Università di Pisa

Dipartimento di Ingegneria dell'Informazione

Laurea Triennale in Ingegneria Informatica

Analisi di un Limitatore di Banda per il Bus AXI4

Relatori:
Prof. Giovanni Stea
Prof. Alessandro Biondi

Candidato:
Jacopo Del Granchio

Anno Accademico 2023/2024

Sommario

Garantire fairness e prestazioni per dispositivi real-time critici, a livello di bus AXI4, è un problema complesso dovuto sia alla complessità del protocollo AXI4 stesso sia al sottostante problema di scheduling.

Una soluzione a questo problema è l'unità AXI-REALM, composta da tre componenti principali – Splitter, Buffer e Isolate – che lavorano insieme per limitare la banda e garantire l'isolamento temporale tra i dispositivi, viene analizzata in dettaglio.

Studiando il comportamento al variare dei parametri di configurazione, verrà messo in luce il comportamento non lineare e l'intrinseco errore nella limitazione commesso dall'unità. Lo studio viene realizzato modellando il traffico in ingresso come processo aleatorio e lo stato dei componenti interni come catene di Markov, da cui è possibile ricavare utili proprietà sui componenti.

Viene implementata una struttura alternativa, più modulare rispetto alla soluzione iniziale, che permette la gestione di dispositivi con più porte e l'implementazione di un algoritmo basato sul Surplus Round Robin. La nuova implementazione elimina le problematiche rivelate nell'analisi e migliora l'utilizzazione complessiva realizzabile, risultando allo stesso tempo più semplice da configurare e adattabile a contesti real-time.

In conclusione, si discutono i possibili sviluppi futuri, tra cui l'integrazione con l'interconnect per ulteriori miglioramenti.

Indice

1 Introduzione

1.1 Introduzione al Bus AXI

Il bus AXI4[2] è altamente complesso, ma al fine di questa analisi è sufficiente considerare il protocollo di handshake, e in particolare, concentrarsi sulle scritture. AXI4 prevede 5 canali dedicati ognuno dei quali viene controllato tramite due segnali `VALID` e `READY`. Chi invia imposta il segnale `VALID` a 1 solamente se i dati sono validi, mentre chi riceve imposta il segnale `READY` a 1 solamente se è pronto a ricevere i dati. Quando entrambi i segnali sono ad 1 viene effettuata la transazione. Il protocollo AXI4 non impone alcun vincolo sull'ordine con cui `VALID` e `READY` devono essere controllati e dipendono solamente dal proprio stato.

1.1.1 Anatomia di una scrittura

Per scrivere si utilizzano 3 canali: AW, W e B. Il Manager inizia una transazione di scrittura comunicando al Subordinate l'indirizzo e la lunghezza della scrittura tramite AW, una scrittura di questo tipo è detta burst. In parallelo sul canale W vengono inviate N transazioni, dette beat, con N pari alla lunghezza precedentemente specificata. L'ultimo beat di una transazione è indicato tramite un segnale `LAST` apposito. Il Subordinate, completate tutte le scritture, comunica l'esito tramite il canale B. Al fine della limitazione in banda è sufficiente concentrarsi sul solo canale W dove avviene la vera e propria limitazione.

La Figura 1 mostra un esempio di traffico sul bus AXI4 e l'associata visione semplificata che sarà utilizzata negli altri diagrammi.

1.1.2 Idealità di una transazione

In una transazione generica ci possono essere arbitrari cicli di attesa, considerando una scrittura, sia tra la transazione su AW e il primo beat su W, sia tra beat stessi su W, sia tra l'ultimo beat su W e la risposta su B.

Una transazione senza alcuno stato di attesa è detta ideale. Una transazione ideale è possibile solamente se la sorgente e la destinazione rispondono senza cicli di attesa.

1.2 Criticità in sistemi Eterogenei

1.2.1 Fairness

Il concetto di fairness si riferisce alla capacità del sistema di distribuire in modo equo la banda tra i vari dispositivi connessi al bus AXI4. In molti scenari reali, infatti come analizzato in [10], le transazioni dei diversi master possono avere

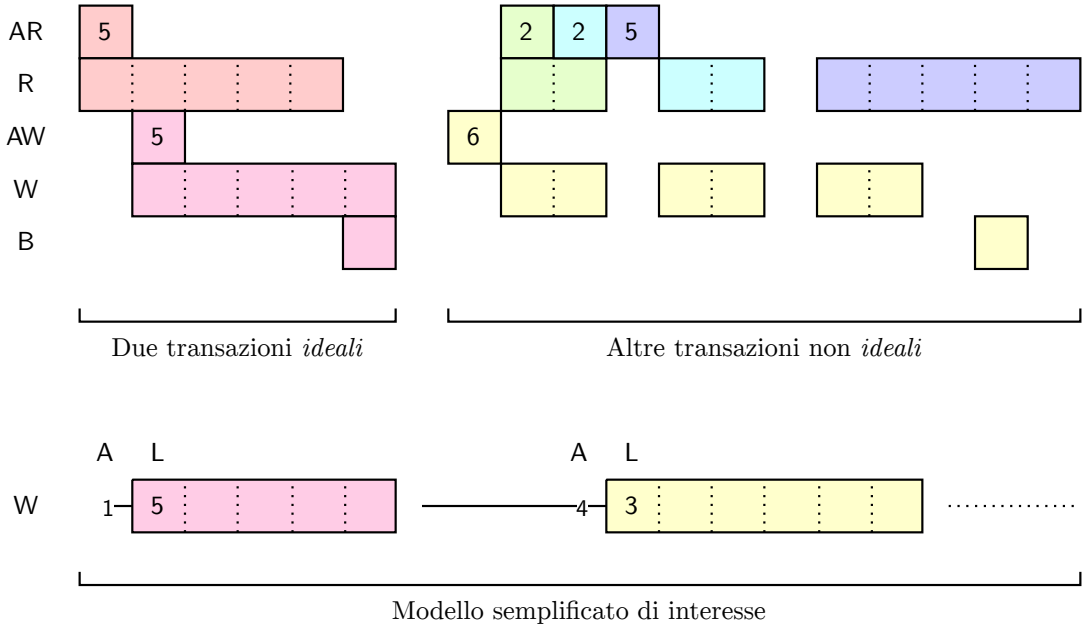

Figura 1: Il bus AXI4 e un equivalente modello semplificato.

lunghezze molto disparate (ad esempio, 8 contro 256 beat). Tale disparità porta, come evidenziato nel paper B1, a una situazione in cui i dispositivi con transazioni più lunghe possono monopolizzare il bus, ottenendo una quota sproporzionata di banda rispetto a quelli con transazioni più brevi.

