf die effiziente Beschreibung des Datenflusses und auf die Optimierung der Schaltung konzentrieren.

1.1.2 Ablauf der Synthese

Die High-Level-Synthese wird sowohl für BC als auch für Monet in folgenden Schritten durchgeführt:

1. Modifiziere die simulierte VHDL-Beschreibung in folgender Hinsicht:
   (a) Festlegung des “Reset” und entsprechende Kodierung. In unserem Fall ist ein synchrones Zurücksetzen ausreichend.
   (b) Festlegung des Eingabe-Ausgabe-Modus (siehe Kapitel: “Eingabe-Ausgabe-Modi”). In unserem Fall ist der “superstate I/O-Modus” angebracht.
   (c) Festlegung für das Aufrollen von Schleifen. In unserem Fall ist es meist angebracht, die Schleifen nicht aufzurollen, d.h. da das Aufrollen meist als Vorgabe (Default) genommen wird, muß ein entprechendes Attribut gesetzt werden. (z.B.: don’t_unroll loop “label”).
2. Simulation des Moduls mit Hilfe eines Testtreibers. Damit wird die Funktionalität überprüft.
3. Synthese des Moduls mit folgenden Parametern:
   (a) I/O-Modus: “superstate I/O”.
   (b) Ansiedlung im Entwicklungsraum: (Design Space): Die Schaltung mit geringer Verzögerung hat Vorrang vor geringer Fläche. Für Monet wird bei den “Allocation Constraints” “fastest” aus drei Stufen (fastest, smallest area, custom) selektiert.
   (c) Festlegung der Taktperiode. In unserem Fall wird jeweils ein Taktperiode von 25 ns gewählt.

   Das Ergebnis ist eine Strukturbeschreibung auf RT-Ebene in VHDL.
4. Simulation des synthestisierten Moduls mit Hilfe des oben verwendeten Testtreibers. Damit ist eine Überprüfung der Funktionalität der synthetisierten Schaltung möglich.

Die Simulation nach der Synthese mit anschließender Überprüfung der Funktionalität ist eine notwendige, wenn auch nicht hinreichende Bedingung für die Korrektheit der synthetisierten Schaltung.

1.2 Die Anwendung

Als Anwendung wird eine in der Hardwarebeschreibungssprache VHDL modellierte ATM-Switch-Steuerung verwendet ([LaRo97]). Auf dem Gebiet der Informationsübertragung ist ATM (Asynchroner Transfer Modus) eine weitverbreitete Übertragungsmethode. Die Datenübertragung erfolgt in Datenpaketen konstanter Länge (53 Bytes). Der hohen Durchsatzrate wegen werden Hardware-Lösungen bei der Steuerung der ATM-Schaltungen benötigt. Das Blockschaltbild (Abbildung 1) zeigt einen Kanal einer ATM-Übertragungsleitung in der sog. ATM-Schicht. Die Funktionen der einzelnen Module (AHT_IN, HT/RT, RC, SRC, CC, RD und AHT_Out) sind in [LaRo97] definiert und die Synthese dieser Module wird in den folgenden Kapiteln beschrieben.

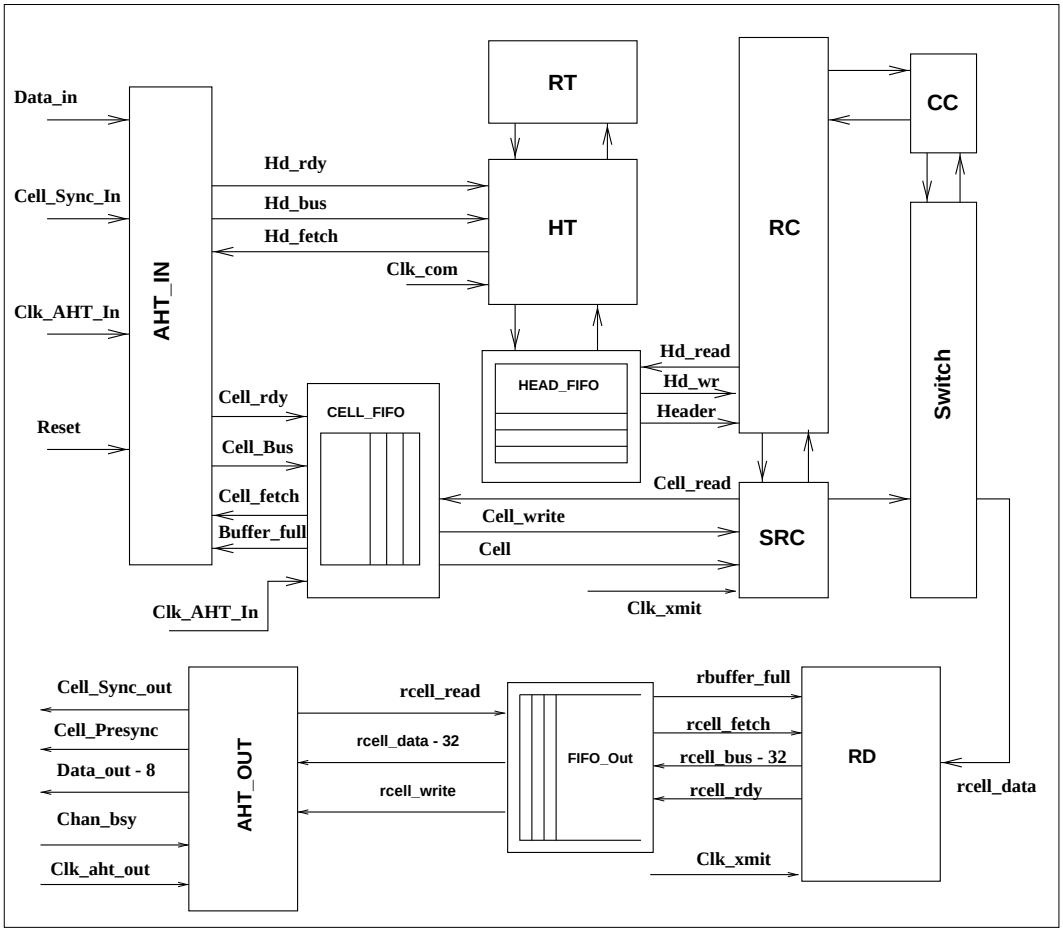


Abbildung 1: Blockschaltbild eines Kanals der ATM-Switch-Steuerung

1.3 Die HLS-Technologie-Bibliothek

Monet benötigt eine Technologie-Bibliothek (HLS Library) für die Abschätzung von Chipfläche (area) und Verzögerung (delay) und für die Erstellung der RTL-Netzliste als

Ergebnis der High-Level-Synthese. In dieser Bibliothek ist eine Sammlung von meist generischen Komponenten basierend auf einer bestimmten Technologie (z.B. FPGA's, LCA's), die mit Attributen über Chipfläche und Verzögerung versehen sind, enthalten. Da die HLS-Bibliotheken nicht genormt sind, hat jedes Werzeug sein eigenes Bibliotheksformat, das anderen Werkzeugen nicht zugänglich ist.

Für Monet werden folgende Bibliotheken eingesetzt:

- Die Komponenten werden den Bibliotheken "mgc_lca300k_comp_dc.lib" "mgc_lca300k_dmag_dc.lib" entnommen.
- Die Ziel Technologie ist eine LCA-Technologie. Die Flächeneinheit 1 wird durch den 1-Bit Inverter dargestellt.

2 Das HLS-Werkzeug Monet von Mentor

Monet zeichnet sich durch eine übersichtliche graphische Oberfläche aus (siehe [Mum98]) [Montw98]). Der Benutzer wird gut durch den gesamten Syntheseprozeß geführt. Die einzelnen Syntheseschritte sind sozusagen miteinander verbunden und werden damit zwangsweise in der richtigen Reihenfolge ausgeführt. Nach jedem Syntheseschritt sind Berichte und Abspeicherung der Ergebnisse möglich.

2.1 Regeln für VHDL-Beschreibungen für die Synthese mit Monet

2.1.1 Synchrones Warten auf ein Signal

Für jedes synchrone Warten auf ein Signal z.B.:
"WAIT UNTIL Clk_com'EVENT and Clk_com = '1'and Hd_rdy = '1';"
wird eine Schleife wie folgt geschrieben:

```
hd_rdy_loop : loop
    wait until clk_com'event and clk_com = '1';-- ------------
    exit hd_rdy_loop when ( hd_rdy = '1' );
end loop hd_rdy_loop;
```

In der Schleife erscheint die "Warteanweisung" auf den Takt, und das Warten auf das Signal (hier "hd_rdy"). Ein zusätzliches Warten auf ein "Reset", d.h. ein zusätzliches "exit" in der obigen Schleife kann zu einem Problem führen. Daher wird für das Zurücksetzen das "globale Reset" gewählt.

2.1.2 Synchronisation der Datenannahme

"Synchronisieren der Datenannahme" bedeutet, Daten von einer Datenleitung (hier z.B.: "Data_In") dann in ein Register zu übernehmen, wenn ein Synchronisationssignal (hier z.B.: das Zellstartsignal "Cell_Sync_In") zusammen mit der positiven Flanke des Taktes aktiv wird.

Bei Monet erreicht man das dadurch, daß in die Warteschleife auf ein Signal (s.o.) die Datenübernahme z.B.:

```
Hd_reg(31 downto 24) := Data_In;
```

wie folgt mit hineingenommen wird.

```
cell_sync_loop: loop     -- fuer monet
      wait until clk_aht_in'event and clk_aht_in = '1';
      Hd_reg(31 downto 24) := Data_In;
      exit cell_sync_loop when (Cell_Sync_in = '1');
    end loop cell_sync_loop;
```

2.1.3 Die Eingabe-Ausgabe-Modi von Monet

Die Eingabe-Ausgabe-Modi sind wie folgt:

1. "Cycle fixed" I/O-Modus: Die einzelnen Kontrollschritte werden zwingend vom Entwickler festgelegt.
2. "Superstate fixed" I/O-Modus: "Superstates" sind Zustände zwischen zwei "Wait"-Anweisungen, in die der Scheduler noch weitere Kontrollschritte, wenn nötig, einfügt. Dieser Modus wird für unsere Anwendung gewählt. Er ist für Module die asynchron miteinader kommunizieren am günstigsten.
3. "Free floating" I/O-Modus: Hier ist es dem Scheduler vollkommen freigestellt, wie die Kontrollschritte gesetzt werden. Dieser Modus kann für unsere Anwendung nicht gewählt werden.

2.1.4 Zurücksetzen und Initialisieren der Schaltung (Reset)

Es gibt die Möglichkeit des expliziten oder impliziten synchronen Zurücksetzens der Schaltung. Beim expliziten synchronen Zurücksetzen muß eine gesonderte "Reset_ Loop" beschrieben werden und nach jeder "Wait"-Anweisung folgende Exit-Anweisung eingesetzt werden:

```
exit Reset\_loop when reset = '1';
```

Einfacher ist das implizite synchrone Zurücksetzen. Im Deklarationsteil der Architektur einer Schaltungsbeschreibung wird das Attribut "sync_reset" zusammen mit der Bezeichnung und der Phase (positiv oder negativ) des Reset-Signals eingeführt. Damit kann mit einem "Reset"-Signal nach jedem Zustand eine Rückkehr zum Zustand 1 der FSM erreicht werden.

2.2 Randbedingungen der Synthese:

Für alle Module werden folgende Randbedingungen für die Synthese mit Monet gewählt:

1. I/O-Modus: "superstate fixed".
2. Scheduling Constraint: "fastest": (Lösung mit geringster Verzögerung)
3. Taktzyklus: 25 ns mit 10% Overhead.

4. Zieltechnologie:

   mgc_lca300k_dmag_dc.lib

   mgc_lca300k_comp_dc.lib.

2.3 Simulation der Netzliste nach der Synthese

Die Simulation der Netzliste nach der Synthese wird mit dem Simulator "Quicksim^{TM}" der Fa. Mentor durchgeführt. Die Simulation gilt als notwendige Überprüfung der Korrektheit der synthetisierten Schaltung.

2.4 Verwendete Versionen:

Bis 30. 7. 1998: Version R_31, danach R_33.

3 Zuverlässigkeit der Synthese

Um die Zuverlässigkeit des Synthesewerkzeugs beurteilen zu können, wir in diesem Zusammenhang ein Zuverlässigkeitsmaß Z wie folgt definiert:

1. **Z = 10:** Die Synthese wurde **fehlerlos** abgeschlossen. Das Ergebnis der Synthese (die Beschreibung der Schaltung auf RT-Ebene) ist nach Abschätzung des Entwicklers **akzeptabel** und **korrekt simulierbar,** d.h. die Simulation ist funktional gleich wie die Simulation der Verhaltensbeschreibung vor der Synthese. Das Ergebnis der Synthese kann dem nächsten Syntheseschritt (der RT-Synthese) zugeführt werden.

2. **Z = 8:** Die Synthese bricht mit einem **Programmfehler** ab. Durch Ändern der Verhaltensbeschreibung wird eine Umgehung des Fehlers erreicht. Dadurch wird das Ergebnis der Synthese **akzeptabel** und ist **korrekt** simulierbar. Das Ergebnis der Synthese kann dem nächsten Syntheseschritt zugeführt werden.

3. **Z = 5:** Die Synthese bricht mit einem Programmfehler ab. Durch Ändern der Verhaltensbeschreibung wird eine Umgehung des Fehlers erreicht. Dadurch wird das Ergebnis der Synthese nach Abschätzung des Entwicklers **schlechter** als bei Z = 10 oder Z = 8, jedoch das Ergebnis ist **korrekt** simulierbar. Das Ergebnis der Synthese kann dem nächsten Syntheseschritt zugeführt werden.

4. **Z = 2:** Die Synthese bricht mit einem Programmfehler ab. Es wird nur ein Teilergebnis erreicht, das zwar Aufschlüsse über die zu erwartende Schaltung ergibt, aber das Ergebnis der Synthese kann nicht weiter synthetisiert werden.

5. **Z = 0:** Die Synthese bricht mit einem Programmfehler ab. Es gibt keine brauchbaren Ergebnisse.

4 Das AHT-Eingangsmodul AHT_IN

Das AHT-Eingangsmodule AHT_IN nimmt Zellen aus der Physikalischen Schicht über die 8-Bit breite Datenleitung Data_in auf (siehe Bild 1 oder [LaRo97]).

Das erste Datenbyte kommt synchron mit dem Signal Cell_Sync_In und die restlichen Bytes der ATM-Zelle erscheinen synchron mit der positiven Flanke des Taktes Clk_AHT_in. Da die Daten kontinuierlich eintreffen, ist es wichtig, daß das Modul AHT_In auch in jedem Takt für Daten aufnahmenbereit ist.

Das Modul AHT_In umfaßt zwei Prozesse: den Prozeß “AHTIN” und den Prozeß “Payload”. Der AHTIN-Prozeß nimmt den 5 Byte großen Zellkopf an und gibt davon 4 Bytes (ohne HEC-Byte: Header Error Control) im Two-Way-Handshake-Modus an das Zellkopfübersetzer-Modul “HT” weiter. Der Payload-Prozeß nimmt die 48 Bytes der Nutzlast an und gibt sie an den CELL_FIFO-Speicher weiter.

4.1 Synthese mit Monet

Um eine erfolgreiche Synthese zu erreichen, muß der VHDL-Quellcode der Spezifikation entsprechend den Monet-Richtlinien geändert werden. Dies trifft hauptsächlich zu für synchrone WAIT’s auf Signale und für das synchrone Zurücksetzen (Reset).

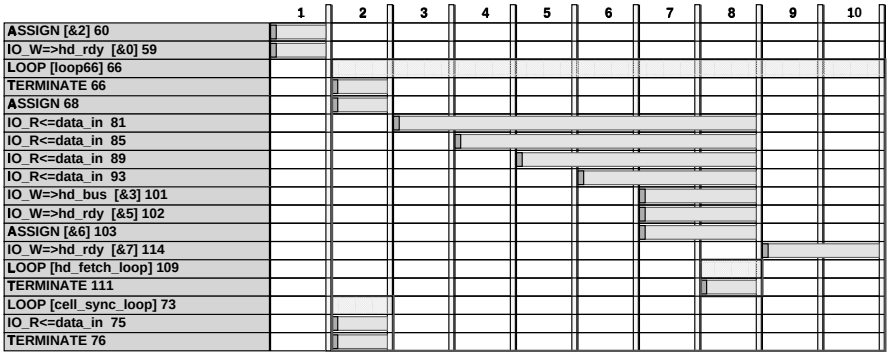


Abbildung 2: Gantt-Chart des Prozesses AHTIN. Die Spalten 1 bis 10 stellen die Kontrollzyklen des Prozesses dar.

4.1.1 Behandlung der Schleife L1.

Im “Payload”-Prozeß werden in einer Schleife (L1) jeweils 4 Bytes zu einem Wort zusammengefaßt und an den CELL_FIFO-Speicher übergeben. Die Nutzlast umfaßt 48 Bytes, d.h. diese Schleife wird zwölfmal durchlaufen.

Ein Aufrollen der Schleife bedeutet, daß die Steuereinheit, (die Control-FSM) sehr viele Zustände (hier 51) erhält.

Daher wird hier zunächst versucht, die Schleife **nicht** aufzurollen. Die FSM wird einfacher, sie erhält nur 7 Zustände, allerdings werden die Daten nicht kontinuierlich zusammengefaßt, es entsteht eine Datenlücke, da für jede Schleifenausführung ein zusätzlicher

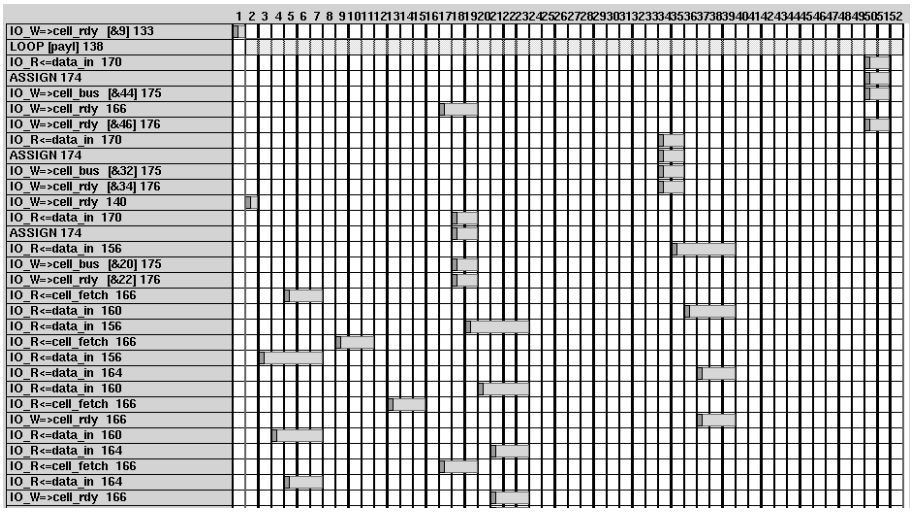


Abbildung 3: Oberer Teil der Gantt-Chart des Payload-Prozesses.

Kontrollzyklus aufgewendet wird. Aus diesem Grund wird die Lösung mit aufgerollter Schleife gewählt.

4.1.2 Gantt-Charts und FSM's:

Die Abbildung 2 zeigt die sog. Gantt-Chart des Prozesses AHTIN, die Abbildung 3 zeigt den oberen Teil der Gantt-Chart des Payload-Prozesses nach der Synthese.

Die Synchronisation des ersten Datenbytes mit dem Signal Cell_Sync_In wird ohne Probleme erreicht.

Beim ersten Versuch übernimmt der Payload-Prozeß die Eingangsdaten einen Taktzyklus zu spät. Das kann dadurch behoben werden, daß im Prozeß AHTIN zwischen das Signal Start_Payl und der Warteschleife auf das Signal Hd_fetch ein Kontrollzyklus eingeschoben wird.

Die Abbildung 4 zeigt die FSM des AHTIN-Prozesses, die Abbildung 5 zeigt die FSM des Payload-Prozesses.

Im Payload-Prozeß wird eine Schleife für das Sammeln der vier Bytes verwendet, die am Stück zum Cell_FIFO übertragen werden. Die Schleife wird 12 mal durchlaufen. Die Simulation Bild 6 zeigt, daß die Daten kontinuierlich übertragen werden. Die Schleife wird bei der Synthese aufgerollt.

4.1.3 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen ("elaborate"): ca. 14 sec.

2. Allokierungs-Schritt (mit Option"fastest"): ca. 3 sec.

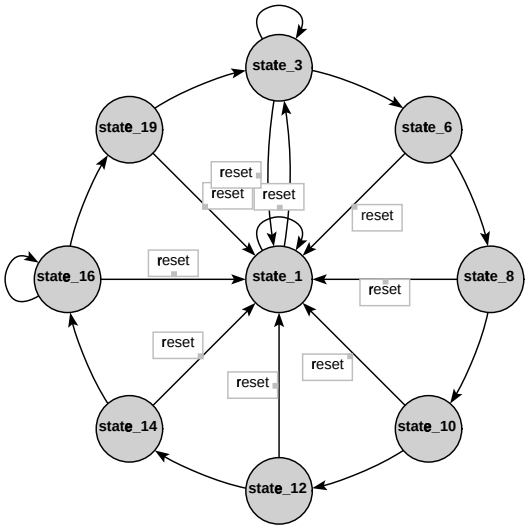


Abbildung 4: FSM des Prozesses AHTIN

3. Scheduling-Schritt: ca. 3 sec.
4. Schritt: “Extract Datapath and FSM’s”: ca. 5 sec
5. Schritt: “Bind to Components”: ca. 5 sec
6. Schritt “Share resources”: ca. 28 sec

Aus den Synthese-Reports (siehe Anhang) ergeben sich folgende Zahlen:

Zuverlässigkeit:
Die Synthese wurde **fehlerlos** abgeschlossen.
Das Ergebnis der Synthese ist **akzeptabel** und **korrekt simulierbar**.
Zuverlässigkeitsmaß: 10.

Daten der synthetisierten Schaltkreise: (RT-Ebene):

Gesamte Schaltungsfläche (DP After Sharing): 1842.8

Geschätzte Verzögerung (Kritischer Pfad): 17.96 ns

1. **Prozeß AHTIN**
   (a) **Anzahl Kontrollzyklen: (Schedule Length)** 10. (Bei einer Taktperiode von 25 ns, simuliert mit Takt 25 ns).
   (b) **Anzahl der Operationen** nach Monet-Definition: “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

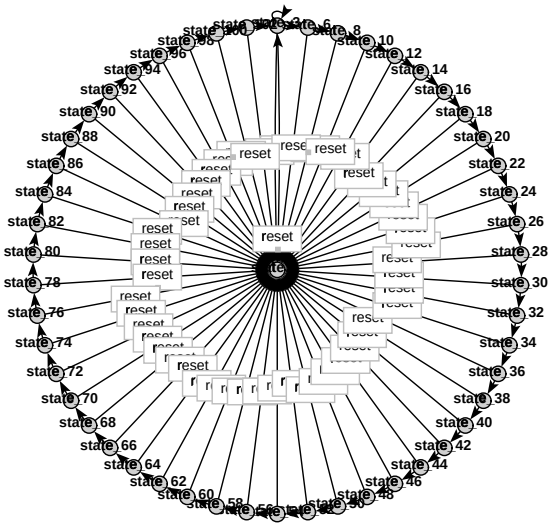


Abbildung 5: FSM des Prozesses AHTIN

- Reale Operationen: 0.
- Pseudo-Operationen: 32. (Fehler im Report)

(c) **Die FSM:** (siehe Bild 4).

- Anzahl der Zustände: 9
- Anzahl der Zustandsübergänge: (transitions) 17.
- Gesamte Fläche: 905.
- Max. Verzögerung (Kritischer Pfad): 0.35 ns.

2. **Prozeß Payload**

(a) **Anzahl Kontrollzyklen:** 52. (Bei einer Taktperiode von 25 ns, simuliert mit Takt 25 ns).

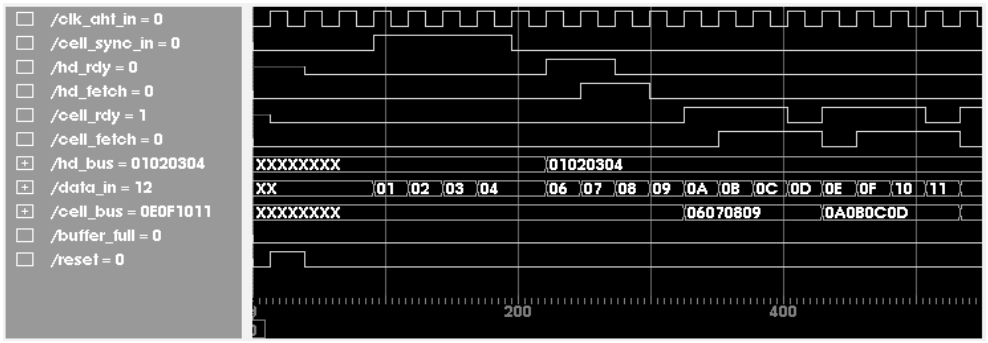
(b) **Schaltungsfläche:** (siehe oben).

(c) **Anzahl der Operationen** nach Monet-Definition: 238. “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

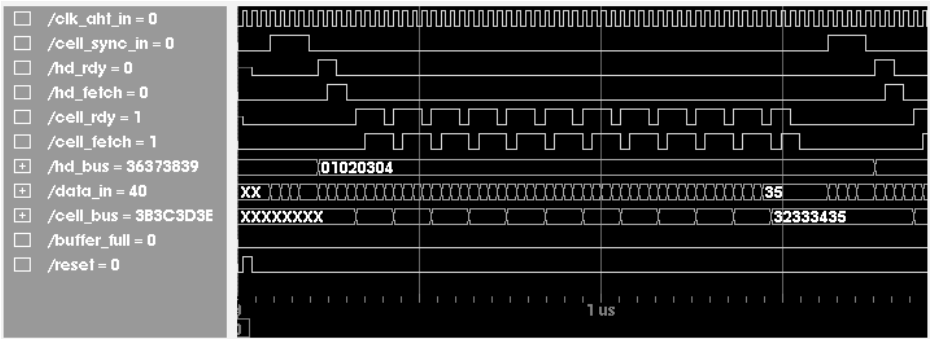
- Reale Operationen: 0.
- Pseudo-Operationen: 238.

(d) **Die FSM:** (siehe Abbildung 5)

- Anzahl der Zustände: 51
- Anzahl der Zustandsübergänge: (transitions) 102.



Simulation des Moduls AHTIN nach der Synthese, die ersten Takte.



Simulation des Moduls AHTIN, Übertragung einer ganzen Zelle.

Abbildung 6: Simulation des Moduls AHT_In nach der Synthese

- Gesamte Fläche: 15809
- Max. Verzögerung: 0.35 ns.

5 Das Zellkopfübersetzungsmodul HT

Das Zellkopfübersetzungsmodul (HT) erhält den Zellkopf vom Modul AHT_IN und adressiert mit den Feldern VPI/VCI die Routing-Tabelle. Die Routing-Tabelle gibt die neuen VPI/VCI-Werte zusätzlich mit einer Routing-Information zurück. Die neuen VPI/VCI-Werte werden in den Zellkopf eingesetzt und in den Header-FIFO abgespeichert.

5.1 Synthese mit Monet

Der VHDL-Quellcode der Spezifikation des Zellkopfübersetzungsmoduls (HT) wird entsprechend den Monet-Richtlinien geändert. Die Änderungen betreffen hauptsächlich das synchrone Warten auf Signale und das synchrone Zurücksetzen (Reset).

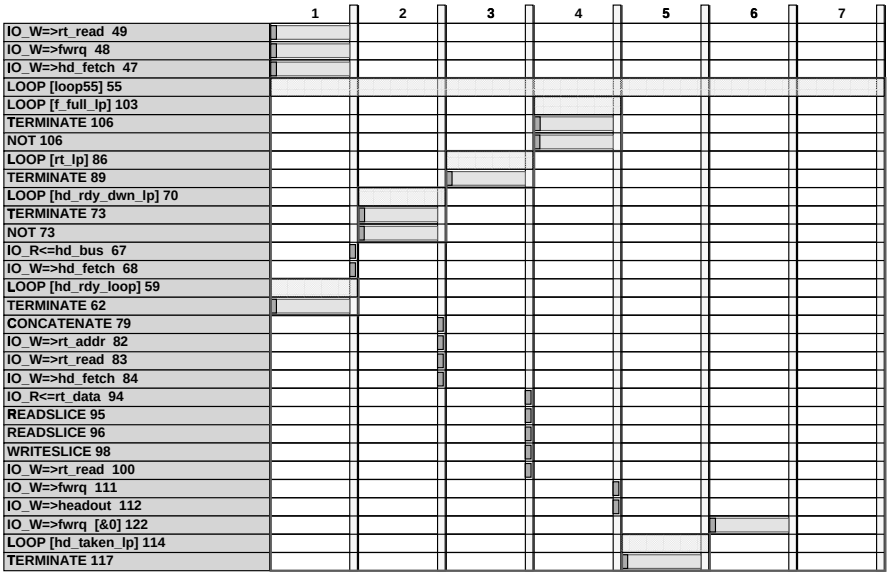


Abbildung 7: Scheduling-Ergebnisse des ht mit Monet: Gantt-chart. Die Spalten zeigen die Taktzyklen.

