4.3 Optimierungsverfahren zur Hardwaresynthese

Zur Vorlesung

Embedded Systems

WS 14/15

Reiner Kolla

4.3.1 Besonderheiten bei der Hardwaresynthese

Aus den oben gemachten Ausführungen erkennt man, dass es bei einer direkten Umsetzung in Hardware Besonderheiten gibt, die wir im Kapitel 3 über Ablaufplanung noch nicht berücksichtigt haben:

- **Module mit Fließbandverarbeitung:** Bei HW-Implementierungen hat man nicht selten Ressourcen, die mit Pipeline-Registern versehen sind. Ein Modul x hat zwar nach wie vor eine Latenz von $w(x)$ Takten, kann aber schon nach $\pi(x) \leq w(x)$ Takten wieder eine Operation aufnehmen.

  Wir nehmen also im Folgenden an, dass zu allen Ressourcen $x \in V_T$ eine Periode $\pi(x)$ gegeben ist, mit der die Ressource neu gebunden werden kann.

  Dies ergibt folgende geänderte Bedingung an die Bindung:

  $\beta(u) = \beta(v) \Rightarrow \tau(u) \geq \tau(v) + \pi(\beta(v))$ oder $\tau(v) \geq \tau(u) + \pi(\beta(u))$

  anstelle von

  $\beta(u) = \beta(v) \Rightarrow \tau(u) \geq \tau(v) + w(\beta(v))$ oder $\tau(v) \geq \tau(u) + w(\beta(u))$

Besonderheiten bei der Hardwaresynthese ff

- **Bandbreitenbeschränkungen:** Bei busbasierten Zielarchitekturen hat man in der Regel auch eine beschränkte Bandbreite auf Operanden- und Resultatsbussen, die im Wesentlichen von der Bandbreite der Registerbank abhängt.

  Wir nehmen dazu im Folgenden an, dass zu allen Ressourcen $x \in V_T$ eine Operandenbreite $o(x)$ und eine Resultatsbreite $\rho(x)$ gegeben ist.

  Dies ergibt folgende Bedingungen an Ablaufplan und Bindung:

$$\sum_{u,\tau(u)=t} o(\beta(u)) \leq O$$

$$\sum_{u,\tau(u)+w(\beta(u))=t} \rho(\beta(u)) \leq R$$

  d.h. in keinem Zyklus darf die verfügbare Operandenbandbreite O oder die Resultatsbandbreite R überschritten werden.

3.3.2 Erweiterung von List Scheduling

List Scheduling kann als sehr flexible Heuristik leicht an die Erfordernisse der HW-Synthese angepasst werden:

- Aufgaben werden Schritt für Schritt, beginnend beim ersten Zeitschritt, geplant. Zu jedem Zeitschritt hält man für jeden Ressourcetyp r eine Liste K_r von Knoten, deren Vorgänger alle beendet und die selbst noch nicht geplant sind, sowie zwei Listen
  G_r von noch blockierenden Operationen auf einer Ressource vomTyp r,
  B_r von noch laufenden Operationen auf einer Ressource vom Typ r.
  Wir halten ferner noch ein Array R[t] mit initial R[t] = R für jeden Zeitschritt t, das die noch verfügbare Resultatsbandbreite anzeigt.

- Wähle nun zu jedem Ressourcetyp r eine maximale Menge $S_r \subseteq K_r$ mit absteigender Priorität so aus, dass

  - $|S_r| + |G_r| \leq \alpha(r)$
  - $\sum_r o(r) * |S_r| \leq O$
  - $R[t+w(r)] - \sum_{u,w(u)=w(r)} \rho(u) * |S_u| \geq 0$

Erweiterung von List Scheduling -- ff

- Plane alle Knoten $v \in S_r$ zu jeder Ressource r mit $\tau(v) = t$ und
  - nehme sie mit Beendigungszeit $t + w(r)$ nach B_r
  - setze $R[t+w(r)] = R[t+w(r)] - \rho(r)$
  - nehme v mit Freigabezeit $t + \pi(r)$ nach G_r
  - setze die Zeit $t = t + 1$ und
    - nehme dann alle Knoten u mit $\tau(u) + \pi(r) \leq t$ aus den G_r heraus,
    - nehme dann alle Knoten u mit $\tau(u) + w(u) \leq t$ aus den B_r heraus, nehme dann alle Nachfolger v von u, die keine laufenden oder ungeplanten Vorgänger mehr haben nach $K_{\beta(v)}$ hinzu und berechne ihre Priorität.