Questa mancanza di equità si traduce in un'assegnazione non bilanciata delle risorse: il traffico di dispositivi con transazioni più lunghe rischia di saturare il bus, penalizzando quelli che inviano transazioni brevi.

1.2.2 Isolamento Temporale

L'isolamento temporale è essenziale per assicurare che, in ambienti con elevata concorrenza, ogni dispositivo possa accedere al bus senza interferenze provenienti dagli altri.

In assenza di un adeguato isolamento, come analizzato in [5], le transazioni di un master possono essere ritardate dall'attività di altri, compromettendo la prevedibilità e la tempestività dei tempi di risposta. Questo è particolarmente critico per dispositivi real-time che necessitano di risposte entro tempi noti.

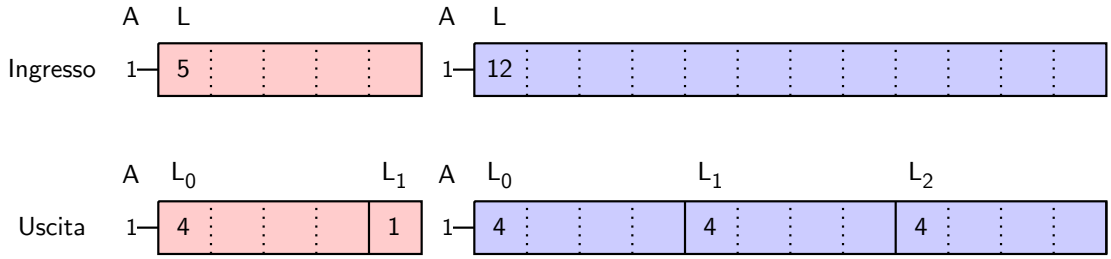

Figura 2: Comportamento dello Splitter con limite pari a 4.

1.3 Soluzione proposta: AXI-REALM

La soluzione qui riportata corrisponde a quella implementata in [5], basata a sua volta su [10], di cui si ignora per semplicità la logica legata all'interfaccia di configurazione, assumendo di poter direttamente inserire i valori nei vari moduli. L'AXI-REALM si interpone tra ogni manager e l'interconnect e prevede una catena di 3 componenti che globalmente realizzano il comportamento desiderato: Splitter, Buffer e Isolate a cui è associata la logica di limitazione.

1.3.1 Splitter

Lo Splitter è un componente semplice, che data una lunghezza limite K, suddivide ogni transazione in ingresso con lunghezza L in più transazioni di lunghezza massima K. La presenza dello splitter ci permette di rendere fair l'interconnect a cui l'AXI-REALM è connesso, evitando che un dispositivo che effettua solamente transazioni molto lunghe possa mandare in starvation altri dispositivi. Inoltre limita la lunghezza delle transazioni a un valore noto che permette ai componenti sottostanti di essere ottimizzati e permette alcune osservazioni sul comportamento globale.

1.3.2 Buffer

Il Buffer è il secondo componente e memorizza i canali AW e W finché non riceve l'ultimo beat su W, e solo a quel punto invia la transazione in uscita. La presenza del Buffer elimina eventuali cicli di attesa presenti nel traffico in ingresso, rendendo ideali le transazioni in uscita. Questa proprietà ci permetterà di poter garantire utilizzazione massima e semplifica notevolmente l'implementazione della logica di limitazione.

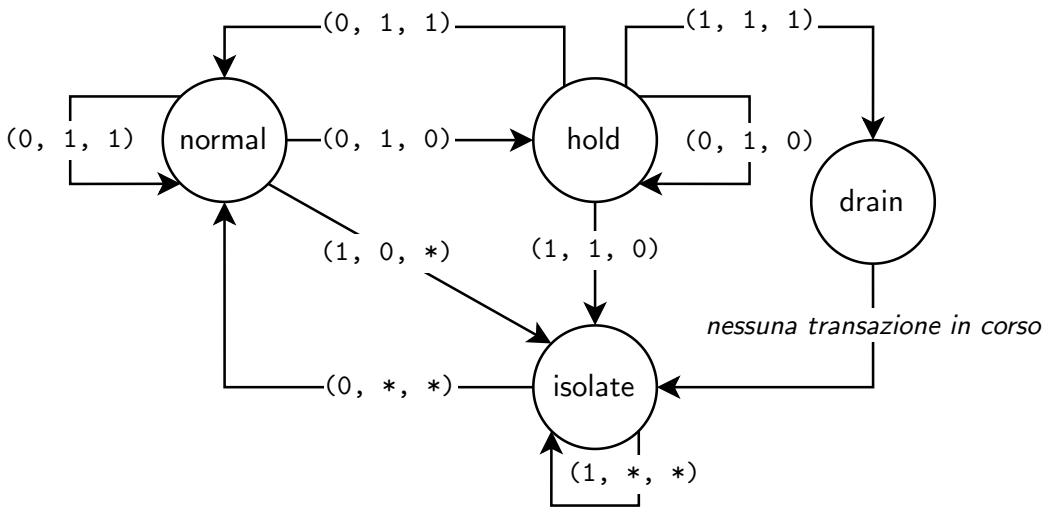

Figura 3: Macchina a stati dell'Isolate, ogni arco riporta la 3-upla dei segnali (`ISOLATE`, `VALID`, `READY`).

1.3.3 Isolate e Logica di Limitazione

L'Isolate è il componente che si occupa di bloccare i canali una volta terminato il budget. È controllata da un singolo segnale detto `ISOLATE` guidato dalla logica associata.

Quando viene ricevuto il segnale di isolamento non è sempre possibile bloccare istantaneamente il canale: se è stata effettuata una transazione su AW, le corrispettive transazioni su W dovranno terminare prima di poter isolare il dispositivo. La macchina a stati finiti nella figura 3 riporta gli stati necessari per implementare questo comportamento.