5.1.1 Gantt-Chart und FSM

Das Zellkopfübersetzungsmodul hat im wesentlichen nur Eingabe- und Ausgabefunktionen, verbunden mit Bitfelder-Verschiebungen. Monet scheduled die VHDL-Beschreibung in 7 Taktzyklen.

Die Ergebnisse des Scheduling zeigt die sog. “Gantt-chart” (Bild 7).

In den Spalten sind die Taktzyklen dargestellt, die Zeilen zeigen die Operationen an. Das synchrone Warten auf eine Signal gelingt Monet in einem Taktzyklus (ein Zustand der FSM).

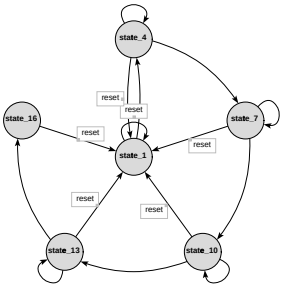


Abbildung 8: Die FSM des Prozesses htproc

Die Abbildung 8 zeigt die FSM für den Prozeß “htproc”. Die FSM ist zyklisch dargestellt, die synchronen Resets bewirken einen Übergang von jedem Zustand zum Zustand 1.

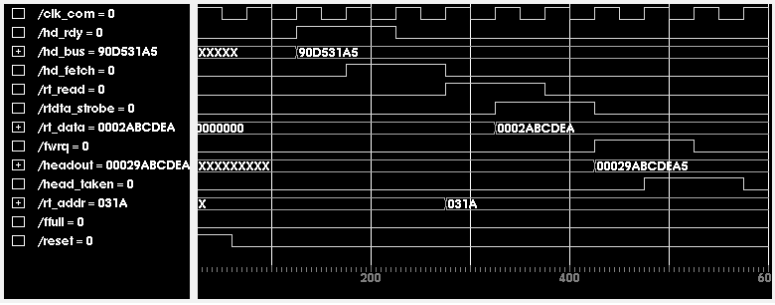


Abbildung 9: Simulation des ht nach dem letzten Schritt der Synthese

Schließlich zeigt die Abbildung 9 die Simulation der Netzliste nach dem letzten Schritt der Synthese mit Monet: (“Share resources”).

5.1.2 Synthese-Daten für den Prozeß “htproc”

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen (“elaborate”): ca. 4 sec.
2. Allokierungs-Schritt (mit Option“fastest”): ca. 3 sec.
3. Scheduling-Schritt: ca. 3 sec.
4. Schritt: “Extract Datapath and FSM’s”: ca. 3 sec
5. Schritt: “Bind to Components”: ca. 3 sec

6. Schritt “Share resources”: ca. 3 sec

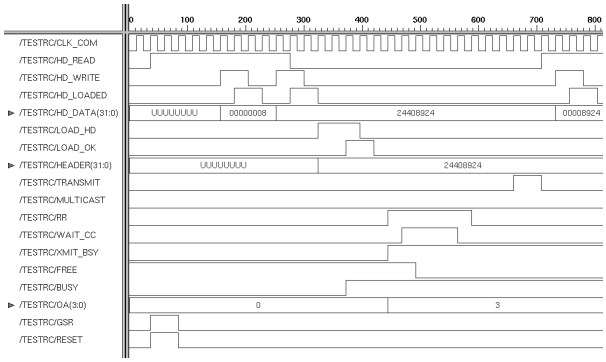
Zuverlässigkeit:
Die Synthese wurde **fehlerlos** abgeschlossen.
Das Ergebnis der Synthese ist **akzeptabel** und **korrekt simulierbar**.
Zuverlässigkeitsmaß: 10.

Aus dem Synthese-Report (siehe Anhang) ergeben sich folgende Zahlen:

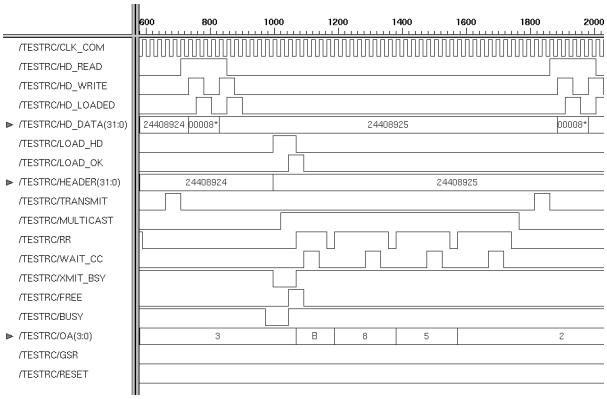
1. **Gesamte Schaltungsfläche (DP After Sharing):** 1242.610
2. **Geschätzte Verzögerung (Kritischer Pfad):** 2.33 ns
3. **Anzahl Kontrollzyklen:** 7 (Bei einer Taktperiode von 25 ns, simuliert mit Takt 50 ns).
4. **Anzahl der Operationen** : “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)
    - Reale Operationen: 2.
    - Pseudo-Operationen: 44.
5. **Daten der FSM** (siehe Bild 8). :
    - Anzahl der Zustände: 7
    - Anzahl der Zustandsübergänge: (transitions) 12,
    - Gesamte Fläche: 430.
    - Max. Verzögerung: 0.44 ns.

6 Die Routingsteuerung RC

Die Routingsteuerung (Routing Control RC) nimmt den Zellkopf aus dem FIFO-Speicher, trennt die Routing-Information ab, interpretiert sie und gibt entsprechende Verbindungsanfragen (RR) an die Verbindungs-Steuerung (CC) weiter.



Funktionale Simulation der Routing-Steuerung (Singlecast)



Funktionale Simulation der Routing-Steuerung (Multicast)

Abbildung 10: Simulation des RC-Moduls. Die oberen Skalen zeigen die Zeit in ns

Die Routing-Information (der Routing-Tag) enthält die Information, ob eine Einzelverbindung (Singlecast) oder eine Mehrfachverbindung (Multicast) gefordert ist.

Die Verhaltensbeschreibung der VHDL-Architektur enthält zwei Prozesse: RCIN, der das Interface zum FIFO-Speicher und zur Verbindungssteuerung bedient und RCOUT, der die Verbindung zur Schieberegister-Steuerung unterhält.

Der Prozeß RCIN ist wesentlich komplizierter als das Zellkopf-Übersetzungs-Modul. Es enthält eine Schleife, die nicht aufgerollt werden darf, sowie mehrfach verschachtelte IF-Strukturen.

Folgende Änderungen des VHDL-Quellcodes waren nötig, um das Modul mit dem Synopsys BC zu synthetisieren:

Während bei der funktionalen Simulation eine synchrone Übertragung des Zellkopfs

plus Routing-Information aus dem Zellkopf-Speicher angenommen wurde, kann dies für eine Synthese mit Monet nicht mehr aufrecht erhalten werden, da durch zusätzliche Verzögerungen Unsicherheiten in der Datenübertragung auftreten. In der Verhaltensbeschreibung wird die Übernahme des Zellkopf mittels eines Two Way Handshaking-Protokolls eingeführt.

Die Simulation der geänderten Verhaltensbeschreibung zeigt BNild 10.

6.1 Synthese mit Monet

Im Quellcode der Routingsteuerung werden folgende Änderungen durchgeführt:

1. Eine “Rest-Loop” mit einem Reset-Signal wird eingeführt.
2. Die Schleife mit der Abfrage nach den Ausgangsbits im 16 Bit breiten Routing Tag darf nicht aufgerollt werden. (Attribut dont_unroll). Die verwendete Version (R_33) von Monet bleibt sonst im Allocation-Step hängen. Dies ist auch allgemein sinnvoll, da das Setzen der Ausgangsadressen für die Verbindungssteuerung nur seriell erfolgen kann.
3. Die Konvertierungen der Laufvariable in der Schleife zur Ausgangsadresse (Typ Std_Logic_Vector, 4 Bit breit) wird mit Standardfunktionen aus der Bibliothek Std_Logic_arith durchgeführt.

Die Ergebnisse des Scheduling zeigen die “Gantt-charts” Abbildung 11 für den RC-Eingangs-Prozeß rcin und Abbildung 12 für den RC-Ausgangs-Prozeß rcout.

Die Abbildung 13 zeigt die FSM für den Prozeß “rcin” und die Abbildung 14 zeigt die FSM für den Prozeß rcout.

Die Abbildung 15 zeigt die Simulation der Netzliste nach dem letzten Schritt der Synthese mit Monet: (Share resources).

6.1.1 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen (“elaborate”): ca. 12 sec.
2. Allokierungs-Schritt (mit Option“fastest”): ca. 3 sec.
3. Scheduling-Schritt: ca. 3 sec.
4. Schritt: “Extract Datapath and FSM’s”: ca. 3 sec
5. Schritt: “Bind to Components”: ca. 3 sec
6. Schritt “Share resources”: ca. 8 sec

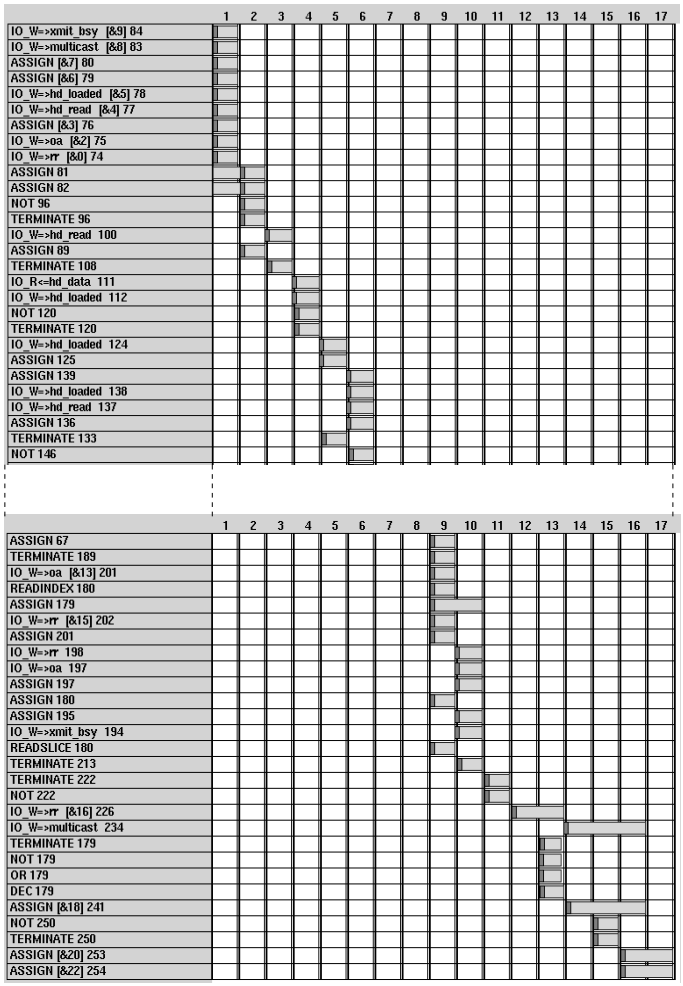


Abbildung 11: Scheduling-Ergebnisse des Prozesses rcin der Routingsteuerung mit Monet: Gantt Chart. Nur der obere und der untere Teil der Gantt Chart sind dargestellt.

Zuverlässigkeit:

Die Synthese bleibt im “Allocation”-Schritt hängen. Durch Ändern der Verhaltensbeschreibung (Die Loop L1 darf nicht aufgerollt werden) wird eine Umgehung des Fehlers erreicht. Dadurch wird das Ergebnis der Synthese **akzeptabel** und ist **korrekt** simulierbar.

Zuverlässigkeitsmaß: 8.

Aus dem Synthese-Report (siehe Anhang) ergeben sich folgende Zahlen:

Daten der synthetisierten Schaltkreise: (RT-Ebene):

Gesamte Schaltungsfläche (DP After Sharing): 1688.9

Geschätzte Verzögerung (Kritischer Pfad): 6.94 ns

1. **Prozeß rcin:**

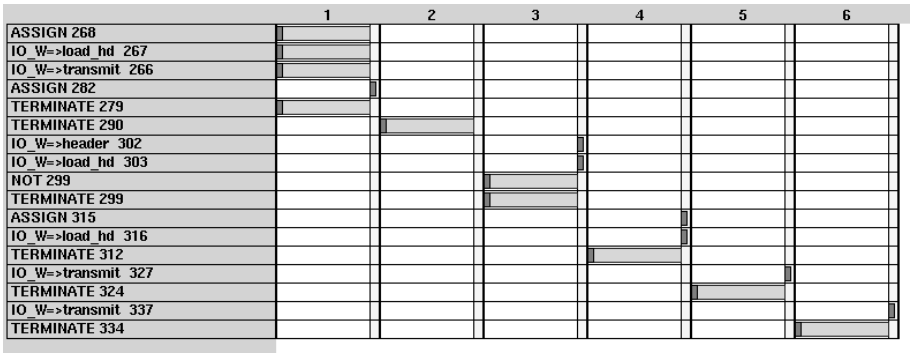


Abbildung 12: Scheduling-Ergebnisse des Prozesses rcout der Routingsteuerung mit Monet: Gantt Chart. Es sind nicht alle Operationen dargestellt.

(a) **Anzahl Kontrollzyklen:** 17.

(b) **Anzahl der Operationen** nach Monet-Definition: 172 “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

- Reale Operationen: 13.
- Pseudo-Operationen: 159.

(c) **Daten der FSM** :

- Anzahl der Zustände: 18
- Anzahl der Zustandsübergänge: (transitions) 33.
- Gesamte Fläche: 3590
- Max. Verzögerung: 1.09 ns.

2. **Prozeß “rcout”**

(a) **Anzahl Kontrollzyklen:** 6.

(b) **Anzahl der Operationen** nach Monet-Definition: 39

- Reale Operationen: 1.
- Pseudo-Operationen: 38.

(c) **Daten der FSM** :

- Anzahl der Zustände: 7
- Anzahl der Zustandsübergänge: (transitions) 12.
- Gesamte Fläche: 517
- Max Verzögerung: 0.44 ns.

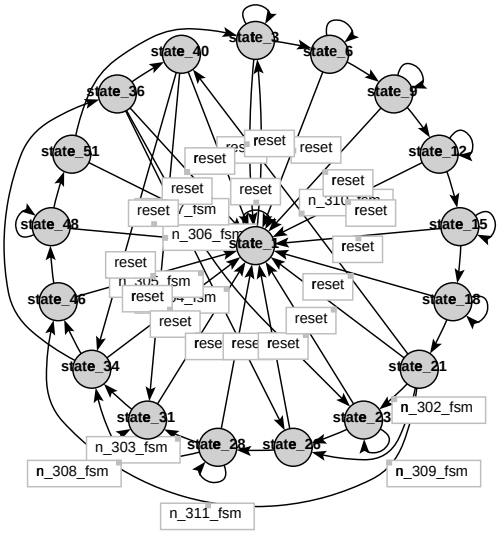


Abbildung 13: FSM des Prozesses rcin.

7 Die Schieberegister-Steuerung SRC

Die Schieberegister-Steuerung übernimmt den Zellkopf von der Routing-Steuerung und die Nutzlast direkt aus dem “Payload-FIFO”. Um eine sichere Übertragung zu gewährleisten, werden sowohl Zellkopf als auch Nutzlast in Einheiten zu 32 Bit mit dem 4B/5B Verfahren verschlüsselt. Jeder Einheit wird ein Startzeichen vorangestellt, wobei der Zellkopf ein eigenes Startzeichen erhält, damit der Zellstart eindeutig erkennbar ist.

Die Verschlüsselungsfunktion wird im Package “utilc_src” beschrieben. Hier ist es angebracht, das Synopsys-Attribut “preserve_function” zu wählen. Dadurch wird die Verschlüsselungsfunktion vom Scheduler wie eine Komponente behandelt. Es wird vermieden, daß bei jeder Verschlüsselungs-Aktion (je zwei pro Byte) der Verschlüsselungs-Quellcode inline in die VHDL-Beschreibung hineinkopiert wird.

Die Einheiten werden in ein 48 Bit breites Schieberegister geladen und seriell durch den Switch geschoben, wenn die Verbindung geschaltet ist.

Abweichend von der ersten Verhaltensbeschreibung wird das Schieberegister als externe Komponente angenommen.

Letzteres ist auch insofern sinnvoll, als das Schieberegister auf das Spannungsniveau des ATM-Switches, der in PECL-Technologie vorliegt, angehoben werden muß. Für das Schieberegister lohnt es sich auch, eine Technologie zu wählen, die eine sehr hohe Taktfrequenz erlaubt, um die Datenübertragung zu beschleunigen. Die Schnittstellen des SRC mit externem Schieberegister zeigt Bild 16.

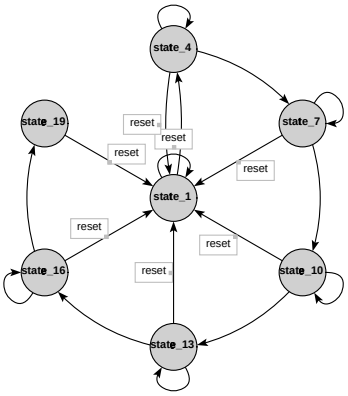


Abbildung 14: FSM des Prozesses rcout

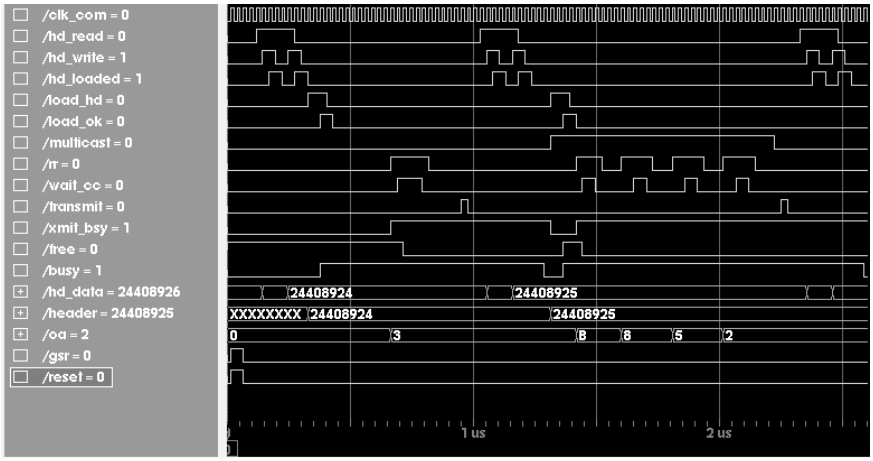


Abbildung 15: Simulation der Routingsteuerung nach der Synthese.

7.1 Synthese mit Monet

Der Quellcode der Schieberegistersteuerung (SRC) wird entsprechend den Monet-Richtlinien geändert. Die Änderungen betreffen hauptsächlich das synchrone Warten auf Signale und das synchrone Zurücksetzen (Reset).

7.1.1 Schleifen-Behandlung

Die Schleife (L1), die jeweils 4 Bytes der Nutzlast (Payload) erhält und verschlüsselt, wird nicht aufgerollt. Die Schleife L1 wird zwölfmal durchlaufen. Aus funktionalen Gründen ist dies sinnvoll, zusätzlich werden dadurch die Zustände der FSM reduziert.

Die geschätzte Schaltungsfläche ist relativ groß. Das kommt daher, daß Monet für die Datenverschlüsselung in SRC die Funktion “encd()” nicht automatisch wie eine Komponente behandelt (wie dies beim BC durch das Attribut “preserve_function” möglich

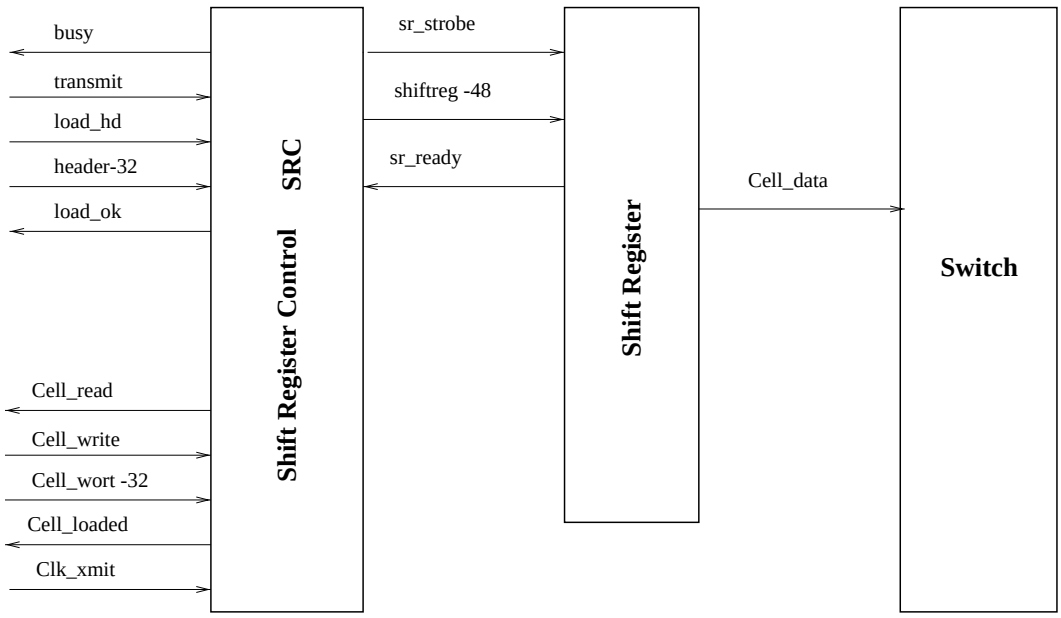


Abbildung 16: Schnittstelle des SRC-Moduls mit externem Schieberegister

ist), sondern jedesmal beim Aufruf die Funktion encd() inline setzt und damit die Fläche erhöht.

Eine separate Komponente zu definieren, ist aufwendig. Es wird in diesem Rahmen darauf verzichtet.

7.1.2 Gantt-Charts und FSM's:

Die Abbildung 17 zeigt den oberen Teil der Gantt-Chart des Prozesses Shift_Proc.

Die Abbildung 18 zeigt die FSM des Shift_Proc-Prozesses,

Die Abbildungen 19 und 20 zeigen die Simulation der Schieberegister-Steuerung nach der Synthese.

7.1.3 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen ("elaborate"): ca. 16 sec.
2. Allokierungs-Schritt (mit Option"fastest"): ca. 5 sec.
3. Scheduling-Schritt: ca. 5 sec.
4. Schritt: "Extract Datapath and FSM's": ca. 5 sec
5. Schritt: "Bind to Components": ca. 5 sec

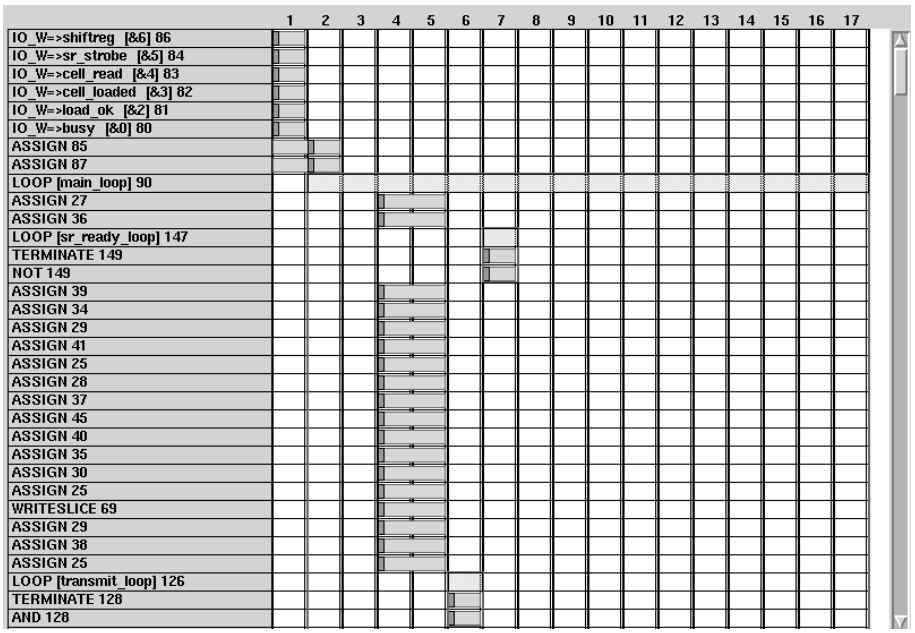


Abbildung 17: Oberer Teil der Gantt-Chart des Prozesses SRC.

6. Schritt “Share resources”: ca. 7 sec

Zuverlässigkeit:
Die Synthese wurde **fehlerlos** abgeschlossen.
Das Ergebnis der Synthese ist **akzeptabel** und **korrekt simulierbar**.
Zuverlässigkeitsmaß: 10.

Aus dem Synthese-Report (siehe Anhang) ergeben sich folgende Zahlen:

Daten der synthetisierten Schaltkreise: (RT-Ebene):

Gesamte Schaltungsfläche (DP After Sharing): 7120.5

Geschätzte Verzögerung (Kritischer Pfad): 7.04 ns

Prozeß: Shift_Proc

1. **Anzahl Kontrollzyklen: (Schedule Length)** 17. (Bei einer Taktperiode von 25 ns, simuliert mit Takt 25 ns).

2. **Anzahl der Operationen** nach Monet-Definition: “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

   - Reale Operationen: 7
   - Pseudo-Operationen: 703

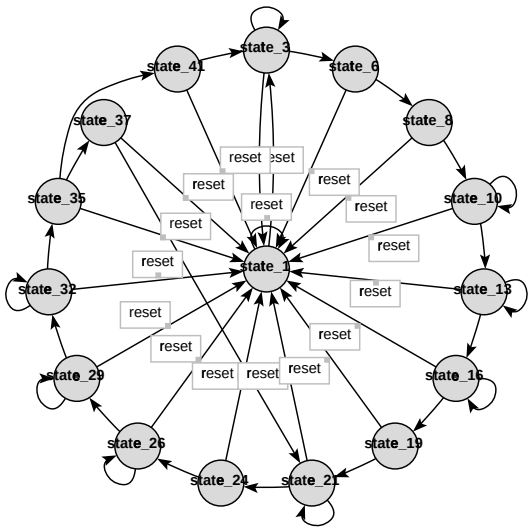


Abbildung 18: FSM des Prozesses SRC

3. **Die FSM:** (siehe Bild 18).
    - Anzahl der Zustände: 16
    - Anzahl der Zustandsübergänge: (transitions) 32.
    - Gesamte Fläche: 2093.
    - Max. Verzögerung: 0.61 ns.

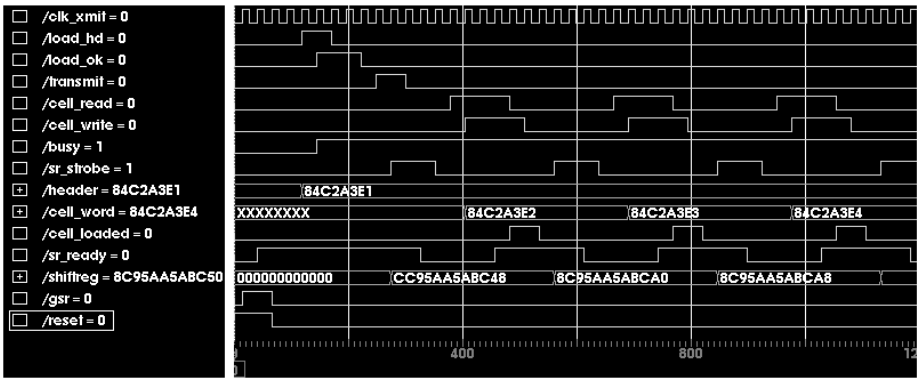


Abbildung 19: Simulation nach der Synthese: Die ersten Takte.

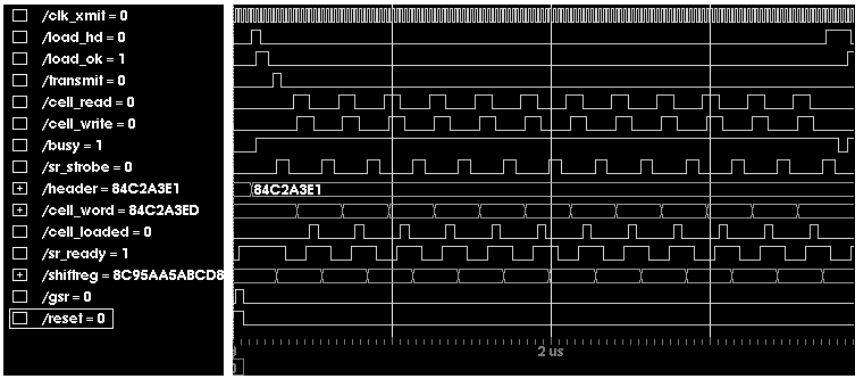


Abbildung 20: Simulation nach der Synthese: Eine Zelle wird übertragen

8 Die Verbindungs-Steuerung CC

Die Verbindungs-Steuerung (Connection Control CC) nimmt die Verbindungsanfragen (Routing Requests RR) von 14 Routing-Steuerungen (RC) an und bearbeitet sie der Reihe nach. Eine Verbindungsanfrage kann ‘Singlecast’ sein, d.h. die Verbindung zu nur einem Ausgang wird angefordert, sie kann ‘Multicast’ sein, (Verbindungen zu mehreren Ausgängen) oder sie kann ‘Broadcast’ sein (Verbindung von Kanal 0 zu allen Ausgängen).