Erweiterung von List Scheduling -- ff

Anmerkungen:

List Scheduling als Heuristik zur Latenzminimierung unter Ressourcenbeschränkungen lässt sich ohne Effizienzverlust, was seine Laufzeit betrifft, auf die besonderen Gegebenheiten verallgemeinern. Als Prioritäten können nun auch Kräftemodelle oder andere Maße eingesetzt werden.

Man bleibt aber bei diesem Ansatz immer bei einer Heuristik.

Wir wollen als Alternative dazu im Folgenden zwei Ansätze entwickeln, die Optimierung der HW-Synthese über Integer Linear Programming, ILP, durchzuführen.

4.3.3 Zeitschlitzmodelle

Zeitschlitzmodelle basieren auf der Tatsache, dass zu einem gegebenen Sequenzgraphen eine Latenzschranke

$$\underline{L} \geq \max_v (\tau^{ASAP}(v) + min\{ w(u) | (v,u) \in E_R \}) = L_{min}$$

vorgegeben wird. Daraus wird dann ein Integer Programm erstellt, das die Ressourcenkosten unter dieser Latenzschranke minimiert:

Im folgenden seien $l_v := \tau^{ASAP}(v)$ und $h_v := \tau^{ALAP}(v)$ die frühesten und spätesten Startzeiten (Zeitschlitze) für die Aufgabe v.

Für v gibt es demnach $slack(v) + 1 + \underline{L} - L_{min}$ viele Zeitschlitze.

Wir formulieren nun in Abhängigkeit von den Aufgaben und verfügbaren Zeitschlitzen und den verfügbaren Ressourcen ein Integer Programm, dessen Lösung einen Ablaufplan mit minimalen Ressourcenkosten liefert:

Zeitschlitzmodelle: Variablen und Zielfunktion

Variablen:

Wir betrachten zu jeder Aufgabe v, jedem Ressourcentyp u mit $(v,u) \in E_R$ und jedem gültigen Zeitschlitz t von v eine binäre Variable:

- $x_{v,u,t} \in \{0,1\}$ $\quad \forall v \in V_S\ \forall (v,u) \in E_R\ \forall t$ mit $l_v \leq t \leq h_v$:

Für den Ablaufplan bedeutet $x_{v,u,t} = 1 :\Leftrightarrow \tau(v) = t$
und für die Bindung $x_{v,u,t} = 1 :\Rightarrow \beta(v) = u$.
Wir betrachten ferner für jeden Ressourcentyp u eine Integervariable:

- $y_u \in \boldsymbol{N}_0$

Der Wert y_u einer Lösung gibt an, wieviele Ressourcen vom Typ u alloziiert werden müssen. Die Zielfunktion ist demnach bei Kosten $c(u)$ für Ressourcentyp u:

➔ Ziel: Minimiere $\sum_{u \in V_R} c(u) \cdot y_u$

Zeitschlitzmodelle: Das Polytop

Wir müssen nun noch das Polytop über dem die Zielfunktion zu minimieren ist, durch lineare Ungleichungen definieren.

❶ Bindungsbeschränkungen:

Die einfachsten Beschränkungen ergeben sich aus der Forderung, dass jede Aufgabe zu genau einem Zeitpunkt auf genau einer Ressource starten darf:

⇨ $\forall v \in V_S$: $\boxed{\sum_{u,(v,u)\in E_R} \sum_{t=l_v}^{h_v} x_{v,u,t} = 1}$ $\#V_S$ Gleichungen

❷ Datenabhängigkeitsbeschränkungen:

Wir können Laufzeit $d(v)$ und Startzeit $\tau(v)$ leicht durch folgende lineare Terme ausdrücken:

$$d(v) := \sum_{u,(v,u)\in E_R} \sum_{t=l_v}^{h_v} w(u) \cdot x_{v,u,t} \qquad \tau(v) := \sum_{u,(v,u)\in E_R} \sum_{t=l_v}^{h_v} t \cdot x_{v,u,t}$$

⇨ $\forall (v,w) \in E_S$: $\boxed{\tau(w) - \tau(v) \geq d(v)}$ $\#E_S$ Ungleichungen

Zeitschlitzmodelle: Das Polytop -- ff

❸ Zeitbeschränkungen:

Jede Operation muss zum Zeitpunkt $\underline{L}$ abgeschlossen sein. Da wir lineare Terme $d(v)$ und $\tau(v)$ für Delay und Startzeit angeben konnten, führt dies zu folgender einfachen linearen Ungleichung:

⇨ $\forall v \in V_S$: $\boxed{\tau(v) + d(v) \leq \underline{L}}$ $\#V_S$ Ungleichungen

❹ Ressourcenbeschränkungen:

Zu keinem Ressourcentyp u dürfen zu keinem Zeitpunkt mehr als y_u Aufgaben u belegen:

⇨ $\forall u \in V_T \;\; \forall \, min_v\, l_v \leq t \leq max_v\, h_v$:

$$\boxed{\sum_{v,(v,u)\in E_R} \sum_{p=max\{0,t-h_v\}}^{min\{w(u)-1,t-l_v\}} x_{v,u,t-p} \leq y_u}$$

$(max_v\, h_v - min_v\, l_v + 1) * \#V_T$ Ungleichungen

Zeitschlitzmodelle: Komplexität

Variablenzahl:

Wir haben insgesamt $\leq \#E_R * (max_v\, h_v - l_v + 1)$ binäre Variablen und $\#V_T$ viele Integervariablen. Die Gesamtzahl an Variablen hängt somit von der Größe des Ressourcegraphen und von der Zahl der verfügbaren Zeitschlitze ab und beträgt

$$O(\#V_T + \#E_R * (max_v\, h_v - l_v + 1))$$

Ungleichungen:

Als Bilanz an Ungleichungen erhalten wir

$$O(\#V_S + \#E_S + (max_v\, h_v - min_v\, l_v + 1) * \#V_T)$$

Erweiterungen von Zeitschlitzmodellen zur Hardwaresynthese

❹ Ressourcenbeschränkungen bei Modulen mit Pipelining:

Hier ergibt sich nur eine geringfügige Änderung in der Ungleichung, man muss $w(u)$ durch die Periode $\pi(u)$ ersetzen:

⇨ $\forall u \in V_T \;\; \forall \, min_v\, l_v \leq t \leq max_v\, h_v$:

$$\boxed{\sum_{v,(v,u)\in E_R} \sum_{p=max\{0,t-h_v\}}^{min\{\pi(u)-1,t-l_v\}} x_{v,u,t-p} \leq y_u}$$

❺ Bandbreitenbeschränkungen:

Zu jedem Zeitpunkt t dürfen terminierende Aufgaben die Resultatsbandbreite R nicht überschreiten, und startende Aufgaben nicht die Operandenbandbreite O:

⇨ $\forall t, \, min_v\, l_v \leq t \leq max_v\, h_v$:

$$\boxed{\sum_{v} \sum_{u,(v,u)\in E_R} \rho(u) \cdot x_{v,u,t-w(u)} \leq R} \qquad \boxed{\sum_{v} \sum_{u,(v,u)\in E_R} o(u) \cdot x_{v,u,t} \leq O}$$

Erweiterungen von Zeitschlitzmodellen zur Hardwaresynthese

❻ Lebensdauer von Variablen:

Durch Hinzunahme von binären Variablen $z_{v,t}$ für jeden Knoten v und jeden Zeitpunkt t, kann man mit folgenden Ungleichungen mit $z_{v,t} = 1$ anzeigen, dass die mit v assoziierte Variable zum Zeitpunkt t lebt:

⇨ $\forall (v,w) \in E_S$:

$$z_{v,t} \geq \sum_{u,(v,u)\in E_R} \sum_{t'=l_v+w(u)}^{min\{t,h_v+w(u)\}} x_{v,u,t'-w(u)} - \sum_{u,(w,u)\in E_R} \sum_{t'=l_w}^{min\{t,h_w\}} x_{w,u,t'}$$

$z_{v,t}$ muss in der obigen Summe ≥ 1 werden, genau dann, wenn v an eine Ressource u gebunden ist, auf der sie spätestens $w(u)$ Zeitschritte vor t gestartet ist, und für mindestens einen Nachfolger w, dieser bis zum Zeitpunkt t noch nicht gestartet ist. Ansonsten wird durch die Ungleichung nur ≥ 0 oder ≥ -1 gefordert.

Dies ist aber genau die Bedingung dafür, dass das von v produzierte Resultat lebt.

Erweiterungen von Zeitschlitzmodellen zur Hardwaresynthese

❼ Registerzahlminimierung:

Wir führen eine weitere Integer Variable r für die Registerzahl und folgende Ungleichungen für jeden Zeitschritt t ein:

⇨ $\forall t: min_v\, l_v \leq t \leq max_v\, h_v$

$$r \geq \sum_{v\in V} z_{v,t}$$

Ein minimales r, das diese Gleichungen erfüllt, liefert die Registerzahl, die benötigt wird, um alle lebendigen Variablen zu halten. Wir können mit c_r für die Kosten eines Registers durch folgende neue Zielfunktion nun auch die Registerzahl mit einbeziehen:

➔ Ziel: Minimiere $c_r \cdot r + \sum_{u\in V_T} c(u) \cdot y_u$

Zeitschlitzmodelle: Ausblick

Zeitschlitzmodelle sind sehr flexibel und lassen im Hinblick auf eine Synthese in Hardware vielfältige Erweiterungen zu.

So kann man z.B. auch die Option, aufeinanderfolgende Aufgaben (v,w), die an sehr schnelle Ressourcen (a,b) $(w(a)=w(b) = 1)$ gebunden werden können, mit $delay(a) + delay(b) < T$, wo T die Taktperiode ist, zu einer Operation verketten. (Übung)

Der entscheidende Nachteil ist, dass die Zahl der Variablen und die Zahl der Ungleichungen mit dem Schlupf, den man durch die Latenzvorgabe $\underline{L}$ einbaut, aber auch schon mit dem Schlupf, den unkritische Aufgaben haben, wächst.

Gibt es Modelle, in denen die Zeit nicht in die Variablenzahl eingeht?

4.3.4 Flussmodelle

Flussmodelle modellieren als gemischtes lineares Integerprogramm (ganzzahlige + reelwertige Variablen) die Übergabe von Ressourcen an Aufgaben und die Weitergabe einer Ressource von einer an die nächste Aufgabe in Form von Flussbedingungen.

Sie eignen sich zudem sowohl zur Latenz- als auch zur Ressourcenminimierung.

Im folgenden seien wieder $l_v := \tau^{ASAP}(v)$ und $h_v := \tau^{ALAP}(v)$ die frühesten und spätesten Startzeiten für die Aufgabe v.

Wir formulieren nun in Abhängigkeit von den Aufgaben und den verfügbaren Ressourcen ein Integer Programm, dessen Lösung einen Ablaufplan mit minimalen Ressourcenkosten unter Latenzbedingungen oder minimaler Latenz unter Ressourcenbedingungen liefert:

Flussmodelle: Die Variablen

Wir betrachten zu jeder Aufgabe v und jedem Ressourcentyp u mit $(v,u) \in E_R$ zwei binäre Variablen, die Flussquellen und Senken:

- $y_{u,v}, y_{v,u} \in \{0,1\}$ $\quad \forall v \in V_S \; \forall (v,u) \in E_R$;

Für den Ressourcentyp u bedeutet $y_{u,v} = 1 :\Leftrightarrow$ v nimmt sich eine Instanz der Ressource u, ohne sie von einer anderen Aufgabe zu erhalten.