L'associata logica di limitazione, dato un budget e un periodo, aggiorna il budget rimanente all'arrivo di ogni transazione su AW, grazie alla presenza del segnale `LENGTH` indicante il numero di beat su W. Inoltre allo scadere del periodo configurato viene ripristinato il budget rimanente al valore iniziale configurato.

Il segnale di controllo segue un comportamento molto semplice permettendo il passaggio delle transazioni (`ISOLATE`=0) fintanto che al loro arrivo il budget rimanente è maggiore di zero, e le blocca (`ISOLATE`=1) invece quando il budget risulta inferiore o uguale a zero.

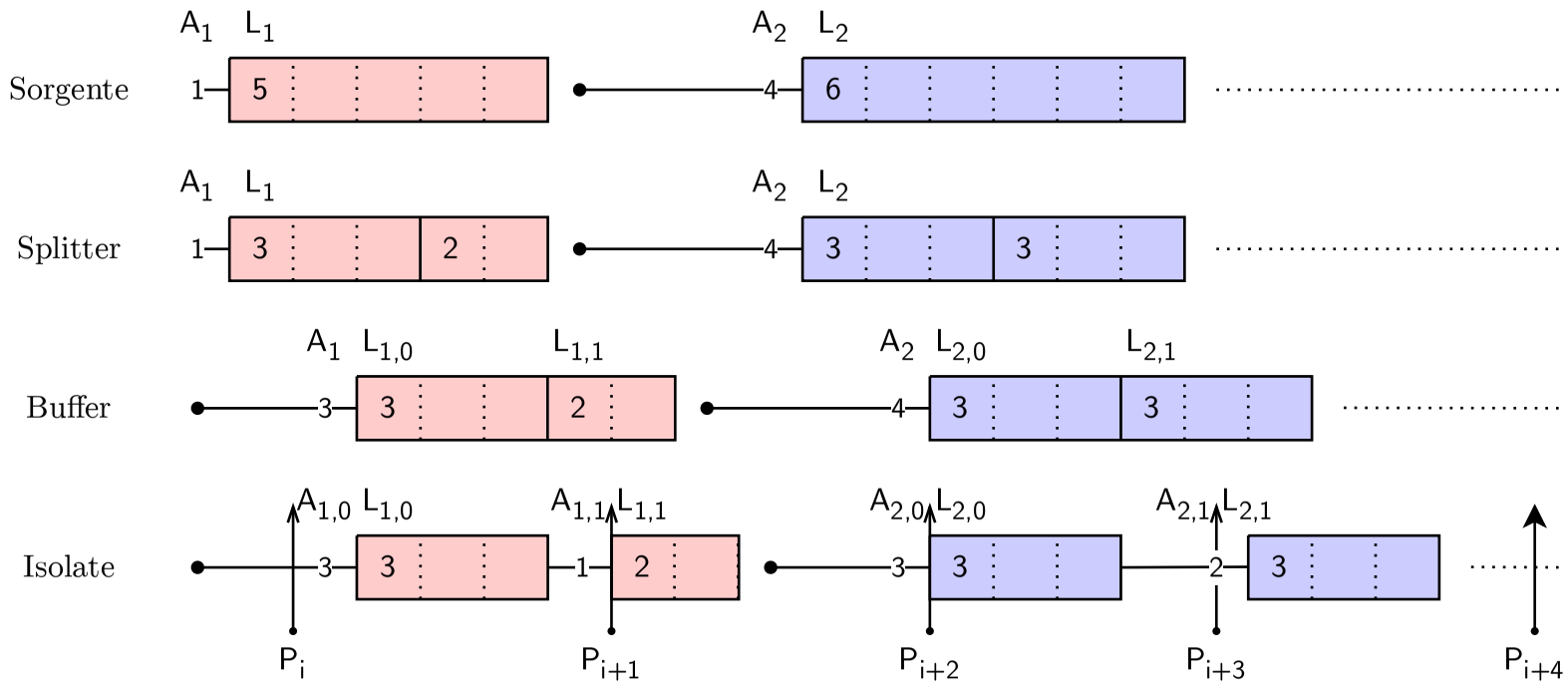

Figura 4: Comportamento dell'AXI-REALM.
Lo Splitter è configurato con limite pari a 3 e l'Isolate con budget di 1 e periodo di 5.

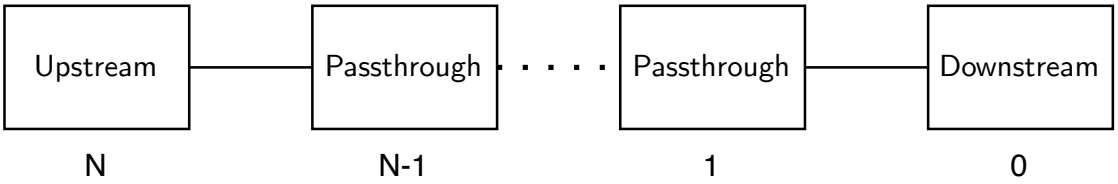

Figura 5: Modello di un generico sistema AXI4.

2 Analisi della Configurazione

2.1 Concetti Generali

Come anticipato nella sezione 1.1 di introduzione al bus AXI4, l'analisi si concentra in particolare sul singolo canale W, dove i componenti descritti influenzano di più il traffico.

Modello delle Transazioni Nelle prossime sezioni sarà necessario poter descrivere il comportamento delle unità per via matematica e statistica. Il modello adottato assume una sorgente e destinazione ideale e prevede una sequenza di transazioni T_i caratterizzate da una singola attesa A_i seguita da un numero arbitrario di lunghezze $L_{i,0}, L_{i,1}, \cdots, L_{i,N}$. Questi campi sono riportati in figura 4.

Questo modello non è completamente generico, infatti non permette di modellare l'uscita del componente Isolate, ma è sufficiente per modellare il traffico in ingresso del sistema. Dove necessario sarà utilizzato un modello equivalente di traffico in uscita, costruito raggruppando le attese all'inizio della transazione.

Infine considereremo un sistema generico come quello in figura 5 e calcoleremo il ritardo che questi introducono nel sistema.

