C-016

多様な資源を対象とした多資源非同期式アービタの一構成
A asynchronous arbiter for multiple resources with various functions

近藤真史 † 大山国晃 † 横川智教 ‡ 佐藤洋一郎 ‡
Masafumi KONDO Kuniaki OHYAMA Tomoyuki YOKOGAWA Yoichiro SATO

1. まえがき

近年，計算機システム等においては，高性能化に伴って，複数の共有資源の一部を指定して使用する場合が増えている [1]．このような状況下で，複数の装置から発せられる資源の使用要求を調停する回路をアービタという．これまでにアービタの構成法が種々提案されているが，大多数は単一資源用で [2]，多資源用の報告もあるが [3]，使用する資源を指定できない．

そこで筆者等は，複数の資源を適宜組み合わせて使用するという前提で，各装置から発せられる資源の使用要求の競合を処理するための機構について検討しており，その構成法について報告する．

2. 提案する競合処理機構の構成と動作

競合処理機構 ($ArbitrationUnit$) の構成を図 1 に示す．$R_j(1 \leq j \leq m)$ は資源であり，$P_i(1 \leq i \leq n)$ は資源の使用要求を発する装置である．P_i は，要求信号 Req_i と要求する資源の組み合わせを表す信号 $Addr_i$ を競合処理機構にアサートし，その承認信号 Ack_i がアサートされることで資源の使用権を得る．その後，資源の使用が完了すると，Req_i をネゲートする．すなわち，Req_i のアサート及びネゲートを，それぞれ資源の使用要求 (以下，単に使用要求という) 及びそれまで使用していた資源の解放要求 (以下，単に解放要求という) の発生を意味する．$StateRegister(SR)$ は，資源の利用状況を表す m ビットのレジスタであり，その j ビット目 $SR(j)$ が 1 ならば R_j が使用されていることを示す．すなわち，$Addr_i$ の j ビット目を $Addr_i(j)$ とすると，$E_i = \bigvee_{j=1}^{m} Addr_i(j) \cdot SR(j)$ が 0 のときのみ P_i からの使用要求が承認可能となる．さらに，$Arbiter$ は，後述する $Controller$ で生成される要求信号 (Set_i, $Reset_i$) の中からただ 1 つを選択し，承認信号 $Grant$ をアサートする単一資源用非同期式アービタである．$Controller$ は，$Arbiter$ への使用要求信号 Set_i や解放要求信号 $Reset_i$ の生成，P_i への承認信号 Ack_i の生成，SR の更新を行う制御回路である．

図 1 の基本動作は以下の通りである．Req_i がアサート及びネゲートされると直ちに，それぞれ Set_i 及び $Reset_i$ をアサートする．$Arbiter$ における競合処理の結果，$Grant_l$ がアサートされたとする．$Set_l = 1(Reset_l = 0)$ の場合，$E_l = 0$ ならば資源を使用可能であるので，Ack_l をアサートすると共に，$Addr_l$ にしたがって SR を更新する ($Addr_l(j) = 1$ であるすべての j に対して $SR(j) = 1$ とする)．さらに，次の競合処理を開始するために，Set_l をネゲートする．また，$E_l = 1$ ならば資源を利用できないので，次の競合処理

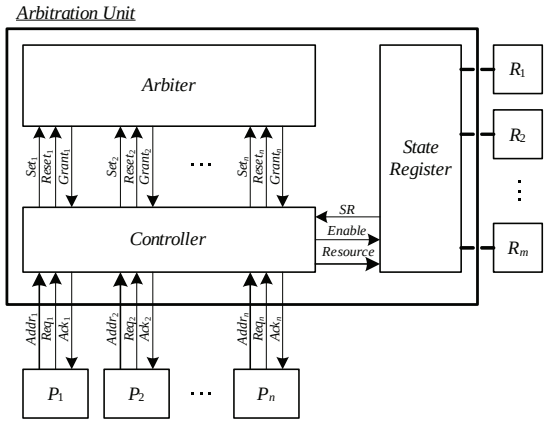

図 1: 競合処理機構

を開始するために直ちに Set_l をネゲートする．一方，$Reset_l = 1(Set_l = 0)$ の場合，Ack_l をネゲートすると共に $Reset_i$ をネゲートし，$Addr_l$ にしたがって SR を更新する ($Addr_l(j) = 1$ であるすべての j に対して $SR(j) = 0$ とする)．

3. $Controller$ の動作仕様

Req_i がアサートされて，$Arbiter$ で承認されるが，資源を使用できないとき ($E_i = 1$) のタイミングチャートを図 2 に示す．図 2 では，$Grant_i$ がネゲートされた直後

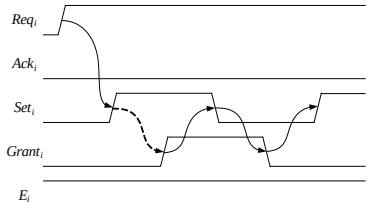

図 2: $E_i = 1$ のときのタイミングチャート

に Set_i を再びアサートしている．これは，次に $Grant_i$ がアサートされるまでの間に，使用したい資源が解放される ($E_i = 0$) 可能性があるためである．

次に，資源を使用できるときの承認動作のタイミングチャートを図 3 に示す．このとき，SR の内容は $Enable$ のアサートにしたがって信号 $Resource$ に更新される．このときの $Resource$ は，その j ビット目を $Resource(j)$ とすると，$Resource(j) = SR(j) \cdot Addr_i(j)$ となる．このように，Ack_i をアサートすると共に Set_i をネゲートすることにより，$\bigwedge_{i=1}^{n} E_i = 1$ となるまで，承認動作が行われることになる．