$y_{v,u} = 1$ heißt, dass v eine Instanz der Ressource u benutzt und sie dann wieder zurückgibt, d.h. nicht an andere Aufgaben, die die gleiche Ressource benutzen können weitergibt. In beiden Fällen gilt für die Bindung $\beta(v) = u$.

$2*\#E_R$ Variablen

Flussmodelle: Die Variablen

Zu jedem Paar $v,w \in V_S$ mit $(v,u) \in E_R$ und $(w,u) \in E_R$ betrachten wir die **Flussvariablen**:

- $x_{v,w,u} \in \{0,1\}$ $\quad \forall v,w \in V_S$ *mit* $(v,u) \in E_R$, $(w,u) \in E_R$;

  Die Variable $x_{v,w,u}$ zeigt durch $x_{v,w,u} = 1$ an, dass v eine Ressourceninstanz vom Typ u an die Aufgabe w weitergibt. Die Bindung ist demnach $\beta(v) = \beta(w) = u$ und $\gamma(v) = \gamma(w)$ für $x_{v,w,u} = 1$. $\quad \leq \#V_T * \#V_S^2$ Variablen

Zu jeder Aufgabe $v \in V_S$ betrachten wir schließlich noch eine reellwertige Variable als Startzeit, sowie eine Latenzschranke L, die als Konstante vorgegeben wird, falls Ressourcenminimierung erfolgen soll.

- $t_v \in \boldsymbol{R}_0$, $L \in \boldsymbol{R}_0$ $\quad \leq \#V_S + 1$ Variablen

Zu jedem Ressourcentyp $u \in V_T$ betrachten wir ferner eine Allokationsschranke, die als Konstante vorgegeben wird, falls eine Latenzminimierung erfolgt:

- $a_u \in \boldsymbol{N}_0$ $\quad \#V_T$ Variablen

Flussmodelle: Das Polytop

Wir müssen nun noch das Polytop, über dem die Zielfunktion zu minimieren ist, durch lineare Ungleichungen definieren. Diese klassifizieren wir in

❶ Bindungsbeschränkungen:

Diese Beschränkungen ergeben sich wieder aus der Forderung, dass jede Aufgabe an genau eine Ressourceninstanz gebunden sein muss:

⇨ $\forall v \in V_S$: $\boxed{\sum_{u,(v,u)\in E_R} \left(y_{u,v} + \sum_{w \neq v,(w,u)\in E_R} x_{w,v,u} \right) = 1}$ $\quad \#V_S$ Gleichungen

❷ Datenabhängigkeitsbeschränkungen:

Wir können das Delay $d(v)$ leicht durch folgenden linearen Term ausdrücken:

$$d(v) := \sum_{u,(v,u)\in E_R} w(u) \cdot \left(y_{u,v} + \sum_{w,(w,u)\in E_R} x_{w,v,u} \right)$$

⇨ $\forall (v,w) \in E_S$: $\boxed{t_w - t_v \geq d(v)}$ $\quad \#E_S$ Ungleichungen

Flussmodelle: Das Polytop

❸ Flussbeschränkungen:

Eine Aufgabe kann eine Ressource von ihrer Quelle nehmen oder von einer anderen Aufgabe übernehmen. Sie muss die Ressource weitergeben, oder an ihre Senke abgeben. Es darf keine Ressource erfunden oder einbehalten werden. Diese Bedingung ist eine Art Flusserhaltungsbedingung. Sie muss für jedes Paar $(v,u) \in E_R$ gelten:

⇨ $\forall (v,u) \in E_R$: $\boxed{\underbrace{y_{u,v} + \sum_{w \neq v,(w,u)\in E_R} x_{w,v,u}}_{\leq 1} - \left(y_{v,u} + \sum_{w \neq v,(w,u)\in E_R} x_{v,w,u} \right) = 0}$ $\quad \#E_R$ Gleichungen

Anmerkung: Durch die Bindungsbeschränkungen (❶) ist der Fluss in einen Knoten bezüglich einer Ressource u stets ≤ 1 .