2.1.1 Ritardo

Nel modello sopra descritto introduciamo i concetti di ritardo R_i^n che rappresenta i cicli introdotti dal componente n che la transazione i dovrà attendere. Inoltre a seconda dei casi sarà più comodo utilizzare direttamente il ritardo cumulato C_i^n introdotto dal componente n alla i-esima transazione. Questi due concetti sono legati dalla seguente relazione:

$$C_i^n = \sum_{j=1}^{i} R_j^n \tag{1}$$

e pertanto vale anche:

$$R_i^n = C_i^n - C_{i-1}^n \tag{2}$$

Considerando una serie di componenti come in figura 5, il ritardo (e analogamente il ritardo cumulato) si può esprimere come somma del ritardo introdotto da ogni componente, semplificando il ritardo della sorgente e della destinazione che si può considerare nullo per ipotesi di idealità:

$$\begin{aligned} R_i &= R_i^n + R_i^{n-1} + \cdots + R_i^0 \\ &= R_i^{n-1} + \cdots + R_i^1 \end{aligned}$$

Infine il ritardo cumulato C_i^n, con i sufficientemente grande da poter assumere il componente a regime, si può approssimare come una retta:

$$C_i^n \approx \bar{R}^n \cdot (i - K^n) + C_K^n \tag{3}$$

dove si indica con K^n il numero di transazioni necessarie per portare a regime il componente n e con $\bar{R}^n$ il ritardo medio a regime del componente n. Considerando una serie di componenti è sufficiente prendere il singolo K^j più grande per poter modellare l'intero sistema nello stesso modo.

Questa approssimazione risulta particolarmente utile per aiutare l'interpretazione dei risultati che seguono.

2.1.2 Banda

Per banda alla transazione i si indica il rapporto tra lunghezze e tempo necessario a svolgerle:

$$B_i = \frac{\sum_j^i L_j}{\sum_j^i L_j + A_j + R_j} \tag{4}$$

Nonostante sia possibile esprimere la banda in byte al secondo, non è utile ai nostri fini: in un canale AXI4 ogni transazione può o meno utilizzare l'intera larghezza del bus (tramite il segnale `SIZE` e ulteriori segnali per specificare la parte utile). Pertanto se esprimessimo il vincolo in byte al secondo non avremmo modo di garantire in realtà alcuna banda senza fare assunzioni stringenti sul traffico in ingresso. Per esempio, in un canale da 32 bit, un dispositivo configurato con 4 byte ogni 4 cicli potrebbe sia utilizzare un quarto della banda (1 transazione da 4 byte) o utilizzarla tutta (4 transazioni da 1 byte).

2.1.3 Modello aleatorio

Per calcolare i valori attesi data una particolare configurazione possiamo modellare il traffico come un processo aleatorio formato dalla ripetizione dalla sequenza di

una variabile aleatoria indipendente[1] rappresentante una singola transazione come descritta all'inizio di questa sezione.

È possibile modellare i componenti dell'AXI-REALM come catene di Markov dalle quali risulta di particolare interesse la distribuzione stazionaria: sia per calcolare valori come il ritardo medio, sia per calcolare la distribuzione del traffico in uscita.

Ogni catena di Markov è scomponibile in parte transiente e un numero di classi irriducibili per le quali è definita la distribuzione stazionaria π. A differenza del caso completamente generico lo stato iniziale è unico e noto: lo stato in cui il componente si porta una volta ricevuto il segnale `RESET`. Pertanto costruiremo le catene in modo che siano completamente raggiungibili dallo stato iniziale e assumeremo che esista una singola classe irriducibile. Anche se esistesse una configurazione tale da generare molteplici classi irriducibili, essa risulterebbe inutilizzabile in pratica: la presenza di più classi indicherebbe che il componente si porterebbe, in base alla specifica sequenza ricevuta, in più stati di regime con proprietà potenzialmente diverse.

Infine reinterpretando nuovamente l'equazione (3) il K a cui fa riferimento non è altro che il numero di transazioni necessario per passare dallo stato iniziale, potenzialmente transiente, alla classe irriducibile sommato al tempo di mixing per arrivare alla distribuzione stazionaria.

2.2 Splitter

2.2.1 Ritardo

Lo Splitter non ha stato e non introduce alcun ciclo di attesa, pertanto il ritardo è identicamente nullo.

2.2.2 Banda

Nel caso di una destinazione non ideale, modellabile con un R_i^0 non nullo, si nota come lo splitter causi una notevole riduzione della banda disponibile. Per esempio se $A_i = 0$, $R_i^0 = 4$, $L_i = 16$ e $\mathit{limite} = 4$ allora la banda passa da $B = \frac{16}{16+4} = \frac{4}{5}$ a $B' = \frac{4}{4+4} = \frac{1}{2}$.

Al variare del limite e a parità di ingresso varia notevolmente il numero di transazioni effettuate. Questa variazione, se associata al ritardo non nullo, può portare a una notevole riduzione della banda disponibile. Per esempio se il ritardo R_i fosse pari a uno e avessimo transazioni in ingresso di lunghezza sempre pari a 8 che limitiamo a 2, la banda disponibile passa dal $\frac{8}{8+1} \approx 89\%$ al $\frac{2}{2+1} \approx 67\%$ riducendo così la banda del 22% circa.

[1] Non tutto il traffico risulta modellabile esattamente, nonostante ciò l'approssimazione rivela proprietà interessanti sui componenti analizzati.

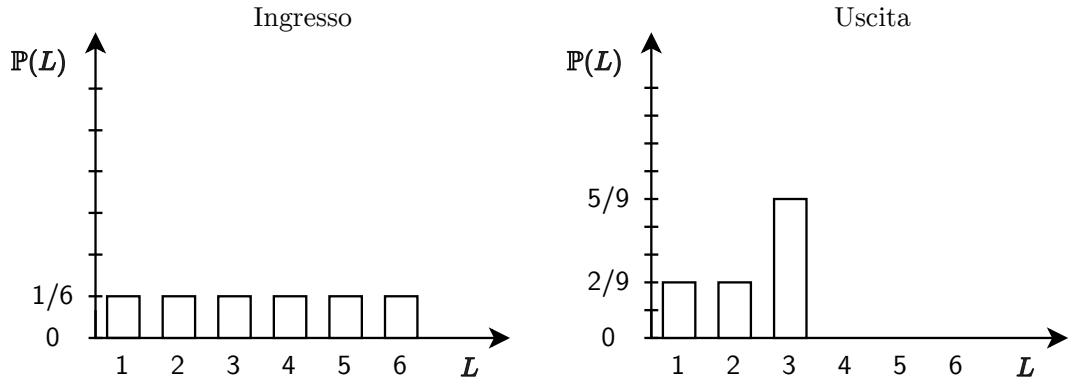

Figura 6: Esempio di distribuzione prima e dopo uno splitter con limite pari a 3.