8.1 Synthese mit Monet

Verbindungs-Steuerung (Connection Control CC)

Der Quellcode der Verbindungs-Steuerung (Connection Control CC) wird entsprechend den Monet-Richtlinien geändert. Die Änderungen betreffen hauptsächlich das synchrone Warten auf Signale und das synchrone Zurücksetzen (Reset).

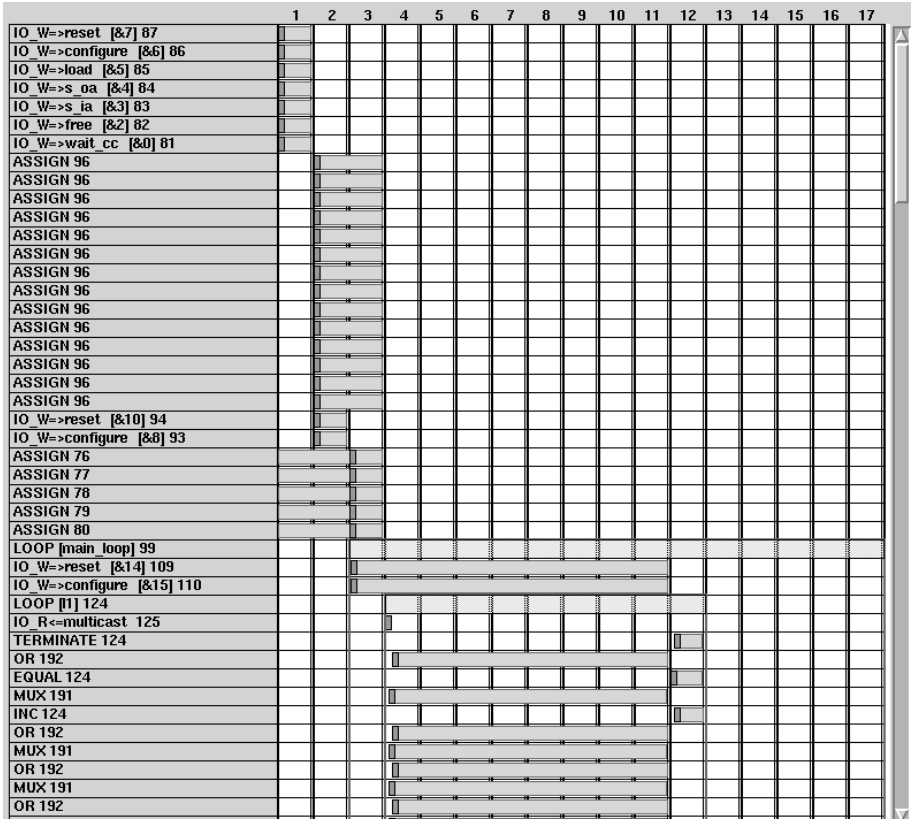


Abbildung 21: Oberer Teil der Gantt-Chart des Prozesses ccin.

8.1.1 Schleifen-Behandlung

Die Schleifen werden nicht aufgerollt. Dies reduziert die Stati der FSM erheblich und ist aus funktionalen Gründen sinnvoll, da der Switch nur sequentiell konfiguriert werden kann.

8.1.2 Gantt-Chart

Die Abbildung 21 zeigt den oberen Teil der Gantt-Chart des Prozesses ccin nach dem Scheduling.

Im nächsten Schritt (Extract Datapath and FSM's) bricht Monet mit einem Bus-Fehler (Bus error) ab.

8.1.3 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen ("elaborate"): ca. 16 sec.

2. Allokierungs-Schritt (mit Option“fastest”): ca. 3 sec.

3. Scheduling-Schritt: ca. 5 sec.

Zuverlässigkeit:
Die Synthese bricht mit einem Programmfehler (s.o) ab. Es wird nur ein Teilergebnis erreicht, das zwar Aufschlüsse über die zu erwartende Schaltung ergibt, aber das Ergebnis der Synthese kann nicht weiter synthetisiert werden.
Zuverlässigkeitsmaß: 2.

Aus dem Scheduling-Report (siehe Anhang) ergeben sich folgende Zahlen:

Schaltungsfläche (nach dem Scheduling): 666.2

Gesamte Verzögerung: Nicht angegeben, da Abbruch nach dem Scheduling.

Prozess ccin

1. **Anzahl Kontrollzyklen (Schedule Length):** 17.

   (Bei einer Taktperiode von 25 ns).

2. **Anzahl der Operationen** nach Monet-Definition: 352

   “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

   - Reale Operationen: 88
   - Pseudo-Operationen: 264

3. **Die FSM:**

   Nicht angegeben, da Abbruch nach dem Scheduling.

9 Das Datenannahme-Modul RD

Für jeden Kanal existiert ein “Datenannahme”-Modul (RD) am Ausgang des ATM-Switches der die seriellen Bitströme empfängt. Die Daten werden in Einheiten zu je 48 Bit übertragen. Jede 48-Bit Einheit ist im 4B/5B Verfahren codiert und trägt zu Beginn ein Startzeichen, wobei zwischen Zellkopf- und Nutzlast- Startzeichen unterschieden wird.

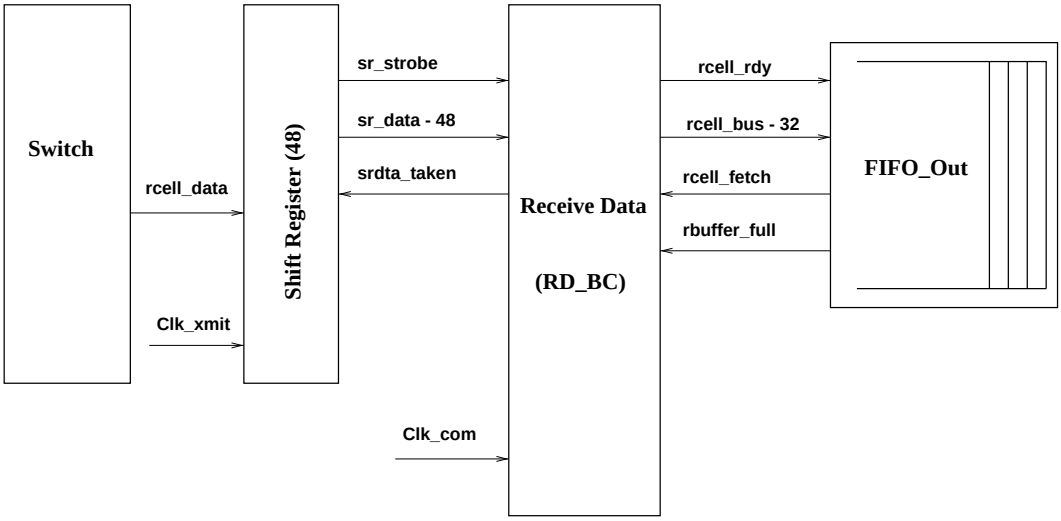


Abbildung 22: Schnittstellen des RD-Moduls nach Separierung des Schieberegisters

Analog zur Schieberegister-Steuerung SRC wird auch hier das Schieberegister für die Synthese mit Monet entfernt. Die sich daraus ergebenden Schnittstellen sind in Abbildung 22 ersichtlich.

9.1 Synthese mit Monet

Nach den Monet-spezifischen Änderungen im Quellcode des Daten-Annahme-Moduls (RD) (z.B. Reset-Loop, Schleifen anstatt kombinierte Wait-statements, etc.) erhalten wir die folgenden Synthese-Ergebnisse.

Die Ergebnisse des Scheduling zeigen die “Gantt-charts” Abbildung 23 für den RD-EingangsProzeß rdin und Abbildung 24 für den RD-AusgangsProzeß rdout.

Die Abbildung 25 zeigt die FSM für den Prozeß “rdin” und die Abbildung 26 zeigt die FSM für den Prozeß rdout.

Die Abbildungen 27 und 28 zeigen die Simulation der Netzliste nach dem letzten Schritt der Synthese mit Monet: (Share resources). Abbildung 27 zeigt die Annahme der ersten Zellworte, Abbildung 28 zeigt die Annahme einer ganzen Zelle.

Die geschätzte Schaltungsfläche ist relativ groß. Das kommt daher, daß Monet für die Datenentschlüsselung in RD die Prozedur “decode()” nicht automatisch wie eine Komponente behandelt (wie dies beim BC durch das Attribut “preserve_function” theoretisch möglich wäre), sondern jedesmal beim Aufruf die Prozedur inline setzt und damit die Fläche erhöht.

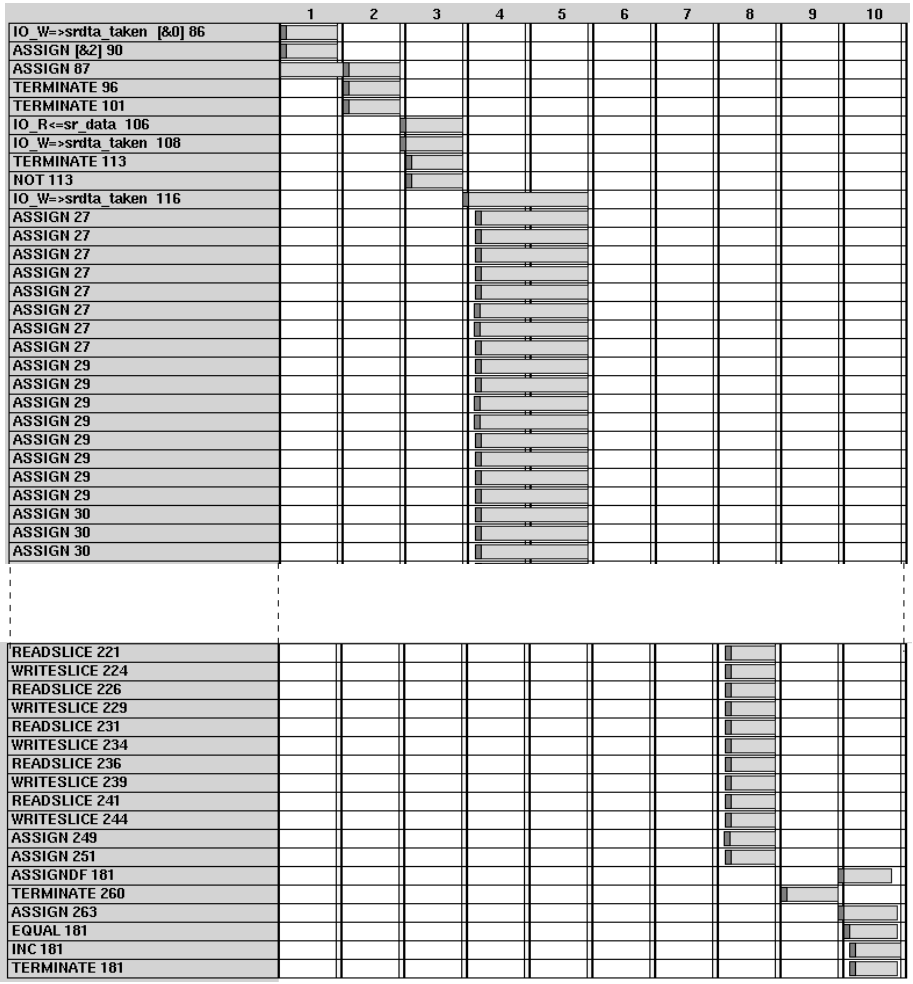


Abbildung 23: Scheduling-Ergebnisse des Prozesses rdin des Daten-Annahme-Moduls mit Monet: Gantt Chart. Nur der obere und der untere Teil der Gantt Chart sind dargestellt.

Eine separate Komponente anstelle der Prozedur zu definieren ist aufwendig. Es wird in diesem Rahmen darauf verzichtet.

9.1.1 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen (“elaborate”): ca. 16 sec.
2. Allokierungs-Schritt (mit Option“fastest”): ca. 5 sec.
3. Scheduling-Schritt: ca. 8 sec.
4. Schritt: “Extract Datapath and FSM’s”: ca. 5 sec

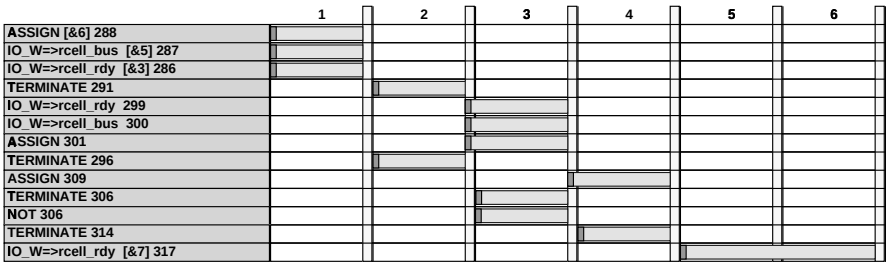


Abbildung 24: Scheduling-Ergebnisse des Prozesses rdout des Daten-Annahme-Moduls mit Monet: Gantt Chart.

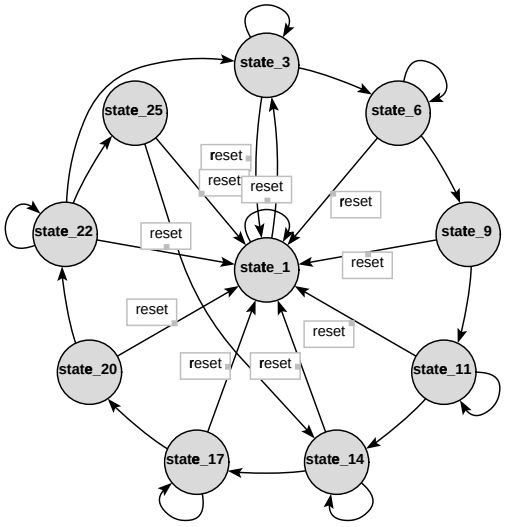


Abbildung 25: FSM des Prozesses rdin.

5. Schritt: “Bind to Components”: ca. 6 sec
6. Schritt “Share resources”: ca. 12 sec

Zuverlässigkeit:
Die Synthese wurde **fehlerlos** abgeschlossen.
Das Ergebnis der Synthese ist **akzeptabel** und **korrekt simulierbar**.
Zuverlässigkeitsmaß: 10.

Aus dem Synthese-Report (siehe Anhang) ergeben sich folgende Zahlen:

Daten der synthetisierten Schaltkreise: (RT-Ebene):

Gesamte Schaltungsfläche (DP After Sharing): 14380.7

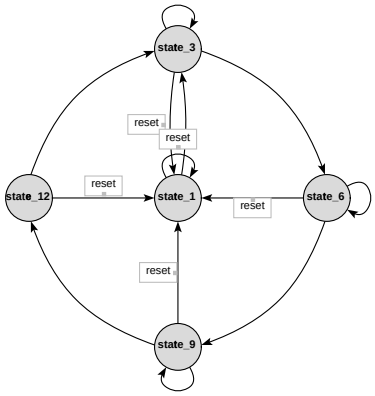


Abbildung 26: FSM des Prozesses rdout

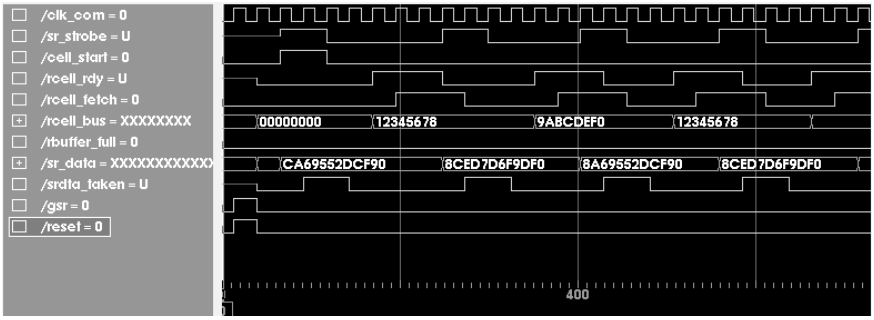


Abbildung 27: Simulation des Daten-Annahme-Moduls nach der Synthese: Annahme der ersten Zellworte.

Der Grund für die hohe Schaltungsfläche ist die parallele Realisierung des 4B/5B Dekoders.

Geschätzte Verzögerung (Kritischer Pfad): 10.1226 ns

1. **Prozeß: rdin:**

   (a) **Anzahl Kontrollzyklen (Schedule Length):** 10. (Bei einer Taktperiode von 25 ns, simuliert mit Takt 25 ns).

   (b) **Anzahl der Operationen** nach Monet-Definition: 701 “Reale Operationen” sind solche Operationen für die eine Komponente in der Bibliothek zur Verfügung steht. (z.B.: alle arithmetische und logische Operationen) Für “Pseudo-Operationen” werden keine Komponenten eingesetzt. (Pseudo-Operationen sind z.B.: Schleifen-Beginn, Schleifen-Ende, Register-Schreiben, Register-Lesen usw.)

   - Reale Operationen: 7.
   - Pseudo-Operationen: 694.

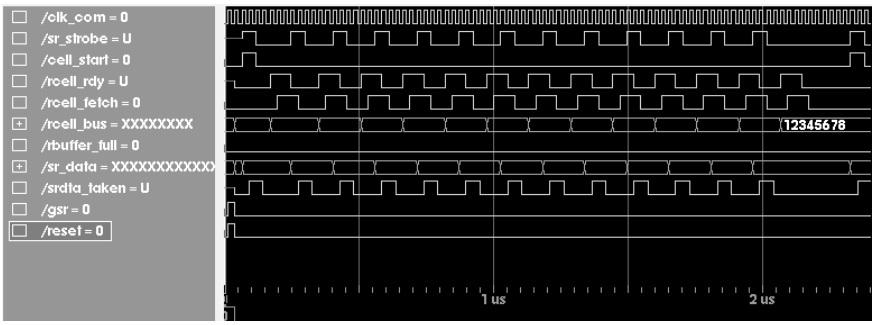


Abbildung 28: Simulation des Daten-Annahme-Moduls nach der Synthese. Annahme einer ganzen Zelle

(c) **Daten der FSM** :

- Anzahl der Zustände: 10
- Anzahl der Zustandsübergänge: (transitions) 20.
- Max. Verzögerung: 0.57 ns.
- Gesamte Fläche”1042.

2. **Prozeß: “rdout”**

(a) **Anzahl Kontrollzyklen:** 6.

(b) **Anzahl der Operationen** nach Monet-Definition:

- Reale Operationen: 1.
- Pseudo-Operationen: 28.

(c) **Daten der FSM** :

- Anzahl der Zustände: 5
- Anzahl der Zustandsübergänge: (transitions) 9, davon 5 für Resets.
- Max Verzögerung: 0.44 ns.
- Gesamte Fläche: 357

10 Das Ausgangsmodul AHT_OUT

Das AHT-Ausgangsmodule AHT_OUT nimmt Zellen aus dem FIFO_OUT-Speicher und gibt sie auf den 8-Bit breiten Ausgangskanal. Einen Takt vor jeder Zellübertragung wird ein Cell_Presync-Signal aktiv. Mit Zellbeginn wird das Signal Cell_Sync_out aktiv gesetzt. Wichtig ist, daß die Zelldaten kontinuierlich synchron mit dem Takt Clk_aht_out auf die Leitung Data_out gegeben werden.

Das Modul AHT_OUT umfaßt zwei Prozesse: AHTOUT, der die Zelldaten in einer Breite von 32-Bit aus dem FIFO_OUT-Speicher holt und AHTOUT_Transmit, der die Daten in einer Breite von 8 Bit auf die Ausgangsleitung setzt.

10.1 Synthese mit Monet

Ähnlich wie beim Eingangsmodul AHT_IN muß auch beim Ausgangsmodul AHT_OUT darauf geachtet werden, daß

- das erste Datenbyte synchron mit dem Signal Cell_Sync_out ausgegeben wird.
- die Daten kontinuierlich auf die Ausgangsleitung gesetzt werden, d.h. es dürfen keine Datenlücken auftreten und keine Daten verloren gehen.

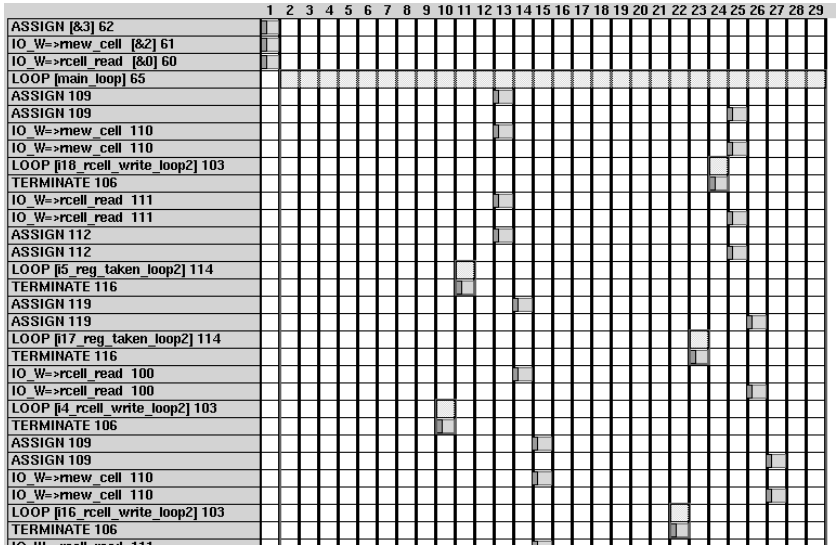


Abbildung 29: Oberer Teil der Gantt-Chart des Prozesses AHTOUT. Die Spalten 1 bis 29 stellen die Kontrollzyklen des Prozesses dar.

Der Quellcode wird entsprechend den Monet-Regeln geändert.

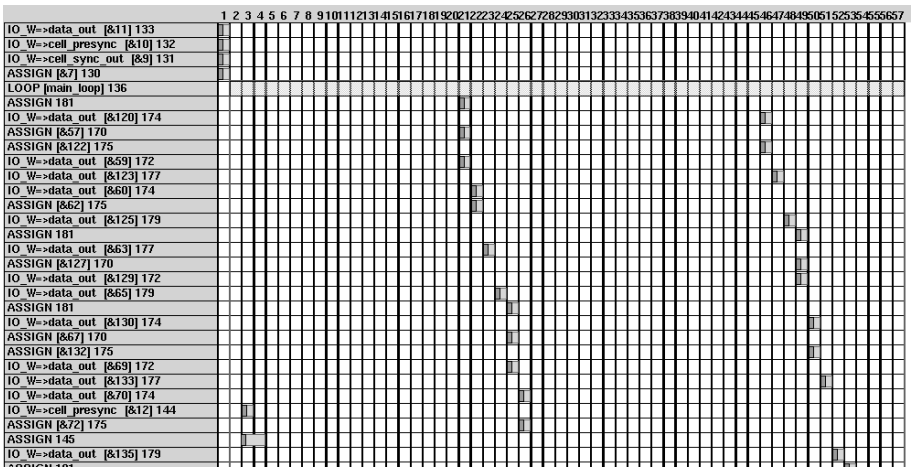


Abbildung 30: Oberer Teil der Gantt-Chart des Prozesses AHTOUT_Transmit.

10.1.1 Behandlung der Schleifen L1 und L2.

Im “AHTOUT”-Prozeß werden in einer Schleife (L1) jeweils 4 Bytes vom FIFO_OUT Speicher geholt und an den “AHTOUT_Transmit”-Prozeß weitergegeben. Die Schleife L1 wird zwölfmal durchlaufen.

Im “AHTOUT_Transmit”-Prozeß werden in einer Schleife (L2) jeweils 4 Bytes auf die Datenleitung (Data_out) zur physikalischen Schicht gegeben. Die Schleife L2 wird zwölfmal durchlaufen.

Ein Aufrollen der Schleifen bedeutet, daß die Steuereinheiten, (die Control-FSM’s) sehr viele Zustände erhalten. Daher wird hier zunächst versucht, die Schleifen **nicht** aufzurollen. Das Ergebnis ist, daß für jede Schleifenausführung ein zusätzlicher Kontrollzyklus aufgewendet wird, der dazu führt, daß eine Datenlücke auf der Ausgangsleitung entsteht. Das Attribut “dont_unroll” wirkt fälschlicherweise für beide Prozesse. Aus diesem Grund wird die Lösung mit aufgerollten Schleifen für beide Prozesse gewählt.

10.1.2 Gantt-Charts und FSM’s:

Die Abbildung 29 zeigt den oberen Teil der Gantt-Chart des Prozesses AHTOUT, die Abbildung 30 zeigt den oberen Teil der Gantt-Chart des AHTOUT_Transmit-Prozesses nach der Synthese.

Die Abbildung 31 zeigt die FSM des ahtout-Prozesses, die Abbildung 32 zeigt die FSM des AHTOUT_Transmit-Prozesses.

Im AHTOUT_Transmit-Prozess wird eine Schleife für das Aufteilen der vier Bytes und das Setzten von einem Byte pro Takt auf die Datenleitung Data_out verwendet. Die Schleife wird 12 mal durchlaufen. Die Simulation Bild 33 zeigt, daß die Daten kontinuierlich übertragen werden. Die Schleifen werden bei der Synthese aufgerollt.

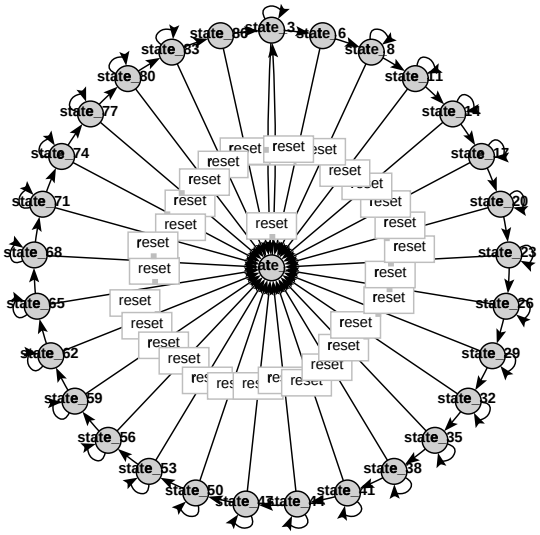


Abbildung 31: FSM des Prozesses AHTOUT

10.1.3 Synthese-Daten

Rechenzeiten: (Sun Ultra 2 mit 640 MB RAM): Es sind die Syntheseschritte aufgeführt, die der Reihe nach auf der graphischen Oberfläche selektiert werden:

1. Einlesen und in eine interen Datendarstellung umformen (“elaborate”): ca. 8 sec.
2. Allokierungs-Schritt (mit Option“fastest”): ca. 5 sec.
3. Scheduling-Schritt: ca. 5 sec.
4. Schritt: “Extract Datapath and FSM’s”: ca. 5 sec
5. Schritt: “Bind to Components”: ca. 3 sec
6. Schritt “Share resources”: ca. 55 sec

Zuverlässigkeit:
Die Synthese wurde **fehlerlos** abgeschlossen.
Das Ergebnis der Synthese ist **akzeptabel** und **korrekt simulierbar**.
Zuverlässigkeitsmaß: 10.

Aus dem Synthese-Report (siehe Anhang) ergeben sich folgende Zahlen:

Daten der synthetisierten Schaltkreise: (RT-Ebene):

Gesamte Schaltungsfläche (DP After Sharing): 1338.03

Geschätzte Verzögerung (Kritischer Pfad): 23.44 ns

1. **Prozess AHTOUT**

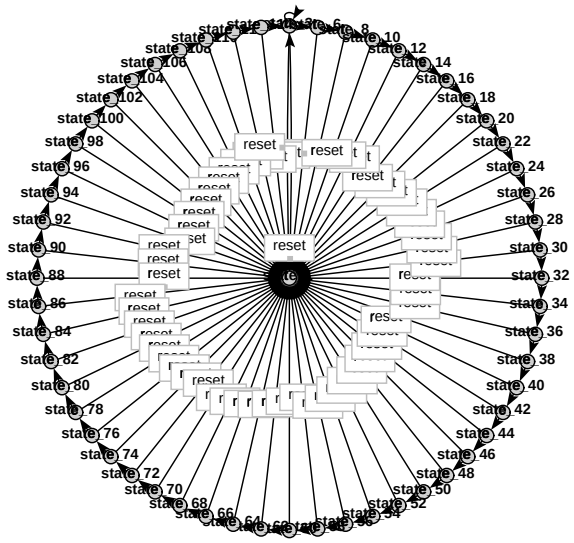


Abbildung 32: FSM des Prozesses AHTOUT_Transmit

(a) **Anzahl Kontrollzyklen: (Schedule Length)** 29. (Bei einer Taktperiode von 25 ns).

(b) **Anzahl der Operationen** nach Monet-Definition: 197.

(c) **Die FSM:** (siehe Bild 31).

- Anzahl der Zustände: 30
- Anzahl der Zustandsübergänge: (transitions) 61.
- Gesamte Fläche: 7676.
- Max. Verzögerung: 0.35 ns.