最後に，資源の解放，すなわち資源使用終了時のタ

†岡山県立大学大学院 情報系工学研究科
‡岡山県立大学 情報工学部

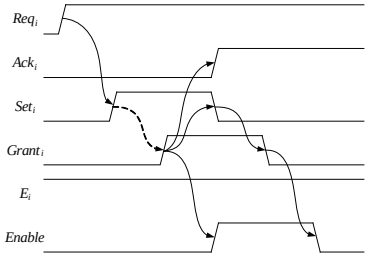

図 3: $E_i = 1$ のときの承認動作のタイミングチャート

イミングチャートを図 4 に示す．このときの $Resource$

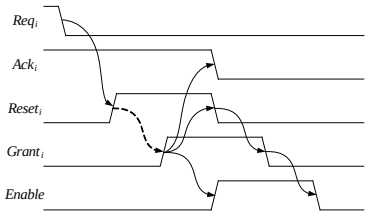

図 4: 資源の使用終了時のタイミングチャート

は，$Resource(j) = SR(j) \cdot \overline{Addr_i(j)}$ となる．この場合，Req_i のネゲートと共に $Addr_i$ もネゲートされるので $Resource$ の情報を記憶する必要がある．

4. $Arbiter$ の構造

統合型準優先アービタ (以下，単に統合型という) の構造を図 5 に示す．PTM は，優先順位付の競合処理機能を付加したツリーモジュールであり，その構成と動作は以下のとおりである．PTM の構成を図 6 に示す．PTM では，Set 及び $Reset$ をそれぞれ 2 要求ずつ，計 4 つの要求における競合を処理する．ただし，$iSet1$ と $iReset1$，$iSet2$ と $iReset2$ はそれぞれ排他的に生起するため，これらに関する競合は存在しない．このため，実際に処理する競合は，**(I)**$iSet1$-$iSet2$，$iReset1$-$iReset2$，**(II)**$iSet1$-$iReset2$，$iSet2$-$iReset1$ の 2 種 4 パターンとなる．ここで，**I** は公平に競合を処理し，**II** は $Reset$ を優先して承認すべきである．そこで PTM は，前者を文献 [2] のモジュール TM により処理し，後者に対しては 4 入力の優先順位付競合処理モジュール

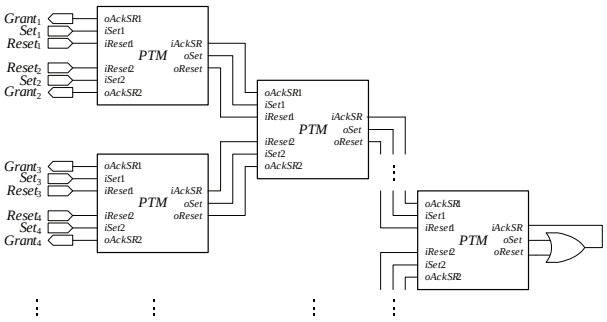

図 5: アービタの構造

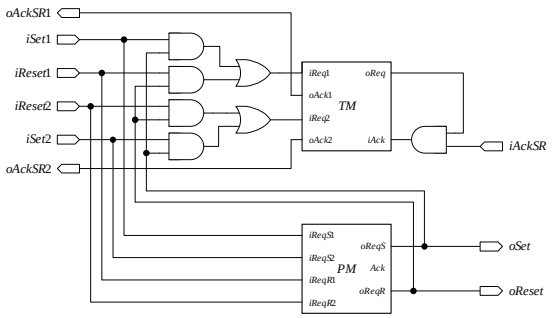

図 6: PM の構成

($PriorityModule$:PM) を設ける．

I の要求が生起している場合は TM によって競合が処理されるが，ME でのメタステーブル動作に起因して TM の処理時間が増大することとなる．一方，このとき PM では，**I** の競合は解決されないものの，Set と $Reset$ のいずれか一方が一意に承認されている．この点に着目すると，PM の承認を次段の PTM への要求として先行してアサートすることで，TM でのメタステーブル動作の終了を待つことなく，後続の PTM は競合処理を開始することができる．そして，すべての競合処理が終了して $iAckSR$ が伝播すると，メタステーブル動作の終了を確認した上で TM での競合処理結果を前段の PTM へ伝播させる．また，PM の承認結果を基に TM への要求を選択することで，**II** の要求が生起している際には，Set と $Reset$ のいずれか一方のみが TM にアサートされて一意に承認される．

以上より，Set 及び $Reset$ は，2 要求毎に PTM にアサートされ，優先順位に基づいた競合処理が段階的に行われる．最後段の PTM においてすべての競合処理が終了すると，1 つでも $Reset$ が生起しているならば，その出力 $oSet$，それ以外ならば $oReset$ が出力される．そして，これらの論理和を $iAckSR$ としてアサートすることで，各段の PTM での競合に勝った要求に対応した承認が伝播し，最前段の PTM において Set 及び $Reset$ のうちただ 1 つが承認されることとなる．

5. あとがき

本稿では，柔軟な資源の使用を可能とする多資源用競合処理機構の構成を与えた．今後，$Controller$ や $Arbiter$ の詳細設計を行う予定である．

参考文献

[1] M.KONDO et al.:”A ring segmented bus architecture for Globally Asynchronous Locally Synchronous System”, ICECSD 2009, pp.596-601(Jun.2009).

[2] M.IMAI et al.:”N-way Ring and Square Arbiters”, ICCD 2009, pp.125-130 (Oct.2009).

[3] 増山博ほか:”非同期アービタの多資源化における一考察”, 信学論 D, Vol.J67-D, No.10, pp.1258-1265 (1984-10).