Nel caso dell'unità AXI-REALM questo è un problema relativo, considerando che generalmente andremo a introdurre ritardo per limitare la banda, sarà sufficiente tener conto di questo ritardo aggiuntivo in fase di configurazione. Nonostante ciò in alcuni casi, soprattutto di grandi limiti, potrebbe non essere possibile aggirare questa limitazione.

2.2.3 Distribuzione in Uscita

Il parametro di limite ha un effetto non trascurabile sulla distribuzione del traffico, in figura 6 si può osservare un esempio con limite pari a 3.

2.3 Buffer

2.3.1 Ritardo

Modellare lo stato del buffer non è immediato: il numero di elementi nel buffer non è sufficiente. Assumendo il buffer sufficientemente grande e considerando attese nulle fittizie per transazioni con più di una lunghezza otteniamo:

- Caso base, buffer vuoto:
  Per definizione il ritardo cumulato $C_1 = R_1$ è pari a $L_1 - 1$ in quanto dovremmo aspettare di avere in ingresso l'ultimo beat della transazione. Un eventuale prima attesa A_1 è ininfluente dato che possiamo semplicemente spostare il riferimento in avanti di A_1 e considerare una transazione in ingresso con $A_1 = 0$.
- Passo induttivo:
  Il buffer è non vuoto e ha ricevuto una transazione A_{i-1}, L_{i-1} e riceve A_i, L_i

allora possiamo calcolare il ritardo cumulato della seconda osservando che prendendo come riferimento la fine all'ingresso della transazione precedente, la transazione precedente terminerà all'uscita dopo C_{i-1} per definizione e la transazione corrente sarà all'ultimo beat tra $A_i + L_i - 1$ allora il ritardo cumulato della transazione corrente risulta il tempo, se positivo, da attendere per far liberare l'uscita sommato al ritardo dovuto all'attesa dell'ultimo beat della transazione corrente:

$$\begin{aligned} C_i &= \max(C_{i-1} - (A_i + L_i - 1),\ 0) + L_i - 1 \\ &= \max(C_{i-1} - (A_i + L_i - 1) + (L_i - 1),\ L_i - 1) \\ &= \max(C_{i-1} - A_i,\ L_i - 1) \end{aligned} \tag{5}$$

Avendo ottenuto un'espressione ricorsiva nel ritardo cumulato possiamo modellare lo stato del buffer tramite il ritardo cumulato stesso. Inoltre se poniamo $C_0 = 0$ e applichiamo la formula (5) trovata per l'ennesima transazione otteniamo

$$C_1 = \max(C_0 - A_1,\ L_1 - 1) = \max(0 - A_1,\ L_1 - 1) = L_1 - 1$$

che coincide con quanto trovato inizialmente. Questa osservazione giustifica la naturale rappresentazione dello stato di buffer vuoto come stato di ritardo cumulato nullo.

Possiamo notare che l'espressione trovata pone un limite inferiore al ritardo pari a $L_i - 1$, indipendentemente dallo stato del buffer. Notiamo anche che se tutte le attese sono nulle[2] allora il ritardo cumulato:

$$\begin{aligned} C_i &= \max(C_{i-1},\ L_i - 1) \\ &= \max(\max(C_{i-2},\ L_{i-1} - 1),\ L_i - 1) = \max(C_{i-2},\ L_{i-1} - 1,\ L_i - 1) \\ &= \max(L_1 - 1,\ \cdots,\ L_{i-1} - 1,\ L_i - 1) \\ &= \max(L_1,\ \cdots,\ L_{i-1},\ L_i) - 1 \end{aligned} \tag{6}$$

risulta semplicemente pari a quello della transazione più grande fino a quella corrente.

Questa osservazione ci permette di esprimere il ritardo cumulato per una transazione generalizzata con più lunghezze: $C_i = \max(C_{i-1} - A_i,\ \max_j(L_{i,j}) - 1)$

L'attesa sommata al ritardo risulta pertanto:

$$\begin{aligned} R_i + A_i &= C_i - C_{i-1} + A_i \\ &= \max(C_{i-1} - A_i,\ L_i - 1) - C_{i-1} + A_i \\ &= \max(0,\ (L_i - 1) + A_i - C_{i-1}) \end{aligned} \tag{7}$$

[2]Caso di nostro interesse dato che possiamo assumere che stiamo limitando la banda e pertanto avremo traffico con pochissime attese.

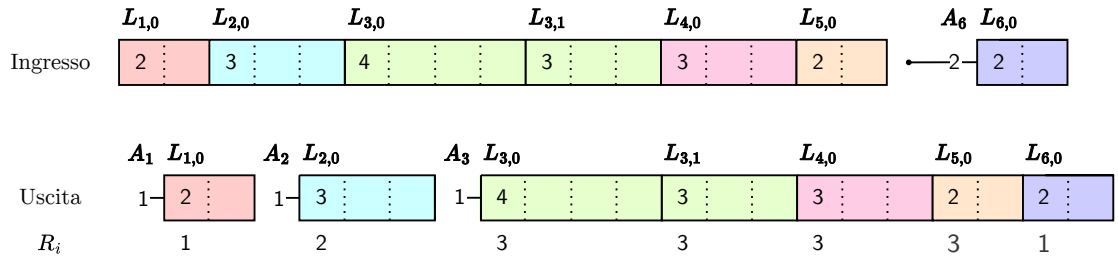

Figura 7: Ritardo introdotto dal Buffer in un traffico di esempio.

interpretabile come, qualora positivo, il tempo necessario alla transazione corrente per essere pronta all'invio ($(L_i - 1) + A_i$) a cui si sottrae il tempo necessario alla transazione precedente per terminare (C_{i-1}).