2. **Prozess AHTOUT_Transmit**

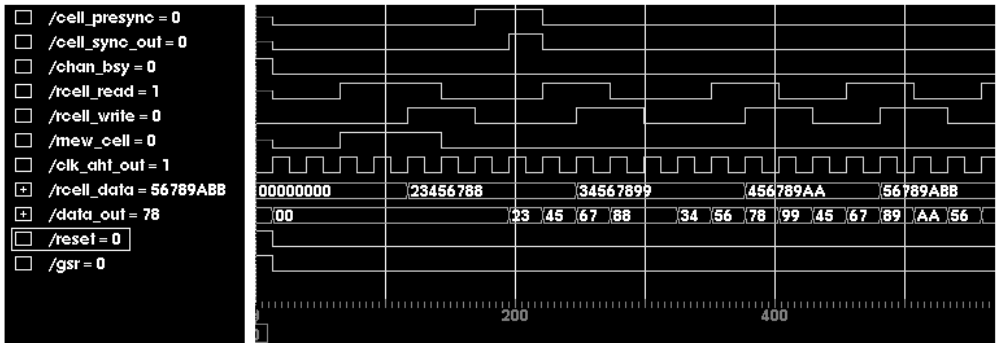
(a) **Anzahl Kontrollzyklen:** 52. (Bei einer Taktperiode von 25 ns, simuliert mit Takt 25 ns).

(b) **Schaltungsfläche:** Keine Angaben.

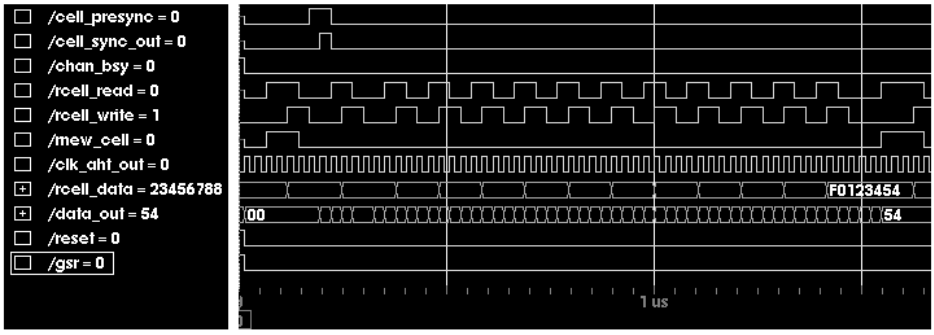
(c) **Anzahl der Operationen** nach Monet-Definition: 162.

(d) **Die FSM:** (siehe Abbildung 32)

- Anzahl der Zustände: 57
- Anzahl der Zustandsübergänge: (transitions) 103.
- Gesamte Fläche: 19266
- Max. Verzögerung: 0.35 ns.



Simulation des Moduls AHT_OUT nach der Synthese: Die ersten Takte



Simulation nach der Synthese: Übertragung einer ganzen Zelle

Abbildung 33: Simulation des Moduls AHT_OUT nach der Synthese

11 Zusammenfassung

In der vorliegenden Arbeit wird eine ATM-Switch-Steuerung (ASS) (siehe [LaRo97]) die in VHDL verhaltensbasiert beschrieben ist, mit Monet von Mentor Graphics synthetisiert. Die Ergebnis-Netzlisten werden simuliert. Von der Beschreibung und der Synthese ausgenommen sind die FIFO-Speicher und die Routing-Tabelle (RT).

Dabei zeigt sich, daß alle in der mit VHDL beschriebenen und simulierten Spezifikationen (nach Einfügen von Änderungen) synthetisiert werden können (Ausnahme: das Modul CC konnte nur bis zum Scheduling-Schritt synthetisiert werden).

Achtet man bei den Verhaltensbeschreibungen darauf, daß nur die synthetisierbare Untermenge der sehr umfangreichen Hardwarebeschreibungssprache VHDL verwendet wird, so halten sich die nötigen Änderungen im Rahmen.

Bei den Eingangs- und Ausgangsmodulen AHT_IN und AHT_OUT ist erreicht worden, daß die Eingangsdaten synchron mit einem Synchronisiersignal aufgenommen und die Ausgangsdaten kontinuierlich weitergeleitet werden.

Vergleicht man die Simulation der Verhaltensbeschreibung eines Moduls mit der Si-

mulation nach der Synthese, so treten bei einigen Modulen Verzögerungen durch die Synthese auf, die bei der RTL- Synthese wahrscheinlich geringer ausfallen würden.

11.1 Verwendete Bibliotheken und Parametereinstellungen

Abbildung 1 zeigt die verwendeten Bibliotheken und Parametereinstellungen.

Bibliotheken	I/O-Modus	Allocation-Modus	Takt-periode	Reset
lca_300k_comp_dc lca_300k_dmag_dc	superstate fixed	fastest	25 ns	synchron

Tabelle 1: Verwendetet Bibliotheken und Parametereinstellungen

Die **Monet-Bibliotheken** sind LCA- (Logic Cell Array) Bibliotheken bei denen der Ein-Bit-Inverter mit der Flächeneinheit 1 die Basis-Zelle darstellt.

Der **I/O-Modus** "superstate fixed" ist in Kapitel 1 beschrieben.

Der **Allocation-Modus** gibt die Ansiedlung der Schaltung im Entwicklungsraum an: In unserem Fall hat bei der Allocation eine Komponente mit geringer Verzögerung Vorrang vor der Komponente mit geringer Fläche.

Bei Monet gibt es im Allocation-Schritt drei Wahlmöglichkeiten: fastest, smallest area, custom. Es wird "fastest" selektiert.

Die **Taktperiode** wird bei allen Modulen mit 25 ns gewählt.

Als **Reset** oder Zurücksetzungsmodus wird das globale synchrone Zurücksetzen gewählt.

11.2 Ergebnisse der Synthese

11.2.1 Fläche, Verzögerung, FSM

Die Tabelle Abbildung 34 zeigt die Ergebnisse der Synthese der einzelnen Module und Prozesse in Bezug auf nc-loc's (non-commentary lines of code), Chipfläche (Area), Verzögerung (delay), Anzahl Kontrollschritte (#csteps), Anzahl Operationen (#ops) und FSM-Daten.

Im ersten Block der Tabelle **"Behavioral VHDL Description"** werden die einzelnen Module und die dazugehörigen Prozesse aufgeführt. Die Spalte **"nc-loc"** (non-commentary lines of code) gibt die effektive Anzahl der VHDL-Quellcode-Zeilen, ohne Kommentar- und Leerzeilen für die simulierbare Verhaltensbeschreibung an. Die Spalte **"min csteps"** gibt die theoretisch kleinste Anzahl Kontrollschritte des Moduls an, bedingt durch die eingefügten "Wait"-Anweisungen.

Im zweiten Block **"Monet"** sind die Ergebnisse der Synthese mit Monet zusammengefaßt.

Das Modul CC (Verbindungssteuerung) konnte nur bis zum "Scheduling-Schritt" synthetisiert werden. Danach hat Monet mit einem Fehler die Synthese abgebrochen. Es fehlen die Abschätzungen für Area und Delay, sowie die FSM-Daten.

Behavioral VHDL Description				Monet						
Modul	Process	nc-loc	min csteps	nc-loc	Area (DP)	Delay (ns)	#csteps	#ops	FSM States	FSM Area
AHT_IN	AHTIN	41	9	60	1843	18	10	32	9	905
	Payload	30	7	31			52	238	51	15809
HT	Htproc	56	7	89	1243	2.3	7	46	7	430
RC	RCIN	94	15	155	1689	6.9	17	172	18	3590
	RCOUT	22	6	48			6	39	7	517
SRC	Shiftproc	123	15	153	7121	7.0	17	710	16	2093
CC	CCin	152	13	178	--	--	17	352	--	--
RD	RDIN	158	9	204	14381	10.1	10	701	10	1042
	RDOUT	20	6	33			6	29	5	357
AHT_OUT	AHTOUT	35	10	71	1338	23.4	29	197	30	7676
	AHTOUT_T	35	15	47			52	162	57	19266

Abbildung 34: Zusammenfassung der Syntheseergebnisse

Die **“nc-loc”** Werte sind höher als die der VHDL-Beschreibung, bedingt durch zusätzliche Attribute, Wait-Anweisungen, und folgende funktionale Änderungen:

1. Prozeß Payload: Um Verzögerungen zu vermeiden, wird die Weitergabe der Zelldaten an den Cell_FIFO-Speicher synchron anstatt asynchron durchgeführt.

2. Prozeß RCIN: Hier wird die umgekehrte Vorgehensweise eingeführt: Um den Zellkopf sicher vom Head_FIFO zu laden, wird die “Two way handshaking” Kommunikation statt der synchronen Übertragung angewendet.

3. Prozeß Shift_Proc: Das Schieberegister wird ausgelagert. Es kann als externe Komponente auf Logikebene in VHDL geschrieben und zugefügt werden.

4. Prozeß RDIN: Auch auf der Datenannahmeseite des Switches wird das Schieberegister ausgelagert.

Bei der Chipfläche **(Area)** ist die Flächeneinheit 1 die Fläche einer LCA-Basiszelle (eines 1-Bit-Inverters).

Die Fläche ist für den Datenpfad und für die FSM getrennt angegeben. Sie ist für den Datenpfad nach dem Schritt “Share Resources” abgeschätzt worden. Dies kommt einem ersten Optimierungsschritt für die Datenpfad-Schaltung gleich.

Die Flächen sind für die Module SRC und RD relativ groß. Das kommt daher, daß für die Verschlüsselung in SRC und für die Entschlüsselung in RD keine separate Komponente sondern eine Funktion verwendet wird. Es gibt bei Monet kein “preserve_function”-Attribut, das eine Funktion automatisch wie eine separate Komponente einbindet. Monet

setzt die Ver- und Entschlüsselungsfunktion bei jedem Aufruf inline und erhöht damit die Fläche.

Separate Komponenten zu definieren ist aufwendig. Es wird in diesem Rahmen darauf verzichtet.

Als Verzögerung **(Delay)** wird die größte Verzögerung (Cumulative Delay) aus dem "Timing-Report" genommen. Diese Verzögerung wird verursacht durch die "langsamste" Operation plus eventuell sequentielle Logik in dem entsprechenden Prozeß. Wichtig ist, daß diese Verzögerung kleiner ist als die Taktperiode (25 ns), sonst wird im Scheduling-Schritt eine Taktperiode hinzugefügt.

Die Anzahl Kontrollschritte **(#csteps)** sind bei Monet gleich oder höher als in der VHDL-Beschreibung, da vom Scheduler im "superstate fixed" I/O-Modus zusätzliche Kontrollschritte eingefügt werden.

Bei den Prozessen Payload (Modul AHT_IN) und AHTOUT_T (letzter Prozeß) generiert Monet 52 Kontrollschritte. Dies kommt durch das "Aufrollen" der Schleifen. Dadurch wird eine kontinuierliche Datenübertragung erreicht, allerdings werden damit die FSM's in Bezug auf die Anzahl Zustände und Flächeneinheiten relativ groß.

11.2.2 Rechenzeit, Speicherbedarf und Zuverlässigkeit

Die Tabelle Abbildung 35 zeigt die Ergebnisse der Synthese der einzelnen Module und Prozesse in Bezug auf Rechenzeit, Speicherbedarf und Zuverlässigkeit.

	Monet		
Modul	Rechen-zeit (sec)	Speicher-bedarf (MB)	Zuver-lässigkeit (Z)
AHT_IN	ca. 14	≤ 64	10
HT	ca. 4		10
RC	ca. 12		8
SRC	ca. 16		10
CC	ca. 16		2
RD	ca. 16		10
AHT_OUT	ca. 55		10

Abbildung 35: Zusammenfassung der Syntheseergebnisse in Bezug auf Rechenzeit, Speicherbedarf und Zuverlässigkeit

Als **Rechenzeit** wird die Zeit für den Schritt angegeben, der im Syntheseablauf am längsten dauert. Meist ist dies die Zeit für den "elaborate"-Schritt. Beim Modul AHT_OUT

benötigt der “Share Resources” Schritt die längste Zeit. Die Monet-Rechenzeiten können nur ungefähr angegeben werden, da sie manuell gemessen wurden. Eine automatische Messung ist hier nicht möglich, da die einzelnen Schritte aus der graphischen Oberfläche mit “Mausklick” selektiert werden. Die Rechenzeit kann nur ein Richtwert sein, da sie auf einem Compute-Server gemessen wurde, der auch anderen Benutzern zur Verfügung steht, d.h. daß die Rechenzeit auch von der momentanen Nutzung des Compute-Servers abhängt.

Speicherbedarf
Monet reichen für alle Module 64 MB Arbeitsspeicher.

Die **Zuverlässigkeit** ist in Kapitel 3 definiert. Das Zuverlässigkeitsmaß Z gibt an, in welchem Maß das Ergebnis der Synthese korrekt und akzeptabel ist.

11.2.3 Dokumentation und Einarbeitungszeit

Dokumentation
Das “Training Workbook” [Montw98] für Monet ist übersichtlich und gut verständlich.

Darüberhinaus gibt es bei Monet relativ wenig Dokumentation [Mum98]. D.h. manche Dinge sucht man vergebens. (Beispiel: Wie wird eine VHDL-Funktion als Komponente eingebunden?)

Die **Einarbeitszeit** für Monet ist relativ kurz. Das “Monet Training Workbook” enthält beinahe alles Wissenswertes. Zusätzlich wird der Benutzer durch eine gute graphische Oberfläche durch alle Syntheseschritte geführt und kommt mit allen wesentlichen Optionen in Berührung.

12 Anhang: Quellcodes und Berichte (reports) der High-Level-Synthese

Um den Anhang nicht zu überladen, wurden überlange Operation-count-Listen und Verzögerungs-Pfad-Listen gekürzt. Die Kürzung ist jeweils angezeigt mit “... removed w.l.”

12.1 Eingangsmodul AHT_IN

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des AHT_In fuer
-- Simulation mit qhsim
-- Synthese mit Monet
-- Cellfifo muesste mit minimal ein viertel der Clk_AHT_IN Rate laufen
-- AHT_In hat 2 Prozesse: einen fuer die Aufnahme und asynchrone Weiterleitung
-- des Zell-Headers, der zweite Prozess nimmt die Payload auf,  und die
-- asynchrone Weitergabe des Payload Registers an den Cell-Fifo.
-- Aenderung fuer Monet:
-- Synchrone resets und Waits nach Monet-Muster einfuehren!
-- dont_unroll - Attribut eingefuehrt (21. 8. 98)
-- es kann leider nicht beutzt werden..
-- Attribute sync_reset eingefuehrt
-- Autor: W. Lange 20. 5. 98 und 21. 8. 98

Library IEEE;
USE IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
--    use STD.Textio.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY aht_in IS
  PORT (Data_In     : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        clk_aht_in   : IN STD_LOGIC;
        cell_sync_in : IN STD_LOGIC;
        reset        : IN STD_LOGIC;
        Hd_fetch     : IN STD_LOGIC;
        Cell_fetch   : IN STD_LOGIC;
        Buffer_full  : IN BOOLEAN;
        Hd_Bus       : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
        Cell_Bus     : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
        Hd_rdy       : OUT STD_LOGIC;
        Cell_rdy     : OUT STD_LOGIC);
END aht_in;
---------------------------------

ARCHITECTURE aht1ar OF aht_in IS
  SIGNAL Cell_part :  STD_LOGIC_VECTOR(31 DOWNTO 0);
  SIGNAL Start_Payl:  STD_LOGIC;
  SIGNAL Cell_ack  :  STD_LOGIC;

  ATTRIBUTE sync_reset OF aht1ar : ARCHITECTURE IS "reset" ;
  ATTRIBUTE reset_phase OF AHTIN : LABEL IS "positive" ;
  ATTRIBUTE reset_phase OF Payload : LABEL IS "positive" ;


BEGIN

 AHTIN: PROCESS
  -- Der AHTIN Process nimmt den Header auf und leitet ihn asynchron
  -- an HT weiter.
     VARIABLE Hd_reg    : STD_LOGIC_VECTOR(31 downto 0);  --Reg fuer Header B.


 BEGIN -- Process

   --  Reset: Setze alle Variablen und Indices zurueck ----------------------

     Hd_rdy <= '0';
     Start_Payl <= '0';
     Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

    ------------------------ Haupt-Loop ---------------------------------
    -- Cell_Sync_In erscheint, eine neue Zelle kommt an...

    WHILE  NOT Buffer_full  loop    -- main loop ------------------------

      Start_Payl <= '0';

      -- Monet: Fuer jedes synchtrone Wait: eine separate loop
      -- Wait UNTIL Clk_AHT_In'EVENT AND ....  and Cell_Sync_In = '1';

      cell_sync_loop: loop     -- fuer monet
```

```
wait until clk_aht_in'event and clk_aht_in = '1';
       Hd_reg(31 downto 24) := Data_In;
exit cell_sync_loop when (cell_sync_in = '1');
      end loop cell_sync_loop;

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1'; -- fuer monet
      -- Cell_Sync ist da, lade den Header..     ---------------------
      Hd_reg(31 downto 24) := Data_In;           -- 1. Byte

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

      Hd_reg(23 downto 16) := Data_In;           -- 2. Byte

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

      Hd_reg(15 downto 8) := Data_In;            -- 3. Byte

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

      Hd_reg(7 downto 0) := Data_In;             -- 4. Byte

      -- Das HEC Byte bleibt unberuecksichtigt.
      --   Hd_Bus <= Hd_reg;    -- Setze das Hdreg. auf den Bus und
      --   Hd_rdy <= '1';       -- Vergiss den HEC...
      -- Start_Payl <= '1';

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';  -- HEC Byte
         Hd_Bus <= Hd_reg;    -- Setze das Hdreg. auf den Bus
         Hd_rdy <= '1';       -- Vergiss den HEC...
         Start_Payl <= '1'; -- Aktiviere den Payload Prozess

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';


      -- Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1' and Hd_fetch = '1';
      Hd_fetch_loop: loop     -- fuer monet
wait until clk_aht_in'event and clk_aht_in = '1';
exit Hd_fetch_loop when ( Hd_fetch = '1' );
      end loop Hd_fetch_loop;

      Hd_rdy <= '0';
      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

    End loop; -- Main Loop
 End Process;


  Payload: PROCESS
  -- Der Payload Process nimmt den Header auf und leitet ihn asynchron
  -- an den Cell-Fifo weiter.

  VARIABLE Cell_reg    : STD_LOGIC_VECTOR(31 downto 0);  -- Reg fuer Cell Bytes
  VARIABLE Cell_reg2   : STD_LOGIC_VECTOR(31 downto 0);  -- Reg fuer Cell Bytes
  -- attribute dont_unroll of L1 : label is true;

  BEGIN -- Process

    --  Reset: Setze alle Variablen und Indices zurueck ---------------

     Cell_rdy  <= '0';
     Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

--  Die Payload wird geladen

   Payl: loop    -- main loop ---------------------------

      Cell_rdy <= '0';

      -- Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1' and Start_Payl = '1';

      start_payl_loop: loop     -- fuer monet
wait until clk_aht_in'event and clk_aht_in = '1';
exit start_payl_loop when ( Start_Payl = '1' );
      end loop start_payl_loop;

      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1'; -- fuer monet

      L1: FOR k in 1 TO 12 loop
      --  4 Byte der Zelle werden geladen..

--        Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1'; -- fuer monet

Cell_reg(31 downto 24) := Data_In;   -- 1. Byte 153

        Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

Cell_reg(23 downto 16) := Data_In;   -- 2. Byte  157

Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

Cell_reg(15 downto 8) := Data_In;    -- 3. Byte  161

        IF Cell_fetch = '1' THEN Cell_rdy <= '0';  END IF;
```

```
Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

Cell_reg(7 downto 0) := Data_In;     --  4. Byte  167

  --      Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

Cell_Reg2 := Cell_reg;
        Cell_Bus <= Cell_reg2;   -- Lade 4 Bytes
        Cell_rdy <= '1';

         Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

        end loop;  -- (For k in 1 to 12)

     Wait UNTIL Clk_AHT_In'EVENT AND Clk_AHT_In = '1';

    End loop;  -- Payload loop   ---------------------------------------

 END Process;  -- Payload Process

END aht1ar;
```

Monet-Scheduling-Report

```
Solution: Solution_1
Project: Untitled
Project File:
/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/ahtin_mon
/Untitled.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src
/std_logic_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs
/qhdl_libs/src/syn_arit.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET
/monet/pkgs/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}
 {/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/ahtin_mon/src
/ahtin_m.vhd -Lwork -87}

extended operation counts and delay paths removed. w.l.

Processes in design:
   ahtin payload

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: ahtin
------------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     0 (0 are filtered out)
      "Pseudo" operations: 238

   Clock Information
   -----------------
      Clock Signal: clk_aht_in (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
                                                               -------------
                                                    Total Area:      0.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ------------------------------
          0    IO_W=>hd_rdy              59   &0             PSEUDO
          1    ASSIGN                    60   &2             PSEUDO
         10    RISINGEDGE                79                  PSEUDO
         11    IO_R<=data_in             81                  PSEUDO
         12    RISINGEDGE                83                  PSEUDO
```

```
      13    IO_R<=data_in            85                  PSEUDO
      14    RISINGEDGE               87                  PSEUDO
      15    IO_R<=data_in            89                  PSEUDO
      16    RISINGEDGE               91                  PSEUDO
      17    IO_R<=data_in            93                  PSEUDO
      18    RISINGEDGE              100                  PSEUDO
      19    IO_W=>hd_bus            101   &3             PSEUDO
       2    RISINGEDGE               61   &1             PSEUDO
      20    IO_W=>hd_rdy            102   &5             PSEUDO
      21    ASSIGN                  103   &6             PSEUDO
      22    RISINGEDGE              105   &4             PSEUDO
      23    LOOP                    109   hd_fetch_l     PSEUDO
      24    RISINGEDGE              110                  PSEUDO
      25    TERMINATE               111                  PSEUDO
      26    IO_W=>hd_rdy            114   &7             PSEUDO
      27    RISINGEDGE              115   &8             PSEUDO
      28    LOOP                     52   ahtin_loop     PSEUDO
      29    LOOPEND                  52                  PSEUDO
       3    LOOP                     66   loop66         PSEUDO
      30    LOOPEND                  66                  PSEUDO
      31    LOOPEND                  73                  PSEUDO
      32    LOOPEND                 109                  PSEUDO
       4    TERMINATE                66                  PSEUDO
       5    ASSIGN                   68                  PSEUDO
       6    LOOP                     73   cell_sync_     PSEUDO
       7    RISINGEDGE               74                  PSEUDO
       8    IO_R<=data_in            75                  PSEUDO
       9    TERMINATE                76                  PSEUDO

  Cycle Constraints
  -----------------
     From                     To                      Constraint   Actual  Met?
     -----------------------  ----------------------  ----------   ------  ----

  Scheduled Operations
  --------------------
     TOTAL SCHEDULE LENGTH   10 C-Steps

     Oper     Operation             Line
     Number   Type                  Number   Label          C-Step  Delay
     ------   --------------------  ------   ----------     ------  -----
        28    LOOP                      52   ahtin_loop        1    0.000
        29    LOOPEND                   52                    10    0.000
         0    IO_W=>hd_rdy              59   &0                 1    0.001
         1    ASSIGN                    60   &2                 1    0.001
         2    RISINGEDGE                61   &1                 2    0.001
         3    LOOP                      66   loop66             2    0.000
         4    TERMINATE                 66                      2    0.001
        30    LOOPEND                   66                    10    0.000
         5    ASSIGN                    68                      2    0.001
         6    LOOP                      73   cell_sync_         2    0.000
        31    LOOPEND                   73                      2    0.000
         7    RISINGEDGE                74                      2    0.001
         8    IO_R<=data_in             75                      2    0.001
         9    TERMINATE                 76                      2    0.001
        10    RISINGEDGE                79                      3    0.001
        11    IO_R<=data_in             81                      3    0.001
        12    RISINGEDGE                83                      4    0.001
        13    IO_R<=data_in             85                      4    0.001
        14    RISINGEDGE                87                      5    0.001
        15    IO_R<=data_in             89                      5    0.001
        16    RISINGEDGE                91                      6    0.001
        17    IO_R<=data_in             93                      6    0.001
        18    RISINGEDGE               100                      7    0.001
        19    IO_W=>hd_bus             101   &3                 7    0.001
        20    IO_W=>hd_rdy             102   &5                 7    0.001
        21    ASSIGN                   103   &6                 7    0.001
        22    RISINGEDGE               105   &4                 8    0.001
        32    LOOPEND                  109                      8    0.000
        23    LOOP                     109   hd_fetch_l         8    0.000
        24    RISINGEDGE               110                      8    0.001
        25    TERMINATE                111                      8    0.001
        26    IO_W=>hd_rdy             114   &7                 9    0.001
        27    RISINGEDGE               115   &8                10    0.001

  Pipelined Loops
  ---------------
                                                 Initiation
     Loop                          N_Unroll     Interval    Latency
     ----------------------------  --------   ----------   -------

  Loops
  -----
                                  Length  Start   End              Initiation
     Loop                         C-step C-step C-steps  N_Unroll   Interval   Latency
     ---------------------------  ------ ------ -------  --------  ----------  -------
     loop66                          9      2      10
       cell_sync_loop                1      2       2
       hd_fetch_loop                 1      8       8


Process: payload
----------------------------------------------------
```

```
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     0 (0 are filtered out)
      "Pseudo" operations: 238

   Clock Information
   -----------------
      Clock Signal: clk_aht_in (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
                                                               -------------
                                                 Total Area:      0.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ------------------------------
      204 operations removed w.l.