Flussmodelle: Das Polytop -- ff

❹ Latenzbeschränkungen:

Jede Operation muss zum Zeitpunkt L abgeschlossen sein. Betreibt man Latenzminimierung, dann ist L eine Variable. Ansonsten gibt L als Konstante folgende lineare Ungleichungen vor:

⇨ $\forall v \in V_S$: $t_v + d(v) \leq L$ $\#V_S$ Ungleichungen

❺ Ressourcenbeschränkungen:

Zu keinem Ressourcentyp u dürfen zu keinem Zeitpunkt mehr als a_u Aufgaben u belegen. Betreibt man Latenzminimierung, dann liefern die a_u als vorgegebene Konstanten folgende lineare Ungleichungen, sonst als Variablen:

⇨ $\forall u \in V_T$

$$\sum_{v,(v,u)\in E_R} y_{u,v} \leq a_u$$

$\#V_T$ Ungleichungen

Flussmodelle: Das Polytop -- ff

❻ Sequenzialisierungsbeschränkungen:

Wenn sich zwei Aufgaben die gleiche Ressourceninstanz teilen, dürfen sie nicht parallel arbeiten. Gibt v also seine Ressourceninstanz an w weiter, so muss eine entsprechende Zeitbedingung eingehalten werden:

⇨ $\forall v,w \in V_S$ mit $(v,u),(w,u) \in E_R$:

$$t_w - t_v \geq d(v) + (x_{v,w,u} - 1) * \delta(v,w,u)$$

$\#V_T * \#V_S^2$ Ungleichungen

Der Wert $\delta(v,w,u)$ muss so gewählt werden, dass im Falle, dass v seine Ressourceninstanz nicht weitergibt, d.h. $x_{v,w,u} = 0$, eine Lösung für die Startzeitpunkte sicher möglich ist.

Eine gute Wahl ist $\delta(v,w,u) = max\{ w(u) \mid (v,u) \in E_R\} - l_w + h_v$, da dann im Falle $x_{v,w,u} = 0$ die Ungleichung wie folgt lautet:

$$t_w - t_v \geq d(v) - max\{ w(u) \mid (v,u) \in E_R\} + l_w - h_v$$

Flussmodelle: Zielfunktionen

Man kann nun über dem Polytop, das über den Variablen definiert ist, verschiedene lineare Zielfunktionen minimieren:

- Ressourcenkosten

  Die Variablen a_u erfassen über die Ungleichungen zur Ressourcenbeschränkung die Allokationsmöglichkeiten. Die Latenz L wird als Konstante vorgegeben.
  Ziel: Minimiere über dem Polytop $\sum_{u\in V_T} c(u) \cdot a_u$

- Latenz

  Die Allokation wird als Konstante a_u in den Ungleichungen zur Ressourcenbeschränkung vorgegeben, während die Latenz L als Variable bleibt.
  Ziel: Minimiere über dem Polytop L

- Linearkombination von Ressourcenkosten und Latenz

  a_u und L bleiben Variablen.
  Ziel: Minimiere über dem Polytop $c_L \cdot L + \sum_{u\in V_T} c(u) \cdot a_u$

Flussmodelle: Komplexität

Variablenzahl:

Wir haben insgesamt $\leq 2*\#E_R + \#V_S + 1 + \#V_T + \#V_T * \#V_S^2$ Variablen. Die Gesamtzahl an Variablen hängt somit von der Zahl der Ressourcentypen und von der Zahl der Aufgaben ab, ist aber unabhängig von der Zahl der Zeitschlitze. Die Schranke ist, wenn es wenig universelle Ressourcen und viele verschiedene Aufgaben gibt, sehr viel niedriger. Sie erreicht den höchsten Wert nur, wenn jede Aufgabe auf jedem Ressourcentyp ausgeführt werden kann.

Ungleichungen:

Als Bilanz an Gleichungen und Ungleichungen erhalten wir

$$\leq \#E_R + \#V_S + \#V_T + \#V_T * \#V_S^2$$

Es gilt das Gleiche, das zu der Variablenzahl schon gesagt wurde.

Man kann nun versuchen, das Integerprogramm durch weitere Analysen zu vereinfachen. Wir wollen dies hier nicht weiter verfolgen.