2.3.2 Modello Aleatorio

La catena di Markov del Buffer segue da quanto detto nel capitolo dedicato e dalle considerazioni sullo stato del Buffer fatte nella sezione precedente. Una volta calcolata la distribuzione stazionaria a regime risulta possibile calcolare il valore atteso del ritardo cumulato:

$$\mathbb{E}[C_i] = \pi \cdot \Big(0,\ 1,\ \cdots,\ \max_{\forall i}(L_i) - 2,\ \max_{\forall i}(L_i) - 1)\Big)$$

assumendo che gli stati siano stati ordinati per valore crescente. Ricercando la scomposizione dell'equazione (3) invece risulta per i sufficientemente grandi:

$$\mathbb{E}[R_i] = \mathbb{E}[C_i] - \mathbb{E}[C_{i-1}] = 0$$

questa proprietà dimostra come il Buffer non comporti riduzione della banda a disposizione.

Infine, fintanto che le attese sono completamente nulle, il ritardo e dunque lo stato, può solo crescere, implicando che uno stato S avrà archi solamente verso stati maggiori di S. Pertanto il singolo stato a ritardo massimo forma una classe irriducibile e ogni altro stato è transiente. In questo caso risulta per i sufficientemente grandi:

$$\mathbb{E}[C_i] = \mathbb{E}[C_K] = (1) \cdot (\max(L_i) - 1) = \max(L_i) - 1$$

dove si indica con K il primo passo in cui il sistema è considerabile a regime.

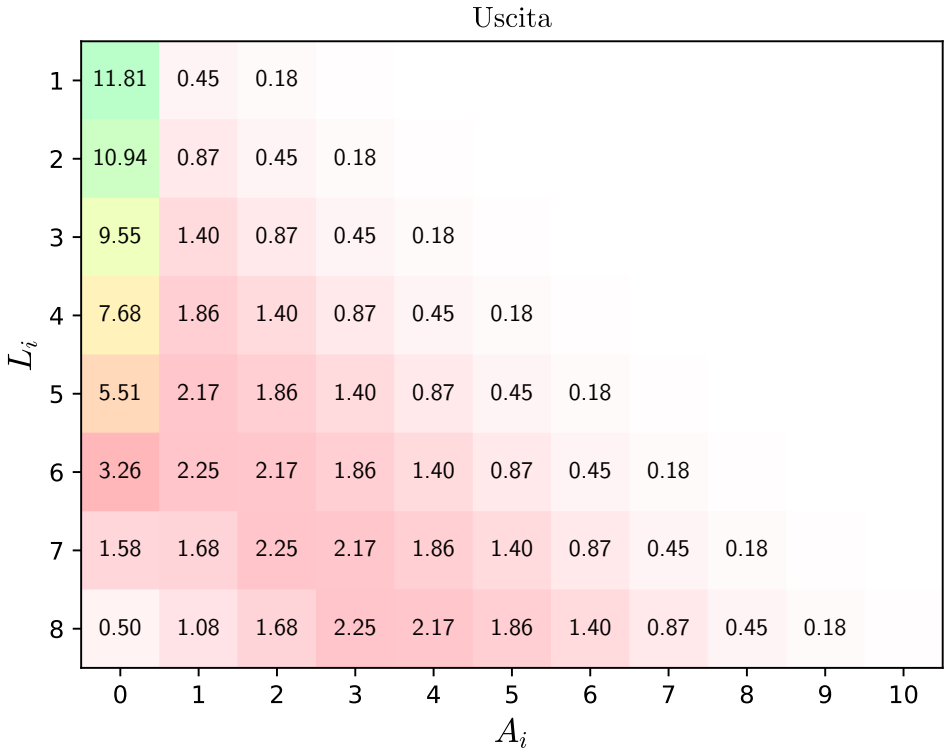

Figura 8: Probabilità all'uscita della coppia $(L_i,\ A_i)$ di un buffer con ingresso uniforme in $L_i \in [1,\ 8],\ A_i \in [0,\ 3]$.

2.3.3 Distribuzione in Uscita

Per quanto detto la distribuzione in uscita avrà la stessa banda di quella in ingresso e mantiene la stessa distribuzione di probabilità in lunghezza.

Le attese invece subiscono un cambiamento seppur mantenendo lo stesso valore medio (perché R_i è nullo a regime). È possibile calcolare la distribuzione in uscita iterando sulle coppie (transazioni, stato) e associando alla transazione in uscita la probabilità di trovare la coppia stessa che risulta in quanto indipendenti:

$$\mathbb{P}(C_{i-1},\ T_i) = \mathbb{P}(C_{i-1})\mathbb{P}(T_i)$$

utilizzando le equazioni (2) e (5).

2.4 Isolate

2.4.1 Errore nella Limitazione

Per come è definito il comportamento dell'Isolate è necessario prevedere un errore sul budget imposto causato da transazioni in ingresso con lunghezze che non esattamente si sommano al budget configurato in un periodo. Per esempio un traffico di sole transazioni di lunghezza 8 fa sì che tutte le configurazioni con budget tra 1 e 8 e con lo stesso periodo si comportino nello stesso esatto modo: facendo passare una transazione sola per periodo. Considerando un caso specifico dell'esempio appena riportato, con budget pari 4 e periodo pari a 16, l'Isolate avrebbe un errore pari a 4 e il budget effettivo risulterebbe pari a 8, ossia il 200% del budget impostato.

Un'alternativa per calcolare l'errore, più utile nella pratica, è utilizzare la rimanenza a fine periodo, qualora negativa, invertita di segno. Nell'esempio appena riportato la rimanenza a fine periodo è pari a -4, pertanto invertita di segno, combacia con l'errore.

Questo errore rende molto più complesso configurare l'unità: se desideriamo limitare un generico traffico a 1/4 della banda non è sufficiente impostare un budget e un periodo pari al quadruplo del budget: le configurazioni (1, 4) e (10, 40) si potrebbero comportare in modi completamente diversi.