   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      ----------------------  ----------------------   ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   52 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
        177    LOOP                     125   payload_lo        1    0.000
        204 operations removed w.l.
        176    RISINGEDGE               182                    52    0.001

   Pipelined Loops
   ---------------
                                          Initiation
      Loop                          N_Unroll    Interval    Latency
      ----------------------------   --------   ----------   -------

   Loops
   -----
                                   Length  Start   End              Initiation
      Loop                         C-step C-step C-steps  N_Unroll   Interval   Latency
      ---------------------------  ------ ------ -------  --------  ----------  -------
      payl                           51      2      52
        start_payl_loop               1      2       2

Area Estimate
-------------
   Component                                    Post   Post   Post
   Name                                  Area   Alloc  Bind  Share
   ---------                          ----------  -----  -----  -----
   AndGate(1,2)                          2.000      0      3      3
   InvGate(1)                            1.000      0      4      4
   Mux1h(1,2)                            3.000      0      2      2
   Mux1h(1,4)                            5.000      0      1      1
   Mux1h(32,2)                          96.000      0      1      1
   Mux1h(32,4)                         160.000      0      1      1
   Mux1h(32,5)                         224.000      0      1      1
   NorGate(1,13)                         9.125      0      1      1
   NorGate(1,14)                         9.750      0      1      1
   NorGate(1,3)                          2.000      0      1      1
   OrGate(1,2)                           2.000      0    110    110
   Register(1,0,0)                       7.000      0      3      3
   Register(32,0,0)                    224.000      0      4      4
   mux_mux2(2,32)                       89.437      0      2      2
   rdcnor_four(4,5)                      3.015      0      1      1
   rdcnor_three(4,4)                     2.000      0      1      1

      TOTAL AREA (After Sharing):               1842.770

Area Estimate
-------------
```

```
                Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area          0.0               0.0            1842.8            1842.8
     Total Reg         0.0               0.0             917.0  (50%)      917.0  (50%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath          0.0               0.0            1842.8 (100%)     1842.8 (100%)
       MUX             0.0               0.0             669.9  (36%)      669.9  (36%)
       FUNC            0.0               0.0               5.0   (0%)        5.0   (0%)
       LOGIC           0.0               0.0             250.9  (14%)      250.9  (14%)
       BUFFER          0.0               0.0               0.0               0.0
       DP-REG          0.0               0.0             917.0  (50%)      917.0  (50%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM               0.0               0.0               0.0               0.0
       FSM-REG         0.0               0.0               0.0               0.0
       FSM-COMB        0.0               0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
--------------
   Critical Path
     Max Delay 17.96
     Slack -18.96
     Resource: r507rg(OrGate_1_2) Signal: n_127tmp Unit delay: 0.33 Delay: 0.33
     51 resources removed w.l.
     Resource: r322rg(Mux1h_1_4) Signal: cell_rdy_pre_ff Unit delay: 0.49 Delay: 17.96

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: -1
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: -1
       Resource r259rg Port hd_bus_pre_ff -1.5 (clock period exceeded)
       Resource r272rg Port hd_rdy_pre_ff -4.98 (clock period exceeded)
       Resource r287rg Port start_payl_pre_ff -5.31 (clock period exceeded)
       Resource r300rg Port hd_reg_pre_ff -3.35376 (clock period exceeded)
       Resource r311rg Port cell_bus_pre_ff -5.87554 (clock period exceeded)
       Resource r317rg Port cell_reg_pre_ff -5.63 (clock period exceeded)
       Resource r323rg Port cell_rdy_pre_ff -18.96 (clock period exceeded)

   FSM name: ahtin (aht_in_sid0_ahtin)
     Total area: 905
       Combinational area: 640 Sequential area: 265

     Input to output delays

     Input to register-input delays
       buffer_full: 0.35
       reset: 0.35
       hd_fetch: 0.35
       cell_sync_in: 0.35

     Register-output to output delays
       clk_aht_in->n_49_fsm: 0
       0 delay-paths removed w.l.
       clk_aht_in->n_27_fsm: 0


   FSM name: payload (aht_in_sid0_payload)
     Total area: 15809
       Combinational area: 14443 Sequential area: 1366

     Input to output delays

     Input to register-input delays
       start_payl: 0.35
       reset: 0.35

     Register-output to output delays
       clk_aht_in->n_273_fsm: 0
       98 delay paths removed w.l.
       clk_aht_in->n_157_fsm: 0


   ----------------------------------------------------------------

Variables Mapped to Memories
-----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Finite State Machines
---------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r311rg          32 bits    cell_bus
   r287rg           1 bits    start_payl
   r317rg          32 bits    cell_reg
   r300rg          32 bits    hd_reg
   r272rg           1 bits    hd_rdy
   r323rg           1 bits    cell_rdy
   r259rg          32 bits    hd_bus
```

```
Filter Settings:
   Pseudo-Operation Filter:
          IO_W                  PADZEROES             LOOPEND
          IO_R                  SIGNEXTEND            SELECT
          READSLICE             NOP                   SELECTEND
          WRITESLICE            END                   CONDITIONALTRANSFER
          ASSIGN                ITERATE               RISINGEDGE
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R         WRITEINDEX   OR           XNOR         SKEDLIST
          IO_W         AND          NOR          NOT
          READINDEX    NAND         XOR          MUX
   Schedule Filter (default):
          PADZEROES             LOOPEND               RISINGEDGE
          SIGNEXTEND            SELECT                FALLINGEDGE
          END                   SELECTEND
          LOOP                  CONDITIONALTRANSFER

End of Report
```

12.2 Zellkopfübersetzungsmodul HT

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des HT (Header Translator)  fuer Monet
-- und Mentor-Simulator
-- ht_mon/ht_m.vhd
-- Autor: W. Lange. Letzte Aenderung: 12. 5. 1998
-- Aenderungen fuer Monet:
-- 1. Jedes synchrone Wait muss in eine loop umgeschrieben werden.

Library IEEE;
USE IEEE.std_logic_1164.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY ht IS
  PORT ( Clk_com     : IN  STD_LOGIC ;       -- Clock common
Hd_Bus       : IN  STD_LOGIC_Vector(31 DOWNTO 0); -- Header - 4 Bytes
Hd_rdy       : IN  STD_LOGIC;   -- AHT1 has Header for you
Hd_fetch     : OUT STD_LOGIC;   -- Header arrived in HT: Header fechted
RT_Addr      : OUT STD_LOGIC_Vector(15 downto 0);
rt_read      : OUT STD_LOGIC;
RT_data      : IN  STD_LOGIC_Vector(39 downto 0);
       rtdta_strobe : IN  STD_LOGIC;
fwrq         : OUT STD_LOGIC;                      -- Fifo write re.
  headout      : Out STD_LOGIC_Vector(47 downto 0);   -- Fifo input
       head_taken   : IN  STD_LOGIC;
ffull        : IN  STD_LOGIC;                       -- fifo ist voll
       Reset        : IN  STD_LOGIC);
END ht;
---------------------------------

ARCHITECTURE htar OF ht IS

BEGIN
 htproc : PROCESS

  VARIABLE  Address    : STD_LOGIC_Vector(0 TO 15);
  VARIABLE  Header     : STD_LOGIC_Vector(47 DOWNTO 0);  -- Header w. RTag
  VARIABLE  rtag       : STD_LOGIC_Vector(15 DOWNTO 0);  -- Hilfsvar. f.rtag
  VARIABLE  vpivci     : STD_LOGIC_Vector(23 DOWNTO 0);  -- Hilfsvar. f. vpivci
  VARIABLE  rtdata     : STD_LOGIC_Vector(39 downto 0);  -- Hilfsvar.
  VARIABLE  rdwrt      : STD_LOGIC;

 BEGIN -- Process ------------------------------------------------------

   reset_loop: loop
     Hd_fetch <= '0';  -- Setze die Signale auf null..
     fwrq     <= '0';
     rt_read  <= '0';
     --   Headfifo_reset;    -- Reset Header - Puffer (noch nicht eingebaut)

    --  WAIT UNTIL Clk_com'event and Clk_com = '1'; (mon)

     ------------------------ Haupt-Loop ---------------------------------
     WHILE TRUE LOOP

     -- Hole einen Header vom AHT1_In ab                ---------------
     --------------------Warte auf Hd_rdy_up  --------------------------
     hd_rdy_loop: loop                            -- sync. loop (mon)
           WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
           exit reset_loop when Reset = '1';      -- Synchronous reset (mon)
   IF (Hd_rdy = '1') THEN
      EXIT hd_rdy_loop;
           END IF;
     end loop;
     ------------------------------------------------------------------------
```

```
      Header(31 downto 0) := Hd_Bus;
      Hd_fetch <= '1';                -- Sag AHT1_In: Header abgeholt!
    ------------------------Warte auf Hd_rdy_dwn   ---------------------
    hd_rdy_dwn_lp: loop                          -- sync. loop (mon)
          WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
          exit reset_loop when Reset = '1';      -- Synchronous reset (mon)
          IF (Hd_rdy = '0') THEN
              exit hd_rdy_dwn_lp;
          END IF;
    end loop;
   ---------------------------------------------------------------------
    --  Adressiere die Routing-Tabelle VPI_2, VCI_2
      Address := Header(27 downto 24) & Header(15 downto 4); -- Get VCI/VPI
      -- Hier kann man die Adressrechnung einsetzen Addr:=F1(Addr);
      -- Adressiere die RT und hole die neuen VPI und VCI's
      RT_Addr <= Address; -- Setze die Adresse
      rt_read <= '1';
      Hd_fetch <= '0';
    ------------ Warte auf das RT-Data Ergebnis ------------------------
    rt_lp: loop                              -- sync. loop (mon)
          WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
          exit reset_loop when Reset = '1';      -- Synchronous reset (mon)
          IF ( rtdta_strobe = '1') THEN
              exit rt_lp;
          END IF;
    end loop;
   ---------------------------------------------------------------------
      rtdata := RT_data;
      rtag := rtdata(39 downto 24);      -- Hilfsvariablen wegen CADDY
      vpivci := rtdata(23 downto 0);   -- Hilfsvariablen wegen CADDY
      -- Header(31 DOWNTO 28) ist GFC, von Bit 3 bis 0: PTI und CLP
      Header(47 downto 4) := rtag & Header(31 DOWNTO 28) & vpivci;

      rt_read <= '0';

   ------ Speichere das Headerregister im Headerpuffer ab    ----------
    f_full_lp: loop                           -- sync. loop (mon)
          WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
          exit reset_loop when Reset = '1';      -- Synchronous reset (mon)
          IF ( ffull   = '0') THEN
              exit f_full_lp;
          END IF;
    end loop;
   ---------------------------------------------------------------------
       fwrq <= '1';
       headout <= Header;
    ----------------------- 5. Takt ---------------------------------
    hd_taken_lp: loop                          -- sync. loop (mon)
          WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
          exit reset_loop when Reset = '1';      -- Synchronous reset (mon)
          IF ( head_taken  = '1') THEN
              exit hd_taken_lp;
          END IF;
    end loop;
   ---------------------------------------------------------------------
      fwrq <= '0';
      WAIT UNTIL Clk_com'event and Clk_com = '1';
      exit reset_loop when Reset = '1';      -- Synchronous reset (mon)

    END LOOP; --

  end loop; -- reset loop

 END Process;

END;           -- End architecture
```

Monet-Scheduling-Report

```
Solution: Solution_1
Project: ht
Project File: /afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht
/ht_mon/ht.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/std_logic_1164.vhd
-Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/hls_pkgs/src/attributes.vhd
-Lmgc_hls -93} {/afs/informatik.uni-tuebingen.de/home/wlange
/hilan/aht/ht_mon/ht_m.vhd -Lwork -87}

Processes in design:
   htproc

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: htproc
```

```
--------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     2 (2 are filtered out)
      "Pseudo" operations: 44

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      InvGate(1)                             1.00         1           1.00
                                                             -------------
                                                 Total Area:         1.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ------------------------------
          0    LOOP                      46   reset_loop     PSEUDO
    45 operations removed w.l.
          9    IO_R<=hd_bus              67                  PSEUDO

   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      -----------------------  -----------------------  ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH    7 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
          0    LOOP                      46   reset_loop        1    0.000
         43 operations removed w.l.
         38    RISINGEDGE               123   &1                7    0.001

   Pipelined Loops
   ---------------
                                                Initiation
      Loop                           N_Unroll    Interval    Latency
      -----------------------------  --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ----------------------------  ------ ------ -------  --------  ----------  -------
      loop55                           7      1       7
        hd_rdy_loop                    1      1       1
        hd_rdy_dwn_lp                  1      2       2
        rt_lp                          1      3       3
        f_full_lp                      1      4       4
        hd_taken_lp                    1      5       5

Area Estimate
-------------
   Component                                    Post   Post   Post
   Name                                 Area    Alloc  Bind   Share
   ---------                         ---------  -----  -----  -----
   InvGate(1)                            1.000      1      5      5
   AndGate(1,2)                          2.000      0      3      3
   Mux1h(1,2)                            3.000      0      3      3
   Mux1h(16,2)                          48.000      0      1      1
   Mux1h(48,2)                         144.000      0      1      1
   Mux1h(48,3)                         192.000      0      1      1
   NorGate(1,2)                          1.000      0      1      1
   OrGate(1,2)                           2.000      0      8      8
   Register(1,0,0)                       7.000      0      3      3
   Register(16,0,0)                    112.000      0      1      1
   Register(48,0,0)                    336.000      0      2      2
   rdcnor_four(4,8)                      5.537      0      3      3

      TOTAL AREA (After Sharing):               1242.610

Area Estimate
```

```
-------------
                 Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area          1.0               1.0            1242.6            1242.6
     Total Reg         0.0               0.0             805.0  (65%)      805.0  (65%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath          1.0 (100%)        1.0 (100%)     1242.6 (100%)     1242.6 (100%)
       MUX             0.0               0.0             393.0  (32%)      393.0  (32%)
       FUNC            0.0               0.0              16.6   (1%)       16.6   (1%)
       LOGIC           1.0 (100%)        1.0 (100%)       28.0   (2%)       28.0   (2%)
       BUFFER          0.0               0.0               0.0               0.0
       DP-REG          0.0               0.0             805.0  (65%)      805.0  (65%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM               0.0               0.0               0.0               0.0
       FSM-REG         0.0               0.0               0.0               0.0
       FSM-COMB        0.0               0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Critical Path
     Max Delay 2.33
     Slack 22.67
     Resource: r317rg(OrGate_1_2) Signal: n_12tmp Unit delay: 0.33 Delay: 0.33
     Resource: r320rg(OrGate_1_2) Signal: n_13tmp Unit delay: 0.33 Delay: 0.66
     Resource: r323rg(OrGate_1_2) Signal: n_14tmp Unit delay: 0.33 Delay: 0.99
     Resource: r326rg(OrGate_1_2) Signal: n_15tmp Unit delay: 0.33 Delay: 1.32
     Resource: r329rg(OrGate_1_2) Signal: n_16tmp Unit delay: 0.33 Delay: 1.65
     Resource: r332rg(OrGate_1_2) Signal: n_17tmp Unit delay: 0.33 Delay: 1.98
     Resource: r27rg(InvGate_1) Signal: n_26tmp Unit delay: 0.09 Delay: 2.07
     Resource: r30rg(AndGate_1_2) Signal: hd_fetch_pre_ff Unit delay: 0.26 Delay: 2.33

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: 25
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: 25
       Resource r31rg Port hd_fetch_pre_ff 22.67
       Resource r40rg Port header_pre_ff 24.33
       Resource r55rg Port fwrq_pre_ff 22.67
       Resource r64rg Port headout_pre_ff 24.5
       Resource r71rg Port rt_addr_pre_ff 24.5
       Resource r84rg Port rt_read_pre_ff 22.67

   FSM name: htproc (ht_sid0_htproc)
     Total area: 430
       Combinational area: 279 Sequential area: 151

     Input to output delays

     Input to register-input delays
       head_taken: 0.35
       ffull: 0.44
       reset: 0.35
       hd_rdy: 0.44
       rtdta_strobe: 0.35

     Register-output to output delays
       clk_com->n_2_fsm: 0
       clk_com->n_19_fsm: 0
       clk_com->n_4_fsm: 0
       clk_com->n_6_fsm: 0
       clk_com->n_8_fsm: 0
       clk_com->n_10_fsm: 0
       clk_com->n_12_fsm: 0
       clk_com->n_14_fsm: 0
       clk_com->n_23_fsm: 0
       clk_com->n_25_fsm: 0
       clk_com->n_0_fsm: 0


   ----------------------------------------------------------------

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Finite State Machines
---------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r64rg           48 bits    headout
   r55rg            1 bits    fwrq
   r84rg            1 bits    rt_read
   r71rg           16 bits    rt_addr
   r40rg           48 bits    header
   r31rg            1 bits    hd_fetch

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                 PADZEROES            LOOPEND
          IO_R                 SIGNEXTEND           SELECT
```

```
          READSLICE             NOP                   SELECTEND
          WRITESLICE            END                   CONDITIONALTRANSFER
          ASSIGN                ITERATE               RISINGEDGE
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R         WRITEINDEX   OR           XNOR         SKEDLIST
          IO_W         AND          NOR          NOT
          READINDEX    NAND         XOR          MUX
   Schedule Filter (default):
          PADZEROES             LOOPEND               RISINGEDGE
          SIGNEXTEND            SELECT                FALLINGEDGE
          END                   SELECTEND
          LOOP                  CONDITIONALTRANSFER

End of Report
```

12.3 Routing-Steuerung (RC)

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des RC fuer
-- Synthese mit Monet
-- rc_bc.vhd hat ein two-way handshaking IF auch zum Header_FIFO
-- rc_bc.vhd hat Interfaces zu Head_FIfo, CC und SRC_Out,
-- in dieser Version wurde die Abfrage von 'free' umgestellt.
-- dadurch ist die Simulation stabiler.
-- moegliche Verbesserungen: siehe notes.
-- 2 Prozesse werden verwendet RCIN, RCOUT.
-- RCIN nimmt header und Routing Tag auf, untersucht den Routing Tag
-- und kommuniziert mit dem Connection Controller (CC)
-- RCOUT gibt den header an SRC weiter und kommuniziert mit SRC
-- multicast muss sein, damit der CC weiss, wann "configure" gegeben wird
-- Das Signal free gibt an, wann xmit_bsy nach einer Uebertragung
-- wieder aktiv werden darf, d.h. wann die Belegung eines Ausgangs
-- aufgehoben ist.
-- Aenderungen, um v_parser zum Laufen zu bringen:
-- use IEEE.std_logic_unsigned.all; auskommentiert
-- Aenderungen fuer BC:
-- Attribute: dont_unroll fuer die LOOPS.
-- viele Wait's gesetzt um HLS-45/etc. Fehler zu eleminieren.
-- Autor: W. Lange.   16. 2. 98
-- Umgeschrieben fuer Monet: 27. 5. 98
-- Attribute: dont_unroll (LOOP L1) auch bei Monet eingefuehrt. (28.7.89)

Library IEEE;
USE IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY rc IS
  PORT (
       reset        : IN STD_LOGIC;
       Clk_com      : IN STD_LOGIC;
       Hd_read      : OUT STD_LOGIC;            -- HEADER_FIFO
       Hd_write     : IN  STD_LOGIC;
       Hd_Data      : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
       Hd_loaded    : OUT STD_LOGIC; -- for tw-handshake
       busy         : IN  STD_LOGIC;            -- SRC_OUT
       transmit     : OUT STD_LOGIC;
       load_hd      : OUT STD_LOGIC;
       header       : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
       load_ok      : IN  STD_LOGIC;
       multicast    : OUT STD_LOGIC;   -- CC
       RR           : OUT STD_LOGIC;
       OA           : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
       WAIT_CC      : IN  STD_LOGIC;
       free         : IN STD_LOGIC;
       xmit_bsy     : OUT STD_LOGIC);
 END rc;
---------------------------------
-- Use work.utilssrc.all;

ARCHITECTURE rcar OF rc IS
  SIGNAL connected    : BOOLEAN;
  SIGNAL head_reg     : STD_LOGIC_VECTOR(31 DOWNTO 0);
  SIGNAL hd_reg_bsy   : BOOLEAN;
  SIGNAL hd_r_b       : BOOLEAN;    -- sagt RCOUT proc, head_reg busy
  SIGNAL head_ready   : BOOLEAN;

BEGIN

 RCIN: PROCESS -- Der SRCIN Process nimmt den Header und den R-tag auf
   VARIABLE r_tag        :  STD_LOGIC_VECTOR(15 DOWNTO 0);
   VARIABLE r_tag_old    :  STD_LOGIC_VECTOR(15 DOWNTO 0);
   -- VARIABLE hilf         :  STD_LOGIC_VECTOR(31 DOWNTO 0);
   VARIABLE first        :  BOOLEAN;
   -- attribute dont_unroll : boolean;
```

```
   attribute dont_unroll of L1 : label is true;

 BEGIN -- Process
  reset_loop: loop
  --  Reset: Setze alle Variablen und Indices zurueck ----------------------
    RR <= '0';
    OA <= "0000";
    hd_r_b <= false;
    Hd_read <= '0';
    Hd_loaded <= '0';
    head_ready <= false;
    connected       <= FALSE;
    r_tag_old := "0000000000000000";
    r_tag     := "0000000000000000";
    multicast <= '0';
    xmit_bsy <= '0';
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
    exit reset_loop when (reset = '1');

  main_loop: loop    -- main loop ---------------------------
    first := TRUE;
    -- Warte bis das Header Reg. frei ist.
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and not hd_reg_bsy;

    hd_reg_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit hd_reg_loop when (hd_reg_bsy = FALSE);
    end loop hd_reg_loop;

    -- Setze Hd_read: Lies den Zellkopf ein  -----------------
    Hd_read <= '1';
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-45)(10)
    -- Warte bis ein Header da ist.
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and Hd_write = '1';

    Hd_write_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit Hd_write_loop when (Hd_write = '1');
    end loop Hd_write_loop;

    r_tag := Hd_Data(15 downto 0);
    Hd_loaded <= '1'; -- two way handshake

    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-45)
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and Hd_write = '0';

    Hd_write2_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit Hd_write2_loop when (Hd_write = '0');
    end loop Hd_write2_loop;


    Hd_loaded <= '0';
    hd_r_b <= true; -- kuendige den Header an

    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and Hd_write = '1';

    Hd_write3_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit Hd_write3_loop when (Hd_write = '1');
    end loop Hd_write3_loop;

    head_reg <= Hd_Data;
    Hd_read <= '0';
    Hd_loaded <= '1';
    head_ready <= true; -- bedeutet: der Header ist geladen
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and Hd_write = '0';

    Hd_write4_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit Hd_write4_loop when (Hd_write = '0');
    end loop Hd_write4_loop;

    Hd_loaded <= '0';
    --  Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
    --  Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and  busy = '0';

    busy_loop: loop     -- fuer monet
wait until Clk_com'event and Clk_com = '1';
exit reset_loop when (reset = '1');
exit busy_loop when (busy = '0');
    end loop busy_loop;

    IF (r_tag_old /= r_tag) THEN
      r_tag_old := r_tag;
      xmit_bsy  <= '0';  -- CC: Gib die Verbindungen frei
```

```
      connected <= FALSE;

      -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

      IF r_tag(15) = '1' THEN  multicast <= '1'; END IF;

      --   Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-45)(7)
      -- ELSE
      --   Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
      -- END IF;

      Wait UNTIL Clk_com'EVENT AND Clk_com = '1'; -- versetzt fuer monet
      exit reset_loop when (reset = '1');         -- fuer monet

      -- Die loop L1 prueft 14 bits im rtag nach requests fuer
      -- Verbindungen. (Multicast) Fuer jedes aktive Bit:
      -- Generierung einer Ausgangsadresse
      L1:  FOR i IN 13 DOWNTO  0 loop  -- Wir haben nur 14 Kanaele
        IF r_tag(i) = '1' THEN

           IF first  THEN  -- Tue's nur das erste mal

             -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and free = '1';

             free_loop: loop     -- fuer monet
        wait until Clk_com'event and Clk_com = '1';
        exit reset_loop when (reset = '1');
         exit free_loop when (free = '1');
             end loop free_loop;

             -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

             xmit_bsy <= '1';  -- Wenn die Verbindung geloest ist ..
             first := FALSE;
             -- OA <= convert(i);
             OA  <= std_logic_vector(conv_unsigned(i, 4)); -- library fct..
             RR <= '1';
           ELSE
             -- OA <= convert(i);
      OA <= std_logic_vector(conv_unsigned(i, 4)); -- library fct..
             RR <= '1';
      Wait UNTIL Clk_com'EVENT AND Clk_com = '1'; -- fuer BC(HLS47)
             exit reset_loop when (reset = '1'); -- fuer monet
           END IF;  -- IF first

    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';  -- fuer BC(HLS-4
           -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and WAIT_CC = '1';

           WAIT_CC_loop: loop     -- fuer monet
        wait until Clk_com'event and Clk_com = '1';
        exit reset_loop when (reset = '1');
         exit WAIT_CC_loop when (WAIT_CC = '1');
           end loop WAIT_CC_loop;

    -- WAIT UNTIL Clk_com'EVENT AND Clk_com = '1'; -- fuer BC (12)
           -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and WAIT_CC = '0';

           WAIT_CC2_loop: loop     -- fuer monet
        wait until Clk_com'event and Clk_com = '1';
        exit reset_loop when (reset = '1');
         exit WAIT_CC2_loop when (WAIT_CC = '0');
           end loop WAIT_CC2_loop;
           -- WAIT_CC = '0' heisst: die Verbindung steht, es geht weiter

           RR <= '0';
           Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
ELSE
       Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-47)(5)
        END IF;    -- IF r_tag(i) = '1'

       END loop;
       -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-47)(8)
       multicast <= '0';
       -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
     ELSE
       Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-47)(9)

    END IF;  -- If r_tag-old /= r_tag ..

    connected <= TRUE;
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';   -- fuer BC(HLS-45)(11)
    exit reset_loop when (reset = '1');

    --  Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and not hd_reg_bsy;

    hd_reg_bsy_loop: loop     -- fuer monet
   wait until Clk_com'event and Clk_com = '1';
   exit reset_loop when (reset = '1');
           exit hd_reg_bsy_loop when (hd_reg_bsy = FALSE);
     end loop hd_reg_bsy_loop;

     hd_r_b <= false;
     head_ready <= false;
```

```
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
    exit reset_loop when (reset = '1');
  end loop; -- main loop
  end loop; -- reset loop
 END process;


RCOUT: PROCESS
 BEGIN -- Process
   reset_loop: loop
     --  Reset: Setze alle Variablen und Indices zurueck ---------------
     transmit  <=  '0';
     load_hd <= '0';
     hd_reg_bsy <= false;
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

   main_loop: loop    -- main loop --------------------------

     -- Warte bis SRC_Out der header kommt   ----------------------
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and hd_r_b;

     hd_r_b_loop: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit hd_r_b_loop when (hd_r_b = TRUE);
     end loop hd_r_b_loop;

     hd_reg_bsy <= true;

     -- Warte bis der Header geladen ist. (head_ready)
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and head_ready;

     head_ready_loop: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit head_ready_loop when (head_ready = TRUE);
     end loop head_ready_loop;


     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and busy = '0';

     busy2_loop: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit busy2_loop when (busy = '0');
     end loop busy2_loop;

     header <= head_reg;
     load_hd <= '1';
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';  -- fuer BC(HLS-45)(1)
     -- Der header ist abgegeben, Das Register ist wieder frei

     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and load_ok = '1';

     load_ok_loop: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit load_ok_loop when (load_ok = '1');
     end loop load_ok_loop;

     hd_reg_bsy <= FALSE;
     load_hd <= '0';
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';  -- fuer BC(HLS-45)(2)
     -- Warte bis die Verbindung steht, dann kann uebertragen werden  ---

     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and connected;
     connected_loop: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit connected_loop when (connected = TRUE);
     end loop connected_loop;

     transmit <= '1';
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';  -- fuer BC(HLS-45)(3)
     -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1' and busy = '1';

     busy_loop3: loop     -- fuer monet
  wait until Clk_com'event and Clk_com = '1';
  exit reset_loop when (reset = '1');
         exit busy_loop3 when (busy = '1');
     end loop busy_loop3;

     transmit <= '0';

 END loop; -- main loop
 end loop; -- reset loop
END Process; -- Prozess rcout..

END rcar;
```

Monet-Scheduling-Report

```
Solution: rc_fastest
Project: rc
Project File:
/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/rc_mon
/rc.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/std_logic
_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs
/src/syn_arit.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs
/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}

{/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/rc_mon/src/rc_mon.vhd
 -Lwork -87}

Processes in design:
   rcin rcout

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: rcin
-----------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     13 (11 are filtered out)
      "Pseudo" operations: 159

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ----------------------------------  --------  --------   -------------
      InvGate(1)                             1.00         1            1.00
      cmp_cla(0,1,1,1,16,16)               130.98         1          130.98
      dec_rpl(1,5)                          28.00         1           28.00
      rdcor_two(3,4)                         3.00         1            3.00
      read_indexB(14,1)                      0.00         1            0.00
      read_indexB(16,0)                      0.00         1            0.00
                                                                  -------------
                                                      Total Area:    162.98

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ------------------------------
          0    LOOP                      72   reset_loop     PSEUDO
         130 ops removed w.l.
         99    OR                       179                  rdcor_two(3,4)

   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      -----------------------  -----------------------   ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   17 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
        126    ASSIGN                    67                     9    0.001
        130 ops removed w.l.
        113    RISINGEDGE               255   &21              17    0.001

   Pipelined Loops
   ---------------
                                                 Initiation
      Loop                             N_Unroll    Interval    Latency
      -------------------------------   --------   ----------   -------
```

```
   Loops
   -----
                                  Length  Start   End             Initiation
      Loop                        C-step C-step C-steps  N_Unroll   Interval   Latency
      --------------------------  ------ ------ -------  --------  ----------  -------
      main_loop                     16      2      17
        hd_reg_loop                  1      2       2
        hd_write_loop                1      3       3
        hd_write2_loop               1      4       4
        hd_write3_loop               1      5       5
        hd_write4_loop               1      6       6
        busy_loop                    1      7       7
        l1                           5      9      13
          free_loop                  1      9       9
          wait_cc_loop               1     10      10
          wait_cc2_loop              1     11      11
        hd_reg_bsy_loop              1     15      15


Process: rcout
------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     13 (11 are filtered out)
      "Pseudo" operations: 159

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      InvGate(1)                           1.00        1            1.00
                                                              -------------
                                                 Total Area:      1.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ----------------------------
          0    LOOP                     264   reset_loop     PSEUDO
          37 ops removed w.l.
          9    LOOP                     287   head_ready     PSEUDO

   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      -----------------------  -----------------------   ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH    6 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
          0    LOOP                     264   reset_loop        1    0.000
         37 ops removed w.l.
         31    IO_W=>transmit           337                     6    0.001

   Pipelined Loops
   ---------------
                                              Initiation
      Loop                          N_Unroll    Interval    Latency
      ----------------------------   --------   ----------   -------

   Loops
   -----
                                  Length  Start   End             Initiation
      Loop                        C-step C-step C-steps  N_Unroll   Interval   Latency
      --------------------------  ------ ------ -------  --------  ----------  -------
      main_loop                      6      1       6
        hd_r_b_loop                  1      1       1
        head_ready_loop              1      2       2
        busy2_loop                   1      3       3
        load_ok_loop                 1      4       4
        connected_loop               1      5       5
        busy_loop3                   1      6       6
```

```
Area Estimate
-------------
   Component                                  Post   Post   Post
   Name                              Area     Alloc  Bind   Share
   ---------                      ----------  -----  -----  -----
   InvGate(1)                          1.000      2     20     20
   cmp_cla(0,1,1,1,16,16)            130.980      1      1      1
   dec_rpl(1,5)                       28.000      1      1      1
   rdcor_two(3,4)                      3.000      1      0      0
   read_indexB(14,1)                   0.000      1      0      0
   read_indexB(16,0)                   0.000      1      0      0
   AndGate(1,2)                        2.000      0     10     10
   AndGate(1,3)                        2.000      0      5      5
   Mlatch(1)                           5.000      0      3      3
   Mux1h(1,2)                          3.000      0      8      8
   Mux1h(1,3)                          4.000      0      5      5
   Mux1h(16,2)                        48.000      0      1      1
   Mux1h(16,3)                        64.000      0      1      1
   Mux1h(32,2)                        96.000      0      2      2
   Mux1h(4,2)                         12.000      0      1      1
   Mux1h(4,3)                         16.000      0      1      1
   NandGate(1,2)                       1.000      0      3      3
   NorGate(1,19)                      12.875      0      1      1
   NorGate(1,2)                        1.000      0      7      7
   NorGate(1,20)                      13.500      0      5      5
   NorGate(1,3)                        2.000      0      1      1
   OrGate(1,2)                         2.000      0     45     45
   OrGate(1,3)                         2.000      0      1      1
   Register(1,0,0)                     7.000      0     13     13
   Register(16,0,0)                  112.000      0      2      2
   Register(32,0,0)                  224.000      0      2      2
   Register(4,0,0)                    28.000      0      2      2
   mux_aoi(2,1)                        4.000      0      4      4
   mux_aoi(2,4)                       12.683      0      1      1
   mux_mux4(14,1)                     27.178      0      1      1
   rdcand_three(1,4)                   3.000      0      2      2
   rdcnor_four(4,7)                    4.742      0      1      1
   rdcnor_four(4,9)                    6.301      0      3      3