Un limite superiore dell'errore può essere trovato considerando il caso peggiore con budget rimanente pari a 1 e una transazione in ingresso di lunghezza massima $\hat{L}$ ottenendo un errore massimo pari a $\hat{L} - 1$.

Qualora il traffico in ingresso sia ignoto possiamo sfruttare il comportamento dello Splitter, ricordando che ogni transazione avrà lunghezza massima pari al limite K con cui è configurato, l'errore massimo sarà sempre pari a $K-1$. Questa relazione ci invita ad abbassare il limite il più possibile per semplificare la configurazione dell'Isolate, idealmente a 1, ricordando però le varie problematiche di limiti più bassi in presenza di componenti non ideali presentate nella sezione 2.2.2.

2.4.2 Ritardo

Il ritardo di una transazione dipende dallo stato, identificato dal budget disponibile e dalla posizione all'interno del periodo:

- Budget disponibile maggiore di 0:
  Il ritardo R_i è identicamente nullo.
  Le transazioni sono inoltrate appena arrivano in ingresso.
- Budget disponibile minore o uguale a 0:
  Il ritardo R_i è pari al periodo rimanente.

Notiamo come la sequenza dei ritardi cumulati sia debolmente crescente, dimostrando come l'unità effettivamente riduca la banda a disposizione. Come anticipato nella sezione precedente la banda a cui si è limitati non è il rapporto $\frac{budget}{periodo}$ ma è calcolabile conoscendo il ritardo medio introdotto e utilizzando l'equazione (4) che definisce la banda stessa.

2.4.3 Modello Aleatorio

La catena di Markov dell'Isolate segue da quanto detto nel capitolo dedicato e dalle considerazioni sullo stato fatte nella sezione precedente. La costruzione risulta più complessa in confronto a quella del Buffer data la mancanza di una relazione semplice tra stato iniziale e stato finale.
A questo fine può essere utilizzata la seguente procedura:

```
def state(start: tuple[int, int], t: Transaction) -> tuple[int, int]:
    available, offset = start

    # Wait
    offset += t.wait
    if offset ⩾ period:
        available = budget
        offset %= period

    # Lengths
    for l in t.lengths:
        # Wait until period is over.
        if available ⩽ 0:
            available = budget
            offset = 0

        # Account for the length.
        available -= l
        offset += l
        if offset ⩾ period:
            available = budget
            offset %= period

    return int(available), int(offset)
```

Possiamo calcolare il ritardo medio $\mathbb{E}[R_i]$ come funzione della distribuzione stazionaria π, considerando che per ogni stato e transazione generalizzata

possiamo replicare la procedura per la creazione e definire cosi una funzione *ritardo*(*stato*, *transazione*) con la quale possiamo definire:

$$\mathbb{E}[R_i] = \sum_{(s,t)\in(Stati,Ingresso)} ritardo(s,t)\mathbb{P}(s,t) \tag{8}$$

$$= \sum_{(s,t)\in(Stati,Ingresso)} ritardo(s,t)\mathbb{P}(s)\mathbb{P}(t) \tag{9}$$

$$= \sum_{(s,t)\in(Stati,Ingresso)} ritardo(s,t)\pi_s\mathbb{P}(t) \tag{10}$$

Una volta calcolato il ritardo medio è possibile ottenere la banda in uscita e verificare il comportamento dell'unità AXI-REALM.

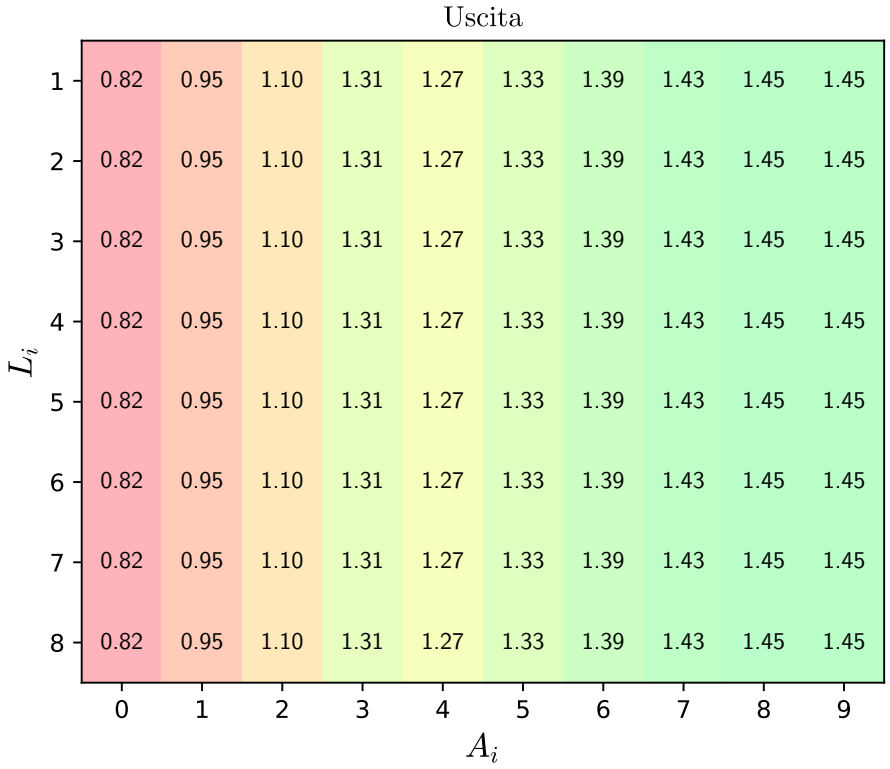

Figura 9: Probabilità all'uscita della coppia ($L_i,\ A_i$) di un isolate configurata con budget 3 e periodo 12 con ingresso l'uscita di un buffer che a sua volta ha traffico in ingresso uniforme in $L_i \in [1,\ 8],\ A_i \in [0,\ 3]$.

2.4.4 Distribuzione in Uscita

Seppur il modello utilizzato non preveda attese tra le lunghezze di una singola transazione, è possibile come per il Buffer calcolare una distribuzione del traffico in uscita equivalente in banda semplicemente raggruppando le attese all'inizio. Il calcolo risulta uguale a quello del buffer, ottenendo il ritardo come specificato nella sezione precedente.