      TOTAL AREA (After Sharing):               1688.860

Area Estimate
-------------
                Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area         164.0             164.0             1688.9            1688.9
     Total Reg          0.0               0.0              819.0  (48%)      819.0  (48%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath         164.0 (100%)      164.0 (100%)      1688.9 (100%)     1688.9 (100%)
       MUX              0.0               0.0               431.9  (26%)      431.9  (26%)
       FUNC           162.0  (99%)      162.0  (99%)       203.6  (12%)      203.6  (12%)
       LOGIC            2.0   (1%)        2.0   (1%)       234.4  (14%)      234.4  (14%)
       BUFFER           0.0               0.0                 0.0               0.0
       DP-REG           0.0               0.0               819.0  (48%)      819.0  (48%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM                0.0               0.0                 0.0               0.0
       FSM-REG          0.0               0.0                 0.0               0.0
       FSM-COMB         0.0               0.0                 0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Critical Path
     Max Delay 6.94
     Slack 18.06
     Resource: r498rg(OrGate_1_2) Signal: n_67tmp Unit delay: 0.33 Delay: 0.33
     Resource: r501rg(OrGate_1_2) Signal: n_68tmp Unit delay: 0.33 Delay: 0.66
     Resource: r504rg(OrGate_1_2) Signal: n_69tmp Unit delay: 0.33 Delay: 0.99
     Resource: r507rg(OrGate_1_2) Signal: n_70tmp Unit delay: 0.33 Delay: 1.32
     Resource: r510rg(OrGate_1_2) Signal: n_71tmp Unit delay: 0.33 Delay: 1.65
     Resource: r513rg(OrGate_1_2) Signal: n_72tmp Unit delay: 0.33 Delay: 1.98
     Resource: r516rg(OrGate_1_2) Signal: n_73tmp Unit delay: 0.33 Delay: 2.31
     Resource: r519rg(OrGate_1_2) Signal: n_74tmp Unit delay: 0.33 Delay: 2.64
     Resource: r522rg(OrGate_1_2) Signal: n_75tmp Unit delay: 0.33 Delay: 2.97
     Resource: r525rg(OrGate_1_2) Signal: n_76tmp Unit delay: 0.33 Delay: 3.3
     Resource: r528rg(OrGate_1_2) Signal: n_77tmp Unit delay: 0.33 Delay: 3.63
     Resource: r531rg(OrGate_1_2) Signal: n_78tmp Unit delay: 0.33 Delay: 3.96
     Resource: r534rg(OrGate_1_2) Signal: n_79tmp Unit delay: 0.33 Delay: 4.29
     Resource: r537rg(OrGate_1_2) Signal: n_80tmp Unit delay: 0.33 Delay: 4.62
     Resource: r540rg(OrGate_1_2) Signal: n_81tmp Unit delay: 0.33 Delay: 4.95
     Resource: r543rg(OrGate_1_2) Signal: n_82tmp Unit delay: 0.33 Delay: 5.28
     Resource: r546rg(OrGate_1_2) Signal: n_83tmp Unit delay: 0.33 Delay: 5.61
     Resource: r596rg(OrGate_1_2) Signal: n_101tmp Unit delay: 0.33 Delay: 5.94
     Resource: r599rg(OrGate_1_2) Signal: n_102tmp Unit delay: 0.33 Delay: 6.27
     Resource: r654rg(NorGate_1_2) Signal: n_121tmp Unit delay: 0.18 Delay: 6.45
     Resource: r243rg(Mux1h_1_3) Signal: hd_loaded_pre_ff Unit delay: 0.49 Delay: 6.94

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: 25
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: 25
       Resource r136rg Port hd_read_pre_ff 18.71
       Resource r151rg Port head_ready_pre_ff 18.71
       Resource r160rg Port r_tag_pre_ff 24.5
```

```
      Resource r167rg Port r_tag_old_pre_ff 18.72
      Resource r180rg Port n_4tmp_19_pre_ff 20.7522
      Resource r189rg Port multicast_pre_ff 18.39
      Resource r198rg Port rr_pre_ff 18.39
      Resource r204rg Port oa_pre_ff 18.72
      Resource r208rg Port n_47dp 23.0554
      Resource r212rg Port first 24.26
      Resource r216rg Port n_4tmp_19 24.28
      Resource r222rg Port head_reg_pre_ff 24.5
      Resource r235rg Port hd_r_b_pre_ff 18.71
      Resource r244rg Port hd_loaded_pre_ff 18.06
      Resource r254rg Port first_pre_ff 23.95
      Resource r267rg Port l1__i_pre_ff 20.4822
      Resource r282rg Port xmit_bsy_pre_ff 19.04
      Resource r291rg Port connected_pre_ff 18.39
      Resource r455rg Port load_hd_pre_ff 22.34
      Resource r464rg Port header_pre_ff 24.5
      Resource r477rg Port hd_reg_bsy_pre_ff 22.34
      Resource r492rg Port transmit_pre_ff 22.34

   FSM name: rcin (rc_sid0_rcin)
     Total area: 3590
       Combinational area: 3060 Sequential area: 530

     Input to output delays

     Input to register-input delays
       n_302_fsm: 0
       n_303_fsm: 0
       n_304_fsm: 0
       n_305_fsm: 0
       n_306_fsm: 0
       n_47dp: 0.35
       hd_write: 0.44
       n_307_fsm: 0
       n_308_fsm: 0
       n_309_fsm: 0
       n_310_fsm: 0
       n_307_fsm: 0
       hd_reg_bsy: 0.44
       l1__i: 1.09
       busy: 0.44
       wait_cc: 0.44
       free: 0.35
       reset: 0.35

     Register-output to output delays
       clk_com->n_36_fsm: 0
       37 delays removed w.l.  (all 0)
       clk_com->n_34_fsm: 0


   FSM name: rcout (rc_sid0_rcout)
     Total area: 517
       Combinational area: 350 Sequential area: 167

     Input to output delays

     Input to register-input delays
       head_ready: 0.35
       load_ok: 0.35
       hd_r_b: 0.35
       busy: 0.44
       connected: 0.35
       reset: 0.35

     Register-output to output delays
       clk_com->n_335_fsm: 0
       clk_com->n_314_fsm: 0
       clk_com->n_316_fsm: 0
       clk_com->n_318_fsm: 0
       clk_com->n_320_fsm: 0
       clk_com->n_322_fsm: 0
       clk_com->n_324_fsm: 0
       clk_com->n_326_fsm: 0
       clk_com->n_328_fsm: 0
       clk_com->n_347_fsm: 0
       clk_com->n_349_fsm: 0
       clk_com->n_312_fsm: 0


   ----------------------------------------------------------------

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r455rg           1 bits    load_hd
```

```
   r204rg           4 bits    oa
   r222rg          32 bits    head_reg
   r492rg           1 bits    transmit
   r160rg          16 bits    r_tag
   r180rg           1 bits    n_4tmp_19  n_6tmp_20
   r477rg           1 bits    hd_reg_bsy
   r244rg           1 bits    hd_loaded
   r198rg           1 bits    rr
   r208rg           0 bits    n_11tmp
   r282rg           1 bits    xmit_bsy
   r212rg           0 bits    n_12tmp
   r464rg          32 bits    header
   r167rg          16 bits    r_tag_old
   r267rg           4 bits    l1__i
   r151rg           1 bits    head_ready
   r216rg           0 bits    n_13tmp
   r235rg           1 bits    hd_r_b
   r136rg           1 bits    hd_read
   r189rg           1 bits    multicast
   r254rg           1 bits    first
   r291rg           1 bits    connected

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                 PADZEROES            LOOPEND
          IO_R                 SIGNEXTEND           SELECT
          READSLICE            NOP                  SELECTEND
          WRITESLICE           END                  CONDITIONALTRANSFER
          ASSIGN               ITERATE              RISINGEDGE
          ASSIGNDF             TERMINATE            FALLINGEDGE
          CONCATENATE          LOOP
   Allocation Filter:
          IO_R         WRITEINDEX   OR          XNOR         SKEDLIST
          IO_W         AND          NOR         NOT
          READINDEX    NAND         XOR         MUX
   Schedule Filter (default):
          PADZEROES            LOOPEND              RISINGEDGE
          SIGNEXTEND           SELECT               FALLINGEDGE
          END                  SELECTEND
          LOOP                 CONDITIONALTRANSFER

End of Report
```

12.4 Schieberegister-Steuerung (SRC)

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des SRC fuer
-- Simulation mit Mentor qhsim
-- Synthese mit Monet
-- src_mon.vhd hat einen Prozess
-- Umgeschrieben fuer Monet von Synopsys BC-Version
-- Prozeduren shift und encode wieder inline eingearbeitet. - Fuer BC
-- Reset eingefuehrt
-- Das Schieberegister wird ausgelagert, da BC einen
-- eine Taktzyclus hinzufuegt.                              - Fuer BC
-- Letzte Version von srcout.vhd mit 4B/5B encoder
-- Am Anfang vor jeder Uebertragung einer Header-Zelle
-- steht das Header-Startzeichen: "11001" (X19)
-- Am Anfang vor jeder Uebertragung eines 32 - bit - Reg
-- steht das Payload-Register-Startzeichen: "10001" (X11)
-- Autor: W. Lange 1.9.98
-- dont_unroll fuer loop L1 eingefuehrt

Library IEEE;
USE IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all;

ENTITY srcout IS
  PORT (Clk_xmit    : IN STD_LOGIC;
        reset       : IN  STD_LOGIC;             -- Fuer BC und Monet
        busy        : OUT STD_LOGIC;             -- von und zu SRC_IN
        transmit    : IN  STD_LOGIC;
        load_hd     : IN  STD_LOGIC;
        header      : IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
        load_ok     : OUT STD_LOGIC;
        Cell_read   : OUT STD_LOGIC;            -- Cell_FIFO
        Cell_write  : IN  STD_LOGIC;
        Cell_word   : IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
        Cell_loaded : OUT  STD_LOGIC;
--      Cell_data   : OUT  STD_LOGIC   -- Datenleitung entfernt
        sr_ready    : IN   STD_LOGIC;  -- neu fuer SR
        sr_strobe   : OUT  STD_LOGIC;  -- neu fuer SR - Date
        shiftreg    : OUT  STD_LOGIC_VECTOR(47 DOWNTO 0)); -- neuer Ausgang
END srcout;
-------------------------------------------------------------------------
Use work.utils_src.all;
```

```
ARCHITECTURE srcoutar OF srcout IS

  ATTRIBUTE sync_reset OF srcoutar : ARCHITECTURE IS "reset" ;
  ATTRIBUTE reset_phase OF Shift_Proc : LABEL IS "positive" ;

BEGIN

  Shift_Proc: PROCESS

  VARIABLE shift_reg : STD_LOGIC_VECTOR(47 DOWNTO 0);
  VARIABLE aux_reg : STD_LOGIC_VECTOR(31 DOWNTO 0);
  attribute dont_unroll of L1 : label is true;

-- Bitlokationen des shift_reg
--              47  43 42 38 37 33 32 28 27 23 22 18 17 13 12  8 7   3 2 0
-- shift_reg := "11001 00000 00000 00000 00000 00000 00000 00000 00000 000";

   PROCEDURE load_reg_encode IS
   BEGIN

        shift_reg(42 downto 38) := encd(aux_reg(31 downto 28));
  shift_reg(37 downto 33) := encd(aux_reg(27 downto 24));
        shift_reg(32 downto 28) := encd(aux_reg(23 downto 20));
        shift_reg(27 downto 23) := encd(aux_reg(19 downto 16));
        shift_reg(22 downto 18) := encd(aux_reg(15 downto 12));
        shift_reg(17 downto 13) := encd(aux_reg(11 downto 8));
shift_reg(12 downto 8) := encd(aux_reg(7 downto 4));
  shift_reg(7 downto 3) := encd(aux_reg(3 downto 0));

   END;


  Begin  -- Process
   -- reset_loop: loop
   --  Reset: Setze alle Variablen und Indices zurueck ----------------------
      busy <= '0';
      load_ok <= '0';
      Cell_loaded  <=  '0';
      Cell_read  <=  '0';
      sr_strobe  <= '0';
      shift_reg := "00000000000000000000000000000000000000000000000000";
      shiftreg  <= "00000000000000000000000000000000000000000000000000";
      aux_reg :=   "00000000000000000000000000000000";
      Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

    main_loop: loop
     -- Load Header ----------------------------------------------
     -- Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1' and load_hd = '1';

    load_hd_loop: loop     -- fuer monet
wait until Clk_xmit'event and Clk_xmit = '1';
exit load_hd_loop when (load_hd = '1');
    end loop load_hd_loop;

    --  Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

       aux_reg := header;
       load_ok <= '1';
       busy <= '1';

     Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1'; -- f. BC(HLS-45)

     -- Lade Schieberegister und encodiere den Header
     --  load_reg_encode; encode wandelt 4 Bit in 5 Bit um

     load_reg_encode;

     Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

     load_hd2_loop: loop     -- fuer monet
wait until Clk_xmit'event and Clk_xmit = '1';
exit load_hd2_loop when (load_hd = '0');
     end loop load_hd2_loop;

     load_ok <= '0';

     -- Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1'; -- f. BC(HLS-45)(1)

     -- Warte auf transmit und sr_ready   ----------------------

     --  Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1' and transmit = '1';
     transmit_loop: loop     -- fuer monet
wait until Clk_xmit'event and Clk_xmit = '1';
exit transmit_loop when (transmit = '1' and sr_ready = '1');
     end loop transmit_loop;

     -- wenn das und-Verknuepfen hier  geht.. ist das ein grosser Vorteil..

     --  Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1' and sr_ready = '1';

     shift_reg(47 downto 43) := "11001";   -- lade das "Start_Cell" Zeichen

     -- shift;-- call Procedure shift - Schiebe den Header durch den Switch
```

```
     -- entfernt

     sr_strobe <= '1';
     shiftreg  <= shift_reg;     -- lade das Schieberegister

     -- Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

     -- Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1' and sr_ready ='0';

     sr_ready_loop: loop     -- fuer monet
wait until Clk_xmit'event and Clk_xmit = '1';
exit sr_ready_loop when (sr_ready = '0');
     end loop sr_ready_loop;

     sr_strobe <= '0';

     Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

      -- Schiebe die Payload durch den Switch    ---------------------
  L1: FOR k IN 1 TO 12 loop
Cell_read <= '1';

        Cell_write_loop: loop     -- fuer monet
  wait until Clk_xmit'event and Clk_xmit = '1';
  exit Cell_write_loop when (Cell_write = '1');
        end loop Cell_write_loop;

        -- Lade Schieberegister und encodiere die Daten (load_reg_encode;)

        aux_reg := Cell_word;

        Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

        load_reg_encode;

        Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

        Cell_loaded <= '1';
        Cell_read   <= '0';

        Cell_write2_loop: loop     -- fuer monet
  wait until Clk_xmit'event and Clk_xmit = '1';
  exit Cell_write2_loop when (Cell_write = '0');
        end loop Cell_write2_loop;

Cell_loaded <= '0';

        sr_ready2_loop: loop     -- fuer monet
  wait until Clk_xmit'event and Clk_xmit = '1';
  exit sr_ready2_loop when (sr_ready = '1');
        end loop sr_ready2_loop;

       shift_reg(47 downto 43) := "10001";   -- lade das "Start_Reg" Zeichen

       -- Schiebe ein Payload-Wort durch den Switch (call proc shift) -----
       -- entfernt

       sr_strobe <= '1';
       shiftreg  <= shift_reg;     -- lade das Schieberegister

       sr_ready3_loop: loop     -- fuer monet
  wait until Clk_xmit'event and Clk_xmit = '1';
  exit sr_ready3_loop when (sr_ready = '0');
       end loop sr_ready3_loop;

       sr_strobe <= '0';

       Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

     END loop;  -- For i = 1 to 12

     busy <= '0';
     Wait UNTIL Clk_xmit'EVENT AND Clk_xmit = '1';

    END loop;    -- main_loop
    -- end loop;     -- reset loop
   END Process;

  END srcoutar;
```

Quellcode des SRC-Package

```
-------------------------------------------------
Library IEEE;
USE IEEE.std_logic_1164.all;
    use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all;

package utils_src is
```

```
  function encd(v: STD_LOGIC_VECTOR(3 downto 0)) return STD_LOGIC_VECTOR;

end;

--------------------------------------------------

package body utils_src is

function encd(v : STD_LOGIC_VECTOR(3 downto 0)) return STD_LOGIC_VECTOR IS
  VARIABLE temp   : STD_LOGIC_VECTOR(4 downto 0);
  VARIABLE vector : STD_LOGIC_VECTOR(3 downto 0);

  begin
    vector := v;

    CASE vector IS
WHEN  "0000"  => temp := "11110";
WHEN  "0001"  => temp := "01001";
        WHEN  "0010"  => temp := "10100";
WHEN  "0011"  => temp := "10101";
WHEN  "0100"  => temp := "01010";
        WHEN  "0101"  => temp := "01011";
        WHEN  "0110"  => temp := "01110";
WHEN  "0111"  => temp := "01111";
WHEN  "1000"  => temp := "10010";
WHEN  "1001"  => temp := "10011";
WHEN  "1010"  => temp := "10110";
WHEN  "1011"  => temp := "10111";
WHEN  "1100"  => temp := "11010";
WHEN  "1101"  => temp := "11011";
WHEN  "1110"  => temp := "11100";
WHEN  "1111"  => temp := "11101";
WHEN OTHERS => temp := "00000"; -- Probleme mit v_parser
     END CASE;

   return temp;
  end encd;
end utils_src;
--------------------------------------------------
```

Monet-Scheduling-Report

```
Solution: Solution_1
Project: src
Project File: /afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht
/src_mon/src.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src
/std_logic_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet
/pkgs/qhdl_libs/src/syn_arit.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor
/MONET/monet/pkgs/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}
 {/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/src_mon/src
/utils_src.vhd -Lwork -87} {/afs/informatik.uni-tuebingen.de/home
/wlange/hilan/aht/src_mon/src/src_mon.vhd -Lwork -87}

Processes in design:
   shift_proc

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: shift_proc
--------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     7 (5 are filtered out)
      "Pseudo" operations: 703

   Clock Information
   -----------------
      Clock Signal: clk_xmit (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      AndGate(1,2)                          2.00        1            2.00
      InvGate(1)                            1.00        1            1.00
      cmp_cla(0,1,1,1,4,4)                 38.00        1           38.00
```

```
   inc_vn(0,4)                          18.00        1          18.00
                                                                -------------
                                                    Total Area:     59.00

Multi-Cycle (Combinational) Component Usage
-------------------------------------------
   Instance    Component Name                Delay    #Cycles
   --------    ----------------------------  -------  -------

Operation-to-Component Mappings
-------------------------------
   Oper     Operation             Line                    Mapped to
   Number   Type                  Number   Label          Component
   ------   --------------------  ------   -----------    ----------------------------
       0    CONDITIONALTRANSFER       37                  PSEUDO
      708 ops removed w.l.
       99    SELECT                    25                 PSEUDO

Cycle Constraints
-----------------
   From                     To                       Constraint   Actual  Met?
   -----------------------  -----------------------  ----------   ------  ----

Scheduled Operations
--------------------
   TOTAL SCHEDULE LENGTH   17 C-Steps

   Oper     Operation             Line
   Number   Type                  Number   Label          C-Step  Delay
   ------   --------------------  ------   -----------    ------  -----
     394    READSLICE                 18                    4    0.001
     702 operations removed w.l.
     282    RISINGEDGE               225   &15             17    0.001

Pipelined Loops
---------------
                                              Initiation
   Loop                            N_Unroll    Interval    Latency
   -----------------------------   --------   ----------   -------

Loops
-----
                                  Length  Start   End              Initiation
   Loop                           C-step C-step C-steps  N_Unroll   Interval   Latency
   -----------------------------  ------ ------ -------  --------  ----------  -------
   main_loop                        16      2      17
     load_hd_loop                    1      2       2
     load_hd2_loop                   1      5       5
     transmit_loop                   1      6       6
     sr_ready_loop                   1      7       7
     l1                              7      9      15
       cell_write_loop               1      9       9
       cell_write2_loop              1     11      11
       sr_ready2_loop                1     12      12
       sr_ready3_loop                1     13      13

Area Estimate
-------------
   Component                                    Post   Post   Post
   Name                              Area      Alloc   Bind  Share
   ---------                      ----------    -----  -----  -----
   AndGate(1,2)                        2.000        1      3      3
   InvGate(1)                          1.000        1     11     11
   cmp_cla(0,1,1,1,4,4)               38.000        1    113    113
   inc_vn(0,4)                        18.000        1      1      1
   AndGate(48,2)                      96.000        0      2      2
   InvGate(48)                        48.000        0      2      2
   Mux1h(1,2)                          3.000        0     43     43
   Mux1h(1,3)                          4.000        0      2      2
   Mux1h(4,3)                         16.000        0      1      1
   Mux1h(48,3)                       192.000        0      1      1
   Mux1h(48,5)                       336.000        0      1      1
   NorGate(1,18)                      12.250        0      4      4
   NorGate(1,2)                        1.000        0      3      3
   NorGate(1,20)                      13.500        0      1      1
   OrGate(1,2)                         2.000        0    487    487
   Register(1,0,0)                     7.000        0      5      5
   Register(4,0,0)                    28.000        0      1      1
   Register(48,0,0)                  336.000        0      2      2
   rdcand_three(1,4)                   3.000        0      8      8
   rdcor_three(3,4)                    3.000        0      8      8

      TOTAL AREA (After Sharing):               7120.500

Area Estimate
-------------
                Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area         59.0              59.0             7120.5            7120.5
     Total Reg         0.0               0.0              735.0  (10%)      735.0  (10%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath         59.0 (100%)       59.0 (100%)     7120.5 (100%)     7120.5 (100%)
       MUX             0.0               0.0              681.0  (10%)      681.0  (10%)
```

```
      FUNC           56.0  (95%)       56.0  (95%)     4360.0  (61%)     4360.0  (61%)
      LOGIC           3.0   (5%)        3.0   (5%)     1344.5  (19%)     1344.5  (19%)
      BUFFER          0.0                0.0               0.0               0.0
      DP-REG          0.0                0.0             735.0  (10%)      735.0  (10%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM              0.0                0.0               0.0               0.0
       FSM-REG        0.0                0.0               0.0               0.0
       FSM-COMB       0.0                0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Critical Path
     Max Delay 7.04
     Slack 17.96
     Resource: r1220rg(cmp_cla_0_1_1_1_4_4) Signal: n_343dp Unit delay: 2.54 Delay: 2.54
     Resource: r2453rg(OrGate_1_2) Signal: n_682tmp Unit delay: 0.33 Delay: 2.87
     Resource: r2456rg(OrGate_1_2) Signal: n_683tmp Unit delay: 0.33 Delay: 3.2
     Resource: r2459rg(OrGate_1_2) Signal: n_684tmp Unit delay: 0.33 Delay: 3.53
     Resource: r2462rg(OrGate_1_2) Signal: n_685tmp Unit delay: 0.33 Delay: 3.86
     Resource: r2465rg(OrGate_1_2) Signal: n_686tmp Unit delay: 0.33 Delay: 4.19
     Resource: r2468rg(OrGate_1_2) Signal: n_687tmp Unit delay: 0.33 Delay: 4.52
     Resource: r2471rg(OrGate_1_2) Signal: n_688tmp Unit delay: 0.33 Delay: 4.85
     Resource: r2474rg(OrGate_1_2) Signal: n_689tmp Unit delay: 0.33 Delay: 5.18
     Resource: r2477rg(OrGate_1_2) Signal: n_690tmp Unit delay: 0.33 Delay: 5.51
     Resource: r2480rg(OrGate_1_2) Signal: n_691tmp Unit delay: 0.33 Delay: 5.84
     Resource: r1574rg(Mux1h_1_2) Signal: n_346dp Unit delay: 0.41 Delay: 6.25
     Resource: r1048rg(Mux1h_48_5) Signal: n_1043tmp Unit delay: 0.53 Delay: 6.78
     Resource: r1055rg(AndGate_48_2) Signal: shift_reg_pre_ff Unit delay: 0.26 Delay: 7.04

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: 25
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: 25
       Resource r1024rg Port load_ok_pre_ff 19.37
       Resource r1033rg Port l1__k_pre_ff 23.52
       Resource r1042rg Port sr_strobe_pre_ff 18.72
       Resource r1056rg Port shift_reg_pre_ff 17.96
       Resource r1071rg Port cell_loaded_pre_ff 19.37
       Resource r1086rg Port busy_pre_ff 19.37
       Resource r1103rg Port shiftreg_pre_ff 19.7
       Resource r1112rg Port cell_read_pre_ff 19.05

   FSM name: shift_proc (srcout_sid0_shift_proc)
     Total area: 2093
       Combinational area: 1694 Sequential area: 399

     Input to output delays

     Input to register-input delays
       n_409dp: 0.26
       n_409dp: 0.35
       transmit: 0.61
       load_hd: 0.44
       sr_ready: 0.61
       cell_write: 0.44
       reset: 0.35

     Register-output to output delays
       clk_xmit->n_155_fsm: 0
       27 delays removed (all 0) w.l.
       clk_xmit->n_115_fsm: 0


   ----------------------------------------------------------------

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Finite State Machines
---------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r1112rg          1 bits    cell_read
   r1103rg         48 bits    shiftreg
   r1071rg          1 bits    cell_loaded
   r1086rg          1 bits    busy
   r1042rg          1 bits    sr_strobe
   r1033rg          4 bits    l1__k
   r1024rg          1 bits    load_ok
   r1056rg         48 bits    shift_reg

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                 PADZEROES            LOOPEND
          IO_R                 SIGNEXTEND           SELECT
          READSLICE            NOP                  SELECTEND
          WRITESLICE           END                  CONDITIONALTRANSFER
          ASSIGN               ITERATE              RISINGEDGE
```

```
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R          WRITEINDEX    OR            XNOR          SKEDLIST
          IO_W          AND           NOR           NOT
          READINDEX     NAND          XOR           MUX
   Schedule Filter (default):
          PADZEROES             LOOPEND               RISINGEDGE
          SIGNEXTEND            SELECT                FALLINGEDGE
          END                   SELECTEND
          LOOP                  CONDITIONALTRANSFER

End of Report
```

12.5 Verbindungs-Steuerung (CC)

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des CC mit Multicast fuer
-- Simulation mit Synopsys. Compilierung mit v_parser,
-- cc.vhd hat Interfaces zu 14 mal AHT_IN, und den Switch,
-- CC nimmt die Routing Requests der RC's auf,
-- und veranlasst den Switch, eine Verbindung zwischen
-- Eingangs - und Ausgangsadresse zu schalten (load und configure)
-- Bis die Verbindung geschaltet ist, wird das Signal
-- Wait_CC aktiviert.  Es wird angenommen, dass jedes neue load
-- und configure die alten Verbindungen ueberladedt, sodass ein
-- reset (zum Passthru) zwischen den einzelnen Umschaltungen unnoetig ist.
-- Broadcast kann alle anderen RR's blockieren. Variable "first" sorgt dafuer,
-- dass nach einem broadcast erst einmal die anderen RR's an die Reihe kommen.
-- Autor: W. Lange, 4. 11. 97
-- Umgeschrieben fuer Monet: 20. Juli 1998:
-- 1. Monet Libraries zufuegen.
-- Reset zugefuegt: Rst
-- Reset loop
-- Reset zu jedem Wait..
-- convert, convert_to_nat mit Library-Funktionen (auch std_logic_unsigned)
-- gibt Probleme, daher convert ins Package kopiert..
-- Die loops sollen nicht aufgerollt werden: (resultiert in zu vielen
-- csteps (110): (attribute dont_unroll of L1 : label is true;)
-- Globales reset eingefuehrt


Library IEEE;
USE IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
-- use IEEE.std_logic_unsigned.all;
Use work.utilsCC.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY cc IS
  PORT (
    rst        : IN STD_LOGIC;
    Clk_com    : IN STD_LOGIC;
    RR         : IN STD_LOGIC_VECTOR(0 to 13); -- Routing Request
    multicast  : IN STD_LOGIC_VECTOR(0 to 13);
    OA         : IN Address;
    xmit_bsy   : IN STD_LOGIC_VECTOR(0 to 13);
    Wait_CC    : OUT STD_LOGIC_VECTOR(0 to 13);
    free       : OUT  STD_LOGIC_VECTOR(0 to 13);
    S_IA       : OUT STD_LOGIC_VECTOR(3 downto 0);      -- zum switch
    S_OA       : OUT STD_LOGIC_VECTOR(3 downto 0);
    load       : OUT STD_LOGIC;
    configure  : OUT STD_LOGIC;
    reset      : OUT STD_LOGIC);
  END cc;

ARCHITECTURE ccar OF cc IS

-- Attribut fuer globales reset:
  ATTRIBUTE sync_reset OF  ccar : ARCHITECTURE IS "rst" ;
  ATTRIBUTE reset_phase OF ccin : LABEL IS "positive" ;


 BEGIN
 ccin: PROCESS -- Der CCIN Process nimmt Routing Requests auf
   VARIABLE active_reg,belegt_reg  :  STD_LOGIC_VECTOR(0 TO 13);
   VARIABLE verbind                :  matrix;
   VARIABLE IAd,r,k,OAd      :  NATURAL := 0;
--   VARIABLE OAD_i           : INTEGER;
   VARIABLE inp_addr, out_addr     :  STD_LOGIC_VECTOR(3 DOWNTO 0);
   VARIABLE vector_14    :  STD_LOGIC_VECTOR(0 to 13) := "00000000000000";
   CONSTANT zeros        :  STD_LOGIC_VECTOR(0 to 13) := "00000000000000";
   VARIABLE first        :  BOOLEAN;
   attribute dont_unroll of L1 : label is true;
   attribute dont_unroll of L2 : label is true;
```

```
 BEGIN -- Process
    --  Reset: Setze alle Signale und Variablen zurueck ----------------------
--  reset_loop: loop
    belegt_reg  := "00000000000000"; -- Zeigt belegte Ausgaenge
    active_reg  := "00000000000000"; -- zeigt aktive Eingaenge
    inp_addr    := "0000";
    out_addr    := "0000";
    first       := TRUE;
    Wait_CC    <=  "00000000000000";  -- 14 Waits auf '0'
    free       <=  "11111111111111";  -- 14 free auf '1'
    S_IA       <=  "0000";
    S_OA       <=  "0000";
    load       <= '0';
    configure  <= '1';
    reset      <= '1';
    -- Set Switch auf Passthru

    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

    -- Warte einen Taktzyklus (mind. 7 ns)
    configure  <= '0';
    reset      <= '0';
    -- Setze Verbindungsmatrix gesamt auf '0'
    For i in 0 to 13 loop  verbind(i) := "00000000000000";  end loop;
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

    main_loop: loop    -- main loop --------------------------
    -- Pruefe ob  ein RR da ist.
    -- Broadcast Behandlung: haelt alle anderen Uebertragungen an ...
     IF RR(0) = '1' and  first THEN
       Wait_CC(0) <= '1';
       -- Warte bis alle Kanaele frei sind
       IF  (belegt_reg  = zeros)  THEN -- alle Ausgaenge frei?
          belegt_reg :=  "11111111111111";
          free       <=  "00000000000000";
          -- Setze den Switch auf Broadcast:
          reset <= '1';
          configure <= '0';
          active_reg(0)  := '1';
          Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

          -- Warte einen Takt lang (mind. aber 7 ns)
          reset <= '0';
          Wait_CC(0) <= '0';
          Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

          first := FALSE; -- sorge dafuer, dass auch andere Kanaele drankommen
       END IF;

    ELSE
    -- Suche die Kanaele ab nach einem aktiven RR
    L1: FOR i in 1 to 13 loop    -- 13 Kanaele-loop
      IF multicast(i) = '1' THEN
        r := 0;
       WHILE multicast(i) = '1' loop  -- multicast loop
         r := r + 1;
         IF RR(i) = '1'  THEN
--           connect_sw(k);    -- der Compiler kennt i nicht
                               -- Die Prozedur muss inline gesetzt werden
         Wait_CC(i) <= '1';            -- Setze Wait aktiv --
         out_addr    := OA(i);         -- conv to std_logic_vector

--         OAd         := conv_to_nat(out_addr);
         -- Konvertierung mit Standard-Funktion (siehe notes)
  OAd  := natural(CONV_INTEGER(out_addr));
--       inp_addr := convert(p);
         inp_addr := std_logic_vector(conv_unsigned(i, 4)); -- library fct..
      IF belegt_reg(OAd) = '0'  THEN  -- Der Ausgang ist frei
          S_IA      <= inp_addr;
         S_OA      <= out_addr;
         load      <= '1';
          IF multicast(i) = '0' THEN configure <= '1'; END IF; --ein config.
           vector_14 := zeros;
          vector_14(OAd) := '1';
          verbind(i) :=  verbind(i) or vector_14;
          belegt_reg(OAd) := '1';
          free(i)    <= '0';

          Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
           --Warte einen Takt mind. 7 ns

          load       <= '0';
          configure  <= '0';
           Wait_CC(i) <= '0';
          active_reg(i) := '1';      -- nach Wait_CC = '0'

          Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

      END IF; -- End.. Der Ausgang ist frei

          END IF;   --  aktive RR's sind da
          EXIT when r = 12; -- 13 waere broadcast
          Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
```

```
        END loop;  -- End while loop
        configure <= '1';
        Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

        configure <= '0';
        Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

      ELSE
        IF (RR(i) = '1' and active_reg(i) = '0') THEN
--           connect_sw(k);

        Wait_CC(i) <= '1';            -- Setze Wait aktiv --
        out_addr    := OA(i);         -- conv to std_logic_vector

        OAd  := natural(CONV_INTEGER(out_addr));
--      OAd         := conv_to_nat(out_addr);
--   inp_addr := convert(i);
        inp_addr := std_logic_vector(conv_unsigned(i, 4));
        IF belegt_reg(OAd) = '0'  THEN  -- Der Ausgang ist frei
           S_IA      <= inp_addr;
           S_OA      <= out_addr;
           load      <= '1';
           IF multicast(i) = '0' THEN configure <= '1'; END IF; --ein config.
           vector_14 := zeros;
           vector_14(OAd) := '1';
           verbind(i) :=  verbind(i) or vector_14;
           belegt_reg(OAd) := '1';
           free(i)    <= '0';

           Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
             -- Warte einen Takt mind. 7 ns

           load       <= '0';
           configure  <= '0';
           Wait_CC(i) <= '0';
           active_reg(i) := '1';      -- nach Wait_CC = '0'

           Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

        END IF; -- End.. Der Ausgang ist frei

       END IF;   -- Ein aktiver RR ist da
      END IF;    -- multicast IF Then Else

      Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

     end loop;  -- 13 Kanaele loop: Abfrage nach aktivem RR
     first := TRUE;
    END IF;    -- broadcast IF

    -- Bei Uebertragungsende, setze belegt-reg und verb-Matrix zurueck
    -- Beginne bei der Broadcast-Abfrage
    IF  active_reg(0) ='1'   THEN  -- broadcast - Abfrage
      IF  xmit_bsy(0) = '0' THEN   -- broadcast - Abfrage
             belegt_reg  := "00000000000000";
             active_reg(0)  := '0';
             free       <=  "11111111111111";
             Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
      END IF;  -- xmit_bsy if ..
    END IF;    -- active_reg if ..

    L2: FOR m in 1 to 13 loop  -- kanaele loop
        IF  xmit_bsy(m) = '0' THEN
         -- Suche erst mal die aktiven Eingaenge
          IF active_reg(m) ='1' THEN
            active_reg(m) := '0';
            -- Loesche die Belegungenn im belegt-reg
            belegt_reg := belegt_reg and not verbind(m);
            verbind(m) := zeros;
            free(m)    <= '1';
            Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

          END IF; -- end active if
        END IF;    -- end xmit_bsy if
        Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

      END loop; -- kanaele-loop
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1';

  END loop; -- main loop
--  END loop; -- reset loop
 END Process;
END ccar; -- End architecture
```

Quellcode des CC-Package

```
-------------------------------------------------
Library IEEE;
USE IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
```

```
package utilsCC is

--TYPE Address IS ARRAY(Integer Range 0 to 13) OF STD_LOGIC_VECTOR(3 downto 0);
-- TYPE matrix  IS ARRAY(Integer Range 0 to 13) OF STD_LOGIC_VECTOR(0 to 13);

-- changed for Monet

 subtype vec4 IS STD_LOGIC_VECTOR(3 downto 0);
 TYPE Address IS ARRAY (0 to 13) OF vec4;

 subtype vec13 IS STD_LOGIC_VECTOR(0 to 13);
 TYPE matrix  IS ARRAY (0 to 13) OF vec13;

  function CONV_INTEGER(ARG: STD_LOGIC_VECTOR) return INTEGER;

  function convert(n: natural) return STD_LOGIC_VECTOR;
  function conv_to_nat(add : STD_LOGIC_VECTOR) return natural;

end;

--------------------------------------------------

package body utilsCC is

    function CONV_INTEGER(ARG: STD_LOGIC_VECTOR) return INTEGER is
        variable result    : UNSIGNED(ARG'range);
    begin
        result    := UNSIGNED(ARG);
        return   CONV_INTEGER(result);
    end;



function convert(n : natural) return STD_LOGIC_VECTOR IS
  VARIABLE temp  : STD_LOGIC_VECTOR(3 downto 0);

  begin

    CASE n IS
WHEN 0  => temp := "0000";
WHEN 1  => temp := "0001";
WHEN 2  => temp := "0010";
        WHEN 3  => temp := "0011";
WHEN 4  => temp := "0100";
WHEN 5  => temp := "0101";
        WHEN 6  => temp := "0110";
        WHEN 7  => temp := "0111";
WHEN 8  => temp := "1000";
WHEN 9  => temp := "1001";
WHEN 10 => temp := "1010";
WHEN 11 => temp := "1011";
WHEN 12 => temp := "1100";
WHEN 13 => temp := "1101";
WHEN 14 => temp := "1110";
WHEN 15 => temp := "1111";
        WHEN OTHERS => temp := "0000";
     END CASE;

   return temp;
  end convert;


   function conv_to_nat(add : STD_LOGIC_VECTOR) return natural IS
     VARIABLE temp : Natural;
     VARIABLE address : STD_LOGIC_VECTOR(3 downto 0);

  begin
    address := add;
    CASE address IS
        WHEN "0000"  => temp := 0;
WHEN "0001"  => temp := 1;
WHEN "0010"  => temp := 2;
        WHEN "0011"  => temp := 3;
WHEN "0100"  => temp := 4;
WHEN "0101"  => temp := 5;
        WHEN "0110"  => temp := 6;
        WHEN "0111"  => temp := 7;
WHEN "1000"  => temp := 8;
WHEN "1001"  => temp := 9;
WHEN "1010"  => temp := 10;
WHEN "1011"  => temp := 11;
WHEN "1100"  => temp := 12;
WHEN "1101"  => temp := 13;
WHEN "1110"  => temp := 14;
WHEN "1111"  => temp := 15;
        WHEN OTHERS => temp := 0;
     END CASE;

   return temp;
  end conv_to_nat;

end utilsCC;
```


Monet-Scheduling-Report

```
Solution: Solution_1
Project: cc
Project File:
/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/cc_mon
/cc.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src
/std_logic_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet
/pkgs/qhdl_libs/src/syn_arit.vhd -Lieee -93}
 {/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/cc_mon/src
/utilsCC.vhd -Lwork -87} {/afs/wsi/sun4m_54/ti/mentor/MONET/monet
/pkgs/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}
{/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/cc_mon/src
/cc_mon.vhd -Lwork -87}

Processes in design:
   ccin

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: ccin
-----------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     88 (84 are filtered out)
      "Pseudo" operations: 264

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: rst (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ----------------------------------  --------  --------   -------------
      AndGate(14,2)                         28.00        1           28.00
      InvGate(14)                           14.00        3           42.00
      OrGate(14,2)                          28.00        1           28.00
      cmp_cla(0,1,1,1,4,4)                  38.00        1           38.00
      inc_vn(0,4)                           18.00        1           18.00
      mux_mux8a(16,14)                     375.01        1          375.01
      mux_mux8a(16,4)                      123.89        1          123.89
      rdcor_two(3,14)                       13.32        1           13.32
      read_indexB(14,0)                      0.00        1            0.00
      read_indexB(14,1)                      0.00        1            0.00
      write_indexB(1,0)                      0.00        2            0.00
      write_indexB(14,1)                     0.00        4            0.00
                                                                 -------------
                                                     Total Area:    666.22

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   -----------    ------------------------------
          0    ASSIGN                    76                  PSEUDO
         349 ops removed w.l.
          99    IO_R<=oa(9)             178                  PSEUDO

   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      -----------------------  -----------------------  ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   17 C-Steps
```

```
      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   -----------    ------  -----
        190    READSLICE               31                      4    0.001
        304 delays removed (all 0.001 ns) w.l.
        326    RISINGEDGE              244                    17    0.001

   Pipelined Loops
   ---------------
                                                 Initiation
      Loop                            N_Unroll    Interval    Latency
      ------------------------------  --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ------------------------------ ------ ------ -------  --------  ----------  -------
      main_loop                       15      3      17
        l1                             9      4      12
          loop127                      4      5       8
        l2                             3     14      16

Area Estimate
-------------
   Component                                    Post   Post   Post
   Name                              Area      Alloc  Bind   Share
   ---------                      ----------   -----  -----  -----
   AndGate(14,2)                    28.000         1      0      0
   InvGate(14)                      14.000         3      0      0
   OrGate(14,2)                     28.000         1      0      0
   cmp_cla(0,1,1,1,4,4)             38.000         1      0      0
   inc_vn(0,4)                      18.000         1      0      0
   mux_mux8a(16,14)                375.012         1      0      0
   mux_mux8a(16,4)                 123.886         1      0      0
   rdcor_two(3,14)                  13.323         1      0      0
   read_indexB(14,0)                 0.000         1      0      0
   read_indexB(14,1)                 0.000         1      0      0
   write_indexB(1,0)                 0.000         2      0      0
   write_indexB(14,1)                0.000         4      0      0

      TOTAL AREA (After Allocation):              666.221

Area Estimate
-------------
                 Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area        666.2              0.0               0.0               0.0
     Total Reg         0.0              0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath        666.2 (100%)       0.0               0.0               0.0
       MUX           498.9  (75%)       0.0               0.0               0.0
       FUNC           69.3  (10%)       0.0               0.0               0.0
       LOGIC          98.0  (15%)       0.0               0.0               0.0
       BUFFER          0.0               0.0               0.0               0.0
       DP-REG          0.0               0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM               0.0               0.0               0.0               0.0
       FSM-REG         0.0               0.0               0.0               0.0
       FSM-COMB        0.0               0.0               0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Detailed timing not available until after Component Binding

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Finite State Machines
---------------------

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                  PADZEROES             LOOPEND
          IO_R                  SIGNEXTEND            SELECT
          READSLICE             NOP                   SELECTEND
          WRITESLICE            END                   CONDITIONALTRANSFER
          ASSIGN                ITERATE               RISINGEDGE
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R          WRITEINDEX   OR           XNOR          SKEDLIST
          IO_W          AND          NOR          NOT
          READINDEX     NAND         XOR          MUX
   Schedule Filter (default):
          PADZEROES             SELECT                CONCATENATE
          SIGNEXTEND            SELECTEND             READSLICE
          END                   CONDITIONALTRANSFER   READINDEX
          LOOP                  RISINGEDGE
          LOOPEND               FALLINGEDGE
```

End of Report

12.6 Datenaufnahme-Modul RD

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des rd (receive data) fuer
-- Simulation mit quicksim.
-- Synthese mit Monet
-- rd_mon.vhd nimmt zunaechst in einem 5-Bit Schieberegister
-- die Daten aus dem Switch auf.
-- Das Interface muss gegenueber rd.vhd geaendert werden,
-- da wegen des BC auch hier das SR ausgelagert wird.
-- das SR liefert 48 bit Daten parallel.
-- Die ersten 5 bit tragen das Startzeichen, die dekodiert werden.
-- Der decoder ist ein 5B/4B  Decoder.
-- Jede Zelle beginnt mit einem Start-Of-Cell Zeichen (’11001’)
-- Jedes der 12 folgenden Zellworte (mit 48 Bit) der Payload
-- beginnt mit einem Start-Of-Payload Zeichen (’10001’) (reg-start)
-- Die letzten 3 bits des Zellworts werden ignoriert.
-- Der 5B/4B decoder prueft auf Fehler: Wenn ein unerlaubtes
-- Zeichen kommt, wird das Signal ‘‘error’’ ungleich Null
-- In diesem Fall wird das Schreiben in den FIFO-OUT Buffer untrdrueckt..
-- Nach jeder Zelluebertragung muessen wenigstens 2 Leertakte kommen
-- rd.vhd  ist in 2 Prozesse aufgeteilt:
-- RDIN nimmt die Daten (rcell_data) auf, im Unterschied zu rd.vhd
-- aus einem 5 bit Schieberegister (hilf5),
-- dekodiert sie und setzt sie in ein 32 Bit Reg (out_reg).
-- RDOUT schreibt das out_reg in das out_fifo mit rcell_rdy, rcell_bus
-- Der Takt wird geaendert zu Clk_com (statt Clk_transmit)
-- Monet doesn’t like BOOLEAN Signals
-- Autor: W. Lange 7. 8. 98

Library IEEE;
USE IEEE.std_logic_1164.all;
    use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY rd IS
  PORT (Clk_com       : IN  STD_LOGIC;
        Reset       : IN  STD_LOGIC;         -- fuer BC eingefuehrt
        sr_strobe     : IN  STD_LOGIC;
        sr_data       : STD_LOGIC_VECTOR(47 DOWNTO 0);
        srdta_taken   : OUT STD_LOGIC;
        rcell_rdy     : OUT STD_LOGIC;
        rcell_fetch   : IN  STD_LOGIC;
        rbuffer_full  : IN  STD_LOGIC;
        rcell_bus     : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END rd;
---------------------------------
Use WORK.utilsrd.all;

ARCHITECTURE rd_ar OF rd IS
  SIGNAL out_reg      :  STD_LOGIC_VECTOR(31 downto 0);
--  SIGNAL write_fifo   :  BOOLEAN;
  SIGNAL write_fifo   :  STD_LOGIC;
  SIGNAL error_h      :  STD_LOGIC_VECTOR(4 downto 0);
  SIGNAL error_pl     :  STD_LOGIC_VECTOR(4 downto 0);
-- SIGNAL wr_ack       :  BOOLEAN;
  SIGNAL wr_ack       :  STD_LOGIC;

BEGIN

  RDIN: PROCESS
  -- Der RDIN Process nimmt die Zelle vom Switch auf, dekodiert sie,
  -- und gibt sie an den FIFO-OUT weiter.

   VARIABLE hilf_reg     :  STD_LOGIC_VECTOR(47 downto 0);
   VARIABLE hilf5        :  STD_LOGIC_VECTOR(4 downto 0);
   VARIABLE err,err_hold :  STD_LOGIC_VECTOR(4 downto 0);
   VARIABLE hilf4        :  STD_LOGIC_VECTOR(3 downto 0);
--   VARIABLE i,k,m        :  NATURAL ;   -- i,k  fuer v_parser .. (BC)
   VARIABLE out_reg_h    :  STD_LOGIC_VECTOR(31 downto 0);
   CONSTANT start_cell   :  STD_LOGIC_VECTOR(4 downto 0):= "11001";
   CONSTANT start_payl   :  STD_LOGIC_VECTOR(4 downto 0):= "10001";
   CONSTANT header_err   :  STD_LOGIC_VECTOR(4 downto 0):= "00001";
   CONSTANT payl_err     :  STD_LOGIC_VECTOR(4 downto 0):= "00010";
   -- attribute dont_unroll : boolean;
   attribute dont_unroll of L3 : label is true;

-- Bitlokationen von out_reg und Zellwort
--                    31  28  27 24 23 20 19 16 15 12 11 8  7  4  3  0
-- out_reg :=          "0000  0000  0000  0000  0000  0000  0000  0000";
--              47  43 42 38 37 33 32 28 27 23 22 18 17 13 12  8 7   3 2 0
-- Zellwort  := "11001 00000 00000 00000 00000 00000 00000 00000 00000 000";


  BEGIN -- Process
  --  Reset: Setze alle Variablen und Signale zurueck ----------------------
```

```
  reset_loop: loop
    srdta_taken <= '0';
    hilf_reg    := "00000000000000000000000000000000000000000000000000";
    err         := "00000";
    err_hold    := "00000";
    write_fifo  <= '0';
    error_h     <= "00000";
    error_pl    <= "00000";
    Wait UNTIL Clk_com'EVENT AND Clk_com = '1'; -- noetig, sonst ist
-- moeglicherweise eine loop ohne wait's
  --  eine neue Zelle kommt an...
  WHILE (rbuffer_full = '0')  loop    -- main loop ---------------------------

    sr_strobe_loop: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit sr_strobe_loop  when (sr_strobe  = '1');
    end loop;

    -- Der Header ist da..     ------------------------------

    hilf_reg := sr_data;

    srdta_taken <= '1';

    sr_strobe_loop_0: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit sr_strobe_loop_0  when (sr_strobe  = '0');
    end loop;

    srdta_taken <= '0';

    IF hilf_reg(47 downto 43) = start_cell THEN
    -- Das Startzeichen ist da, sammle den Header auf..

      hilf5 := hilf_reg(42 downto 38);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(31 downto 28) := hilf4;

      hilf5 := hilf_reg(37 downto 33);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(27 downto 24) := hilf4;

      hilf5 := hilf_reg(32 downto 28);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(23 downto 20) := hilf4;

      hilf5 := hilf_reg(27 downto 23);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(19 downto 16) := hilf4;

      hilf5 := hilf_reg(22 downto 18);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(15  downto 12) := hilf4;

      hilf5 := hilf_reg(17 downto 13);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(11 downto 8) := hilf4;

      hilf5 := hilf_reg(12 downto 8);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(7 downto 4) := hilf4;

      hilf5 := hilf_reg(7 downto 3);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(3 downto 0) := hilf4;

-- IF err = "00000" THEN write_fifo <= '1'; END IF; -- fuer Fehlerbehandlung
      write_fifo <= '1';
      error_h <= err_hold;
      out_reg <=  out_reg_h;

    ELSE error_h <= header_err;

    END IF;

      wr_ack_loop: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit wr_ack_loop  when (wr_ack  = '1');
      end loop;

    write_fifo <= '0';
    -- Wait UNTIL Clk_com'EVENT AND Clk_com = '1';
```

```
   -- Sammle die Payload auf

   L3 : FOR k in 1 to 12 loop

    sr_strobe_loop_1: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit sr_strobe_loop_1  when (sr_strobe  = '1');
    end loop;

    -- Das Zellwort ist da..   ----------------------

    hilf_reg := sr_data;

    srdta_taken <= '1';

    sr_strobe_loop_01: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit sr_strobe_loop_01  when (sr_strobe  = '0');
    end loop;

    srdta_taken <= '0';

     IF  hilf_reg(47 downto 43)  = start_payl THEN
    -- Das Startzeichen ist da, sammle das Zellwort auf..

      hilf5 := hilf_reg(42 downto 38);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(31 downto 28) := hilf4;

      hilf5 := hilf_reg(37 downto 33);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(27 downto 24) := hilf4;

      hilf5 := hilf_reg(32 downto 28);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(23 downto 20) := hilf4;

      hilf5 := hilf_reg(27 downto 23);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(19 downto 16) := hilf4;

      hilf5 := hilf_reg(22 downto 18);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(15  downto 12) := hilf4;

      hilf5 := hilf_reg(17 downto 13);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(11 downto 8) := hilf4;

      hilf5 := hilf_reg(12 downto 8);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(7 downto 4) := hilf4;

      hilf5 := hilf_reg(7 downto 3);
      decode(hilf5,hilf4,err);
      IF err /= "00000" THEN err_hold := err; END IF;
      out_reg_h(3 downto 0) := hilf4;

-- IF err = "00000" THEN write_fifo <= TRUE; END IF; -- fuer Fehlerbehandlung


       write_fifo <= '1';
       error_pl <= err_hold;
       out_reg <=  out_reg_h;

    ELSE error_pl <= payl_err;

    END IF;

    wr_ack_loop_1: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit wr_ack_loop_1  when (wr_ack  = '1');
    end loop;

    write_fifo <= '0';

   End loop;     -- 1 to 12 loop
   -- WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
   End loop;     -- Main Loop
  End loop;      -- reset_loop
  End Process;
```

```
  RDOUT: PROCESS
  -- Der RDOUT Process laedt das out_register in den rd-FIFO
  -- und kommuniziert mit dem rd-FIFO
  -- Achtung, Kommunikation zwischen RDIN und RDOUT
  -- ueber write_fifo ist synchron. Das kann Probleme geben, wenn
  -- rcell_fetch zu lange ausbleibt.
  -- rcell_fetch
  -- sollte innerhalb von 4 Takten nach rcell_rdy kommen.

  VARIABLE Cell_reg    : UNSIGNED(31 downto 0);  -- Reg fuer Cell Bytes

  BEGIN -- Process
  reset_loop: loop
  --  Reset: Setze alle Variablen und Signale zurueck ----------------------
    rcell_rdy  <= '0';
    rcell_bus <= "00000000000000000000000000000000";
    wr_ack <= '0';
    Wait UNTIL Clk_com'EVENT AND Clk_com  = '1';

  WHILE (rbuffer_full = '0')  loop    -- main loop ---------------------------

     wr_fifo_loop_1: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit wr_fifo_loop_1  when (write_fifo  = '1');
     end loop;

     rcell_rdy <= '1';
     rcell_bus <= out_reg;
     wr_ack    <= '1';

     wr_fifo_loop_0: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit wr_fifo_loop_0  when (write_fifo  = '0');
     end loop;

     wr_ack <= '0';

     rcell_fetch_loop: loop   -- fuer monet
WAIT UNTIL  Clk_com'EVENT and Clk_com = '1';
exit reset_loop when (Reset = '1');
exit rcell_fetch_loop  when (rcell_fetch  = '1');
     end loop;

     rcell_rdy <= '0';
     Wait UNTIL Clk_com'EVENT AND Clk_com  = '1';

  END loop; -- main loop
  End loop; -- reset_loop
  END Process;
 END rd_ar;
```

Quellcode des RD-Package

```
-------------------------------------------------
Library IEEE;
USE IEEE.std_logic_1164.all;
    use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

package utilsrd is

  procedure decode(v   : IN  STD_LOGIC_VECTOR(4 downto 0);
                   ret : OUT STD_LOGIC_VECTOR(3 downto 0);
                   err : OUT STD_LOGIC_VECTOR(4 downto 0));

end;

-------------------------------------------------

package body utilsrd is

 procedure decode( v   : IN  STD_LOGIC_VECTOR(4 downto 0);
                   ret : OUT STD_LOGIC_VECTOR(3 downto 0);
                   err : OUT STD_LOGIC_VECTOR(4 downto 0)) IS

  VARIABLE vector : STD_LOGIC_VECTOR(4 downto 0);

  begin
    vector := v;
    err := "00000";
    CASE vector IS
WHEN  "11110" => ret :=  "0000";
WHEN  "01001" => ret :=  "0001";
        WHEN  "10100" => ret :=  "0010";
WHEN  "10101" => ret :=  "0011";
WHEN  "01010" => ret :=  "0100";
        WHEN  "01011" => ret :=  "0101";
        WHEN  "01110" => ret :=  "0110";
WHEN  "01111" => ret :=  "0111";
```

```
WHEN  "10010" => ret :=  "1000";
WHEN  "10011" => ret :=  "1001";
WHEN  "10110" => ret :=  "1010";
WHEN  "10111" => ret :=  "1011";
WHEN  "11010" => ret :=  "1100";
WHEN  "11011" => ret :=  "1101";
WHEN  "11100" => ret :=  "1110" ;
WHEN  "11101" => ret :=  "1111";
WHEN OTHERS   => err :=    v; -- Probleme mit v_parser
     END CASE;

  end decode;
end utilsrd;

--------------------------------------------------
--------------------------------------------------
```