2.5 Risultati

Concludendo l'analisi, l'unità AXI-REALM risulta particolarmente complessa e non priva di difetti.

Configurazione Se noto il traffico in ingresso e la risposta dell'uscita, caso per caso, impostata una certa precisione della limitazione da imporre è possibile ricercare la configurazione ottima. A questo fine è possibile utilizzare i modelli statistici precedentemente introdotti per calcolare l'effetto di una configurazione che seppur in casi semplici, ignorando per esempio la concorrenza tra più dispositivi, risultano comunque significativi.

Quando il traffico e la risposta non sono noti, si può procedere osservando che, impostando il limite a uno e accettando eventuali sprechi di banda, l'errore sulla limitazione risulta nullo. In questo caso, è possibile impostare budget e periodo in base al rapporto desiderato.

Sprechi di Banda Inoltre l'unità non prevede alcun sistema di reclaming della banda, pertanto in sitauzioni dinamiche l'assegnamento di banda a un dispostivo, per esempio disabilitato, viene persa.

Nella prossima sezione viene proposta un'implementazione alternativa della logica di limitazione la quale non è affetta dal problema dell'errore e la cui configurazione risulta più intuitiva, oltre che a implementare un sistema di reclaiming della banda inutilizzata automatico.

3 Alternativa basata sul Deficit

3.1 Introduzione al Deficit Round-Robin

Il Deficit Round Robin (DRR), come in [11] e analizzato in [7, 8, 6, 12], è un algoritmo di scheduling che modifica il Round Robin Pesato, variante a sua volta del Round Robin Classico, che permette l'assegnazione di frazioni di tempo diverse a ogni dispositivo.

In particolare il DRR è un algoritmo che permette lo scheduling fair di traffico caratterizzato da pacchetti a lunghezze intere indivisibili. Il bus AXI4 ricade in questa categorizzazione[3] e l'algoritmo si adatta facilmente al nostro problema.

3.1.1 Spiegazione dell'Algoritmo

Ogni dispositivo è caratterizzato da un budget β_i che rappresenta quanto al più può essere utilizzato il canale a ogni giro, inoltre è presente un contatore δ_i detto di deficit. A ogni giro un dispositivo può inviare una transazione con lunghezza al più $\beta_i + \delta_i$. Se la transazione ha lunghezza superiore a $\beta_i + \delta_i$, il contatore di deficit viene aumentato di β_i e l'invio verrà tentato nel giro successivo.

Infine si definisce il periodo P come la somma dei budget.

Riassumendo il contatore viene gestito con le seguenti regole:

- Impostato a zero quando non ci sono transazioni in attesa;
- Aumentato di β_i quando non è possibile inviare una transazione;
- Decrementato della lunghezza della transazione inviata.

3.1.2 Concetto duale: Surplus

Una variate del DRR, il Surplus Round Robin (SRR) prevede al posto del contatore di deficit, un contatore di surplus. SRR viene presentato in [1], analizzato in [9] e seppur in un contesto diverso, anche in [3, 4]. Questo contatore invece che memorizzare quanto budget si ha a disposizione dai periodi precedenti, memorizza quanto budget è stato anticipato nei periodi precedenti. In questa variante l'invio risulta possibile se il surplus è positivo e non ha dipendenza dalla lunghezza in ingresso.

[3] Nonostante le transazioni si possano dividere, come fatto nello Splitter, sfruttare questa proprietà nell'implementazione richiederebbe uno stravolgimento della struttura e una modifica sostanziale all'interfaccia dell'unità.
Possibili implementazioni sono brevemente discusse nella sezione 4.

```
while True:
    for i in range(1, N):
        if not queue[i].empty():
            DC[i] = DC[i] + Q[i]
            if not queue[i].empty() and DC[i] ⩾ queue[i].head().size():
                DC[i] = DC[i] - queue[i].head().size()
                send(queue[i].head())
                queue[i].dequeue()

            if queue[i].empty():
                DC[i] := 0
```

Figura 10: Procedura per DRR.

```
while True:
    for i in range(1, N):
        if SC[i] < Q[i]:
            SC[i] = min(Q[i], SC[i] + Q[i])

        if not queue[i].empty() and SC[i] > 0:
            SC[i] = SC[i] - queue[i].head().size()
            send(queue[i].head())
            queue[i].dequeue()
```

Figura 11: Procedura per SRR.

Riproponendo le regole illustrate per il DRR:

- All'inizio del giro viene aumentato di β_i fintanto che è minore del budget e limitando la somma al valore del budget stesso;
- Decrementato della lunghezza della transazione inviata, possibilmente assumendo valori negativi.

3.2 Modifiche alla Logica di Isolamento

3.2.1 Problematiche del DRR in Hardware

Tra i due algoritmi presentati nella sezione precedente, il DRR risulta complesso da implementare perché l'inoltro di una transazione dipende dalla lunghezza della transazione stessa.

Questo implica l'esistenza di una rete combinatoria a monte dell'unità Isolate la quale sarebbe sotto vincoli di tempo particolarmente stringenti. Rimane possibile implementare l'algoritmo DRR aggiungendo un ciclo di attesa, ma in confronto, l'algoritmo SRR, precedentemente illustrato, risulta più semplice. Infatti l'invio o meno di una transazione non dipende da alcuna informazione contenuta nella transazione stessa, ma solo dallo stato attuale, ossia dalla sequenza di transazioni precedenti. Questa proprietà rilassa i vincoli di tempo presenti in DRR, permettendo un'implementazione più efficiente.

Concetto di Giro Infine per implementare l'algoritmo è necessario modificare il concetto di giro: non implementando direttamente il round robin non esiste propriamente un giro. Per ovviare a questo problema introduciamo un periodo variabile che rappresenti un giro. Alla fine del periodo precedente l'unità calcola la durata del nuovo periodo sommando il budget di coloro che avrebbero fatto parte del giro.

Per esempio, con 3 dispositivi tutti con budget pari a 3 e contatori rispettivamente a 0, 1 e 3, il nuovo periodo sarà 6 perché i primi due dispositivi necessitano di banda. Contesualmente vengono aggiornati i contatori, che risulteranno per tutti pari a 3.

3.2.2 Bound sul Periodo

In