Monet-Scheduling-Report

```
Solution: Solution_1
Project: rd
Project File:
/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/rd_mon
/rd.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src
/std_logic_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet
/pkgs/qhdl_libs/src/syn_arit.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor
/MONET/monet/pkgs/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}
{/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/rd_mon/src
/utilsrd.vhd -Lwork -87} {/afs/informatik.uni-tuebingen.de/home/
wlange/hilan/aht/rd_mon/src/rd_mon.vhd -Lwork -87}

Processes in design:
   rdin rdout

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: rdin
--------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     7 (3 are filtered out)
      "Pseudo" operations: 694

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      InvGate(1)                           1.00        1            1.00
      cmp_cla(0,1,1,1,5,5)                46.12        1           46.12
      inc_vn(0,4)                         18.00        1           18.00
                                                              -------------
                                                 Total Area:     65.12

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ------------------------------
          0    LOOP                      85   reset_loop     PSEUDO
         670 ops removed w.l.
          99    ASSIGN                   37                  PSEUDO

   Cycle Constraints
   -----------------
      From                    To                      Constraint   Actual  Met?
      ----------------------  ----------------------  ----------   ------  ----
```

```
   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   10 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
         40    ASSIGN                   27                      4    0.001
        668 delays (all 0.001) removed w.l.
        220    ASSIGN                  263                      9    0.001

   Pipelined Loops
   ---------------
                                                 Initiation
      Loop                              N_Unroll    Interval    Latency
      -------------------------------   --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ----------------------------  ------ ------ -------  --------  ----------  -------
      loop96                           9      2      10
        sr_strobe_loop                 1      2       2
        sr_strobe_loop_0               1      3       3
        wr_ack_loop                    1      5       5
        l3                             5      6      10
          sr_strobe_loop_1             1      6       6
          sr_strobe_loop_01            1      7       7
          wr_ack_loop_1                1      9       9


Process: rdout
------------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     7 (3 are filtered out)
      "Pseudo" operations: 694

   Clock Information
   -----------------
      Clock Signal: clk_com (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      InvGate(1)                           1.00        1            1.00
                                                              -------------
                                                     Total Area:      1.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   ----------     ----------------------------
          0    LOOP                     284   reset_loop     PSEUDO
          1    IO_W=>rcell_rdy          286   &3             PSEUDO
         10    ASSIGN                   306                  PSEUDO
         11    IO_W=>rcell_rdy          299                  PSEUDO
         12    IO_W=>rcell_bus          300                  PSEUDO
         13    ASSIGN                   301                  PSEUDO
         14    LOOP                     303   wr_fifo_lo     PSEUDO
         15    RISINGEDGE               304                  PSEUDO
         16    TERMINATE                306                  PSEUDO
         17    NOT                      306                  InvGate(1)
         18    ASSIGN                   309                  PSEUDO
         19    LOOP                     311   rcell_fetc     PSEUDO
          2    IO_W=>rcell_bus          287   &5             PSEUDO
         20    RISINGEDGE               312                  PSEUDO
         21    TERMINATE                314                  PSEUDO
         22    IO_W=>rcell_rdy          317   &7             PSEUDO
         23    RISINGEDGE               318   &8             PSEUDO
         24    LOOPEND                  284                  PSEUDO
         25    LOOPEND                  291                  PSEUDO
         26    LOOPEND                  293                  PSEUDO
         27    LOOPEND                  303                  PSEUDO
         28    LOOPEND                  311                  PSEUDO
          3    ASSIGN                   288   &6             PSEUDO
          4    RISINGEDGE               289   &4             PSEUDO
          5    LOOP                     291   loop291        PSEUDO
```

```
        6    TERMINATE               291                  PSEUDO
        7    LOOP                    293   wr_fifo_lo     PSEUDO
        8    RISINGEDGE              294                  PSEUDO
        9    TERMINATE               296                  PSEUDO

   Cycle Constraints
   -----------------
      From                     To                      Constraint   Actual  Met?
      -----------------------  ----------------------  ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH    6 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   ----------     ------  -----
          0    LOOP                     284   reset_loop        1    0.000
         24    LOOPEND                  284                     6    0.000
          1    IO_W=>rcell_rdy          286   &3                1    0.001
          2    IO_W=>rcell_bus          287   &5                1    0.001
          3    ASSIGN                   288   &6                1    0.001
          4    RISINGEDGE               289   &4                2    0.001
          5    LOOP                     291   loop291           2    0.000
          6    TERMINATE                291                     2    0.001
         25    LOOPEND                  291                     6    0.000
          7    LOOP                     293   wr_fifo_lo        2    0.000
         26    LOOPEND                  293                     2    0.000
          8    RISINGEDGE               294                     2    0.001
          9    TERMINATE                296                     2    0.001
         11    IO_W=>rcell_rdy          299                     2    0.001
         12    IO_W=>rcell_bus          300                     2    0.001
         13    ASSIGN                   301                     2    0.001
         14    LOOP                     303   wr_fifo_lo        3    0.000
         27    LOOPEND                  303                     3    0.000
         15    RISINGEDGE               304                     3    0.001
         16    TERMINATE                306                     3    0.001
         17    NOT                      306                     3    0.090
         18    ASSIGN                   309                     3    0.001
         28    LOOPEND                  311                     4    0.000
         19    LOOP                     311   rcell_fetc        4    0.000
         20    RISINGEDGE               312                     4    0.001
         21    TERMINATE                314                     4    0.001
         22    IO_W=>rcell_rdy          317   &7                5    0.001
         23    RISINGEDGE               318   &8                6    0.001

   Pipelined Loops
   ---------------
                                                 Initiation
      Loop                              N_Unroll    Interval    Latency
      -------------------------------   --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ------------------------------  ------ ------ -------  --------  ----------  -------
      loop291                          5      2       6
        wr_fifo_loop_1                 1      2       2
        wr_fifo_loop_0                 1      3       3
        rcell_fetch_loop               1      4       4

Area Estimate
-------------
   Component                                  Post   Post   Post
   Name                            Area      Alloc   Bind  Share
   ---------                     ----------   -----  -----  -----
   InvGate(1)                       1.000        2      4      4
   cmp_cla(0,1,1,1,5,5)            46.122        1      3      3
   cmp_rpl1(0,1,1,1,5,5)           42.594        0    256    256
   inc_vn(0,4)                     18.000        1      1      1
   AndGate(1,2)                     2.000        0      2      2
   AndGate(32,2)                   64.000        0      1      1
   InvGate(32)                     32.000        0      1      1
   Mux1h(1,2)                       3.000        0      2      2
   Mux1h(1,3)                       4.000        0     66     66
   Mux1h(32,2)                     96.000        0      1      1
   Mux1h(32,3)                    128.000        0      1      1
   Mux1h(4,2)                      12.000        0      1      1
   Mux1h(4,3)                      16.000        0      1      1
   NorGate(1,2)                     1.000        0     68     68
   OrGate(1,2)                      2.000        0    797    797
   Register(1,0,0)                  7.000        0      4      4
   Register(32,0,0)               224.000        0      2      2
   Register(4,0,0)                 28.000        0      2      2
   Register(48,0,0)                336.000        0      1      1
   mux_aoi(2,1)                     4.000        0      1      1
   mux_aoi(2,4)                    12.683        0      1      1
   mux_mux2(2,48)                  129.458        0      1      1
   rdcnor_four(4,8)                 5.537        0      1      1
   rdcnor_four(4,9)                 6.301        0      2      2

      TOTAL AREA (After Sharing):               14380.700
```

```
Area Estimate
-------------
                Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area         66.1              66.1            14380.7           14380.7
     Total Reg         0.0               0.0              868.0   (6%)      868.0   (6%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath         66.1 (100%)       66.1 (100%)    14380.7 (100%)    14380.7 (100%)
       MUX             0.0               0.0              668.1   (5%)      668.1   (5%)
       FUNC           64.1  (97%)       64.1  (97%)    11078.5  (77%)    11078.5  (77%)
       LOGIC           2.0   (3%)        2.0   (3%)     1766.0  (12%)     1766.0  (12%)
       BUFFER          0.0               0.0                0.0               0.0
       DP-REG          0.0               0.0              868.0   (6%)      868.0   (6%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM               0.0               0.0                0.0               0.0
       FSM-REG         0.0               0.0                0.0               0.0
       FSM-COMB        0.0               0.0                0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Critical Path
     Max Delay 10.1226
     Slack 14.8774
     Resource: r1484rg(cmp_rpl1_0_1_1_1_5_5) Signal: n_16dp Unit delay: 2.75263 Delay: 2.75263
     Resource: r2751rg(OrGate_1_2) Signal: n_649tmp Unit delay: 0.33 Delay: 3.08263
     Resource: r2754rg(OrGate_1_2) Signal: n_650tmp Unit delay: 0.33 Delay: 3.41263
     Resource: r2757rg(OrGate_1_2) Signal: n_651tmp Unit delay: 0.33 Delay: 3.74263
     Resource: r2760rg(OrGate_1_2) Signal: n_652tmp Unit delay: 0.33 Delay: 4.07263
     Resource: r2763rg(OrGate_1_2) Signal: n_653tmp Unit delay: 0.33 Delay: 4.40263
     Resource: r2766rg(OrGate_1_2) Signal: n_654tmp Unit delay: 0.33 Delay: 4.73263
     Resource: r2769rg(OrGate_1_2) Signal: n_655tmp Unit delay: 0.33 Delay: 5.06263
     Resource: r4644rg(NorGate_1_2) Signal: n_1280tmp Unit delay: 0.18 Delay: 5.24263
     Resource: r1652rg(Mux1h_1_3) Signal: n_32dp Unit delay: 0.49 Delay: 5.73263
     Resource: r1792rg(Mux1h_1_3) Signal: n_59dp Unit delay: 0.49 Delay: 6.22263
     Resource: r1618rg(Mux1h_1_3) Signal: n_88dp Unit delay: 0.49 Delay: 6.71263
     Resource: r1716rg(Mux1h_1_3) Signal: n_115dp Unit delay: 0.49 Delay: 7.20263
     Resource: r1767rg(Mux1h_1_3) Signal: n_143dp Unit delay: 0.49 Delay: 7.69263
     Resource: r1637rg(Mux1h_1_3) Signal: n_168dp Unit delay: 0.49 Delay: 8.18263
     Resource: r1632rg(Mux1h_1_3) Signal: n_192dp Unit delay: 0.49 Delay: 8.67263
     Resource: r1777rg(Mux1h_1_3) Signal: n_216dp Unit delay: 0.49 Delay: 9.16263
     Resource: r1405rg(Mux1h_4_2) Signal: n_1400tmp Unit delay: 0.41 Delay: 9.57263
     Resource: r1408rg(mux_aoi_2_4) Signal: hilf4_pre_ff Unit delay: 0.55 Delay: 10.1226

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: 25
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: 25
       Resource r1399rg Port write_fifo_pre_ff 19.71
       Resource r1409rg Port hilf4_pre_ff 14.8774
       Resource r1418rg Port l3__k_pre_ff 23.52
       Resource r1427rg Port out_reg_pre_ff 15.3474
       Resource r1431rg Port hilf_reg_pre_ff 22.9997
       Resource r1437rg Port srdta_taken_pre_ff 20.04
       Resource r2165rg Port wr_ack_pre_ff 22.34
       Resource r2180rg Port rcell_rdy_pre_ff 22.34
       Resource r2197rg Port rcell_bus_pre_ff 22.67

   FSM name: rdin (rd_sid0_rdin)
     Total area: 1042
       Combinational area: 783 Sequential area: 259

     Input to output delays

     Input to register-input delays
       n_459dp: 0.48
       n_459dp: 0.48
       n_459dp: 0.45
       wr_ack: 0.48
       sr_strobe: 0.44
       rbuffer_full: 0.57
       reset: 0.57

     Register-output to output delays
       clk_com->n_52_fsm: 0
       16 delays removed: (all 0) w.l.
       clk_com->n_12_fsm: 0


   FSM name: rdout (rd_sid0_rdout)
     Total area: 357
       Combinational area: 222 Sequential area: 135

     Input to output delays

     Input to register-input delays
       write_fifo: 0.44
       rcell_fetch: 0.35
       rbuffer_full: 0.35
       reset: 0.35

     Register-output to output delays
       clk_com->n_80_fsm: 0
       clk_com->n_78_fsm: 0
```

```
       clk_com->n_97_fsm: 0
       clk_com->n_82_fsm: 0
       clk_com->n_84_fsm: 0
       clk_com->n_86_fsm: 0
       clk_com->n_88_fsm: 0
       clk_com->n_76_fsm: 0
       clk_com->n_90_fsm: 0
       clk_com->n_92_fsm: 0


   ---------------------------------------------------------------

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Finite State Machines
---------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r1431rg         48 bits    hilf_reg
   r2165rg          1 bits    wr_ack
   r1427rg         32 bits    out_reg
   r1418rg          4 bits    l3__k
   r1409rg          4 bits    hilf4
   r1399rg          1 bits    write_fifo
   r2197rg         32 bits    rcell_bus
   r2180rg          1 bits    rcell_rdy
   r1437rg          1 bits    srdta_taken

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                  PADZEROES             LOOPEND
          IO_R                  SIGNEXTEND            SELECT
          READSLICE             NOP                   SELECTEND
          WRITESLICE            END                   CONDITIONALTRANSFER
          ASSIGN                ITERATE               RISINGEDGE
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R         WRITEINDEX   OR           XNOR         SKEDLIST
          IO_W         AND          NOR          NOT
          READINDEX    NAND         XOR          MUX
   Schedule Filter (default):
          PADZEROES             LOOPEND               RISINGEDGE
          SIGNEXTEND            SELECT                FALLINGEDGE
          END                   SELECTEND
          LOOP                  CONDITIONALTRANSFER

End of Report
```

12.7 AHT-Ausgangs-Modul

Quellcode der VHDL-Verhaltensbeschreibung

```
-- Verhaltensbeschreibung des ahtout fuer
-- Simulation mit qhsim. Compilierung mit qvcom,
-- Synthese mit Monet
-- aht_out besteht aus 2 Prozessen:
-- AHTOUT holt Zell-Daten aus dem FIFO_out-Speicher und
-- und gibt sie an den PROCESS AHTOUT_Transmit weiter
-- AHTOUT_Transmit gibt die Daten in 8-Bit Paketen (an die PHYS-Schicht)
-- auf die Leitung.
-- Es werden  53 Byte uebertragen (5 Byte fuer den Header, HEC:
-- Wiederholung des vorhergehenden Bytes)
-- Es wird davon ausgegengen, dass der HEC in der PHYS-Schicht
-- eingefuegt wird.
-- Autor: W. Lange 11.8.98


Library IEEE;
USE IEEE.std_logic_1164.all;
    use IEEE.std_logic_arith.all;
LIBRARY mgc_hls;
USE mgc_hls.defs.all ;

ENTITY ahtout IS
  PORT (Clk_aht_out    : IN  STD_LOGIC;
        Cell_Sync_out  : OUT STD_LOGIC;
        Cell_preSync   : OUT STD_LOGIC;
        Chan_bsy       : IN  STD_LOGIC;
        Reset          : IN STD_LOGIC;
        Data_out       : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        rnew_cell      : OUT STD_LOGIC;
        rcell_data     : IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
```

```
        rcell_read     : OUT STD_LOGIC;
        rcell_write    : IN  STD_LOGIC);
END ahtout;
---------------------------------

ARCHITECTURE ahtoutar OF ahtout IS
  SIGNAL transmit     : STD_LOGIC;
  SIGNAL reg_taken    : STD_LOGIC;
  SIGNAL cell_reg     : STD_LOGIC_VECTOR(31 downto 0);  --Reg fuer Cellwort

  -- As a possible Work Around, use the "sync_reset" attributes for
  -- defining the sync. reset condition   PKB
  ATTRIBUTE sync_reset OF ahtoutar : ARCHITECTURE IS "reset" ;
  ATTRIBUTE reset_phase OF AHTOUT_Transmit : LABEL IS "positive" ;
  ATTRIBUTE reset_phase OF AHTOUT : LABEL IS "positive" ;

BEGIN

  AHTOUT: PROCESS
  -- Der AHTOUT Process holt Zell-Daten aus dem FIFO_out-Speicher und
  -- und gibt sie an den Process AHTOUT_TRANSMIT weiter
  -- attribute dont_unroll of L1 : label is true;
   VARIABLE i,k       :  NATURAL;


  BEGIN -- Process
-- PKB The next line is not needed if using attribute "sync_reset"
-- PKB  reset_loop: loop
  --  Reset: Setze alle Variablen und Signale zurueck ----------------------
    rcell_read     <= '0';
    rnew_cell <= '0';
    transmit <= '0';
   Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';

   main_loop:  loop    -- main loop ---------------------------

      chan_bsy_loop: loop   -- fuer monet
        WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
        exit chan_bsy_loop  when ( Chan_bsy = '0');
     end loop;

     -- Der Ausgangskanal ist frei, hole eine Zelle aus dem FIFO-Speicher.
     -- Uebertrage erst den Header

      rcell_read <= '1';
      rnew_cell  <= '1';
      WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';

      rcell_write_loop: loop   -- fuer monet
--        cell_reg <= rcell_data;  -- wird mit rcell_write uebernommen
        WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
        exit rcell_write_loop  when ( rcell_write = '1');
      end loop;

      cell_reg <= rcell_data;  -- wird mit rcell_write uebernommen
      rcell_read <= '0';
      transmit <= '1';
      rnew_cell  <= '0';

     reg_taken_loop: loop   -- fuer monet
        WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
        exit reg_taken_loop  when ( reg_taken = '1');
    end loop;


      transmit <= '0';
     -- Wiederhole 12 mal fuer 32 - Bit-Register (48 Byte)

     L1: FOR i in 1 to 12 loop
       rcell_read <= '1';
       -- Warte auf die Antwort des Speicher-Mgmt's

       rcell_write_loop2: loop   -- fuer monet
--  cell_reg <= rcell_data;
         WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
         exit rcell_write_loop2  when ( rcell_write = '1');
       end loop;

       cell_reg <= rcell_data;
       rnew_cell  <= '0';
       rcell_read <= '0';
       transmit <= '1';

       reg_taken_loop2: loop   -- fuer monet
         WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
         exit reg_taken_loop2  when ( reg_taken = '1');
       end loop;

       transmit <= '0';
     End loop;
  End loop;  -- main loop
  End Process;
```

```
  AHTOUT_Transmit: PROCESS
     VARIABLE hilf         : STD_LOGIC_VECTOR(31 downto 0);
     -- attribute dont_unroll of L2 : label is true;

  BEGIN   -- Process
    -- Reset Part
    reg_taken <= '0';
    Cell_Sync_out  <= '0';
    Cell_preSync   <= '0';
    Data_out       <= "00000000";
    Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';

    main_loop: loop

    transmit_loop: loop   -- fuer monet
--        hilf := cell_reg;
        WAIT UNTIL  Clk_aht_out'EVENT and Clk_aht_out = '1';
        exit transmit_loop  when ( transmit = '1');
    end loop;

     Cell_preSync <= '1';
         hilf := cell_reg;
      Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
        Cell_Sync_out <= '1';
       -- Transmit Bytewise den Header
        reg_taken <= '1';
data_out <= hilf(31 downto 24);
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
data_out <= hilf(23 downto 16);
        Cell_Sync_out <= '0';
        Cell_preSync <= '0';
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
        reg_taken <= '0';
data_out <= hilf(15 downto 8);
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
data_out <= hilf(7 downto 0);
         Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
       -- Das HEC Byte ist die Wiederholung der vorhergehenden Daten
 hilf := cell_reg;
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';

     -- Transmit Bytewise die Payload

     L2: FOR k IN 1 TO 12 loop  -- 12 * 32 Bit register
--      IF (transmit = '1') THEN  -- weggelassen wegen add. Takt n. Syn
--      hilf := cell_reg;
        reg_taken <= '1';

data_out <= hilf(31 downto 24);
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
data_out <= hilf(23 downto 16);
        reg_taken <= '0';
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
data_out <= hilf(15 downto 8);
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
data_out <= hilf(7 downto 0);
        Wait UNTIL Clk_aht_out'EVENT AND Clk_aht_out = '1';
        hilf := cell_reg; -- verlegt
--      End IF;
    end loop; -- 1 to 12 loop

   End loop; -- Main Loop
-- PKB The next line is not needed if using attribute "sync_reset"
-- PKB   End loop; -- reset_loop
  End Process;

 END ahtoutar;
```

Monet-Scheduling-Report

```
Solution: Solution_1
Project: ahtout
Project File: /afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht
/ahtout_mon/ahtout.ise
Design Source Specified:
   {/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src/standard.vhd
 -Lstd -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet/pkgs/qhdl_libs/src
/std_logic_1164.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor/MONET/monet
/pkgs/qhdl_libs/src/syn_arit.vhd -Lieee -93}
{/afs/wsi/sun4m_54/ti/mentor
/MONET/monet/pkgs/hls_pkgs/src/attributes.vhd -Lmgc_hls -93}

{/afs/informatik.uni-tuebingen.de/home/wlange/hilan/aht/ahtout_mon/src
/ahtout_mon.vhd -Lwork -87}

Processes in design:
   ahtout ahtout_transmit

Component Libraries In Use:
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_dmag_dc.lib
```

```
   $MGC_HOME/pkgs/siflibs.any/mgc_lca300k_comp_dc.lib


Process: ahtout
-------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     1 (1 are filtered out)
      "Pseudo" operations: 360

   Clock Information
   -----------------
      Clock Signal: clk_aht_out (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
      InvGate(1)                            1.00         1            1.00
                                                                -------------
                                                     Total Area:      1.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   -----------    ------------------------------
          0    IO_W=>rcell_read          60   &0             PSEUDO
         196 ops removed w.l.
          99    TERMINATE               106                  PSEUDO

   Cycle Constraints
   -----------------
      From                     To                       Constraint   Actual  Met?
      -----------------------  -----------------------  ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   29 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   -----------    ------  -----
        168    LOOP                      53   ahtout_loo        1    0.000
        177 delays removed (all 0.001) w.l.
        167    ASSIGN                   119                    29    0.001

   Pipelined Loops
   ---------------
                                              Initiation
      Loop                             N_Unroll    Interval    Latency
      -------------------------------  --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ----------------------------  ------ ------ -------  --------  ----------  -------
      main_loop                       28      2      29
        chan_bsy_loop                  1      2       2
        rcell_write_loop               1      4       4
        reg_taken_loop                 1      5       5
        i16_rcell_write_loop2          1     22      22
        i4_rcell_write_loop2           1     10      10
        i18_rcell_write_loop2          1     24      24
        i6_rcell_write_loop2           1     12      12
        i22_rcell_write_loop2          1     28      28
        i10_rcell_write_loop2          1     16      16
        i20_rcell_write_loop2          1     26      26
        i12_rcell_write_loop2          1     18      18
        i2_rcell_write_loop2           1      8       8
        i8_rcell_write_loop2           1     14      14
        i0_rcell_write_loop2           1      6       6
        i14_rcell_write_loop2          1     20      20
        i5_reg_taken_loop2             1     11      11
        i9_reg_taken_loop2             1     15      15
        i23_reg_taken_loop2            1     29      29
        i3_reg_taken_loop2             1      9       9
        i15_reg_taken_loop2            1     21      21
        i21_reg_taken_loop2            1     27      27
        i1_reg_taken_loop2             1      7       7
```

```
        i17_reg_taken_loop2             1     23      23
        i13_reg_taken_loop2             1     19      19
        i19_reg_taken_loop2             1     25      25
        i11_reg_taken_loop2             1     17      17
        i7_reg_taken_loop2              1     13      13


Process: ahtout_transmit
-------------------------------------------------
   I/O Scheduling Mode: Super

   Operation Count
   ---------------
      Real operations:     1 (1 are filtered out)
      "Pseudo" operations: 360

   Clock Information
   -----------------
      Clock Signal: clk_aht_out (Edge: Falling)
      Clock Period: 25.0000     Clock Overhead:  10.00 %

   Reset Information
   -----------------
      Reset Signal: reset (Mode: Sync, Phase: Positive)

   Allocation Constraints
   ----------------------
      Qualified Component Name               Area    Quantity   Area Subtotal
      ---------------------------------  --------  --------   -------------
                                                               -------------
                                                    Total Area:      0.00

   Multi-Cycle (Combinational) Component Usage
   -------------------------------------------
      Instance    Component Name                 Delay    #Cycles
      --------    ----------------------------  -------  -------

   Operation-to-Component Mappings
   -------------------------------
      Oper     Operation             Line                    Mapped to
      Number   Type                  Number   Label          Component
      ------   --------------------  ------   -----------    ------------------------------
          0    ASSIGN                   130   &7             PSEUDO
         161 ops removed w.l.
         99    IO_W=>data_out           177   &93            PSEUDO

   Cycle Constraints
   -----------------
      From                    To                       Constraint   Actual  Met?
      ----------------------  -----------------------  ----------   ------  ----

   Scheduled Operations
   --------------------
      TOTAL SCHEDULE LENGTH   57 C-Steps

      Oper     Operation             Line
      Number   Type                  Number   Label          C-Step  Delay
      ------   --------------------  ------   -----------    ------  -----
        159    LOOP                     125   ahtout_tra        1    0.000
        161 delays removed (all 0 or 0.001) w.l.
        158    ASSIGN                   181                     57    0.001

   Pipelined Loops
   ---------------
                                              Initiation
      Loop                             N_Unroll    Interval    Latency
      -------------------------------  --------   ----------   -------

   Loops
   -----
                                    Length  Start   End              Initiation
      Loop                          C-step C-step C-steps  N_Unroll   Interval   Latency
      ----------------------------  ------ ------ -------  --------  ----------  -------
      main_loop                       56      2      57
        transmit_loop                  1      2       2

Area Estimate
-------------
   Component                                      Post   Post   Post
   Name                                  Area     Alloc   Bind  Share
   ---------                          ----------   -----  -----  -----
   InvGate(1)                          1.000         1      3      3
   AndGate(1,2)                        2.000         0      3      3
   Mux1h(1,2)                          3.000         0      3      3
   Mux1h(1,3)                          4.000         0      3      3
   Mux1h(8,7)                         72.000         0      1      1
   NorGate(1,2)                        1.000         0      3      3
   NorGate(1,44)                      28.500         0      1      1
   NorGate(1,59)                      37.875         0      2      2
   OrGate(1,2)                         2.000         0    200    200
   Register(1,0,0)                     7.000         0      6      6
   Register(32,0,0)                  224.000         0      2      2
   Register(8,0,0)                    56.000         0      1      1
```

```
   mux_mux2(2,32)                     89.437         0      2      2
   rdcnor_four(4,6)                    3.906         0      1      1

      TOTAL AREA (After Sharing):               1338.030

Area Estimate
-------------
                Post-Scheduling    Post-DP & FSM      Post-Binding       Post-Sharing
   ------------  ----------------  ----------------  ----------------  ----------------
   Total Area          1.0               1.0             1338.0            1338.0
     Total Reg         0.0               0.0              546.0  (41%)      546.0  (41%)
   ------------  ----------------  ----------------  ----------------  ----------------
     DataPath          1.0 (100%)        1.0 (100%)     1338.0 (100%)     1338.0 (100%)
       MUX             0.0               0.0              271.9  (20%)      271.9  (20%)
       FUNC            0.0               0.0                3.9   (0%)        3.9   (0%)
       LOGIC           1.0 (100%)        1.0 (100%)       516.2  (39%)      516.2  (39%)
       BUFFER          0.0               0.0                0.0               0.0
       DP-REG          0.0               0.0              546.0  (41%)      546.0  (41%)
   ------------  ----------------  ----------------  ----------------  ----------------
     FSM               0.0               0.0                0.0               0.0
       FSM-REG         0.0               0.0                0.0               0.0
       FSM-COMB        0.0               0.0                0.0               0.0
   ------------  ----------------  ----------------  ----------------  ----------------
Timing Report
-------------
   Critical Path
     Max Delay 23.44
     Slack 1.56
     Resource: r557rg(OrGate_1_2) Signal: n_83tmp Unit delay: 0.33 Delay: 0.33
     69 delays removed w.l.
     Resource: r305rg(Mux1h_1_3) Signal: reg_taken_pre_ff Unit delay: 0.49 Delay: 23.44

   Slacks of outputs:
     Clock period or pin-to-pin delay constraint: 25
   Slacks of register inputs:
     Clock period or pin-to-reg delay constraint: 25
       Resource r243rg Port cell_reg_pre_ff 19.7945
       Resource r255rg Port rnew_cell_pre_ff 10.79
       Resource r264rg Port transmit_pre_ff 10.47
       Resource r270rg Port rcell_read_pre_ff 10.47
       Resource r282rg Port cell_presync_pre_ff 5.84
       Resource r297rg Port cell_sync_out_pre_ff 5.84
       Resource r306rg Port reg_taken_pre_ff 1.56
       Resource r312rg Port data_out_pre_ff 5.18259
       Resource r316rg Port hilf_pre_ff 19.7945

   FSM name: ahtout (ahtout_sid0_ahtout)
     Total area: 7676
       Combinational area: 6899 Sequential area: 777

     Input to output delays

     Input to register-input delays
       reg_taken: 0.35
       chan_bsy: 0.44
       reset: 0.35
       rcell_write: 0.35

     Register-output to output delays
       clk_aht_out->n_87_fsm: 0
       55 delays removed (all 0) w.l.
       clk_aht_out->n_85_fsm: 0


   FSM name: ahtout_transmit (ahtout_sid0_ahtout_transmit)
     Total area: 19266
       Combinational area: 17760 Sequential area: 1506

     Input to output delays

     Input to register-input delays
       reset: 0.35
       transmit: 0.35

     Register-output to output delays
       clk_aht_out->n_246_fsm: 0
       109 delays removed w.l.
       clk_aht_out->n_322_fsm: 0


   ----------------------------------------------------------------

Variables Mapped to Memories
----------------------------
   Variable       Process      Line#     Size     ResID  Ram/Rom  Component
   ---------   -------------  -------  ---------  -----  -------  ------------------

Register-to-Variable Mappings
-----------------------------
   Register     Size          Variables
   ----------   -----------   ---------
   r306rg           1 bits    reg_taken
   r270rg           1 bits    rcell_read
```

```
   r243rg          32 bits    cell_reg
   r316rg          32 bits    hilf
   r312rg           8 bits    data_out
   r297rg           1 bits    cell_sync_out
   r282rg           1 bits    cell_presync
   r264rg           1 bits    transmit
   r255rg           1 bits    rnew_cell

Filter Settings:
   Pseudo-Operation Filter:
          IO_W                  PADZEROES             LOOPEND
          IO_R                  SIGNEXTEND            SELECT
          READSLICE             NOP                   SELECTEND
          WRITESLICE            END                   CONDITIONALTRANSFER
          ASSIGN                ITERATE               RISINGEDGE
          ASSIGNDF              TERMINATE             FALLINGEDGE
          CONCATENATE           LOOP
   Allocation Filter:
          IO_R         WRITEINDEX   OR           XNOR         SKEDLIST
          IO_W         AND          NOR          NOT
          READINDEX    NAND         XOR          MUX
   Schedule Filter (default):
          PADZEROES             LOOPEND               RISINGEDGE
          SIGNEXTEND            SELECT                FALLINGEDGE
          END                   SELECTEND
          LOOP                  CONDITIONALTRANSFER

End of Report
```

13 Literatur

Literatur

[LaRo97] W.Lange, W. Rosenstiel, Modellierung einer ATM-Switch-Steuerung. WSI 97-6 Wilhelm-Schickard-Institut Universität Tübingen (Bericht) WSI 1997 ISSN 0964-3852

[DeMi94] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill (1994)

[Gaj92] Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, Steve Y-L Lin, High-Level Synthesis Introduction to Chip and System Design, Kluwer Academic Publishers (1992)

[Mum98] Mentor Graphics Co., Monet User's and Reference Manual Software Release R32 (May 1998)

[Montw98] Mentor Graphics Co., Monet Training Workbook (Q2 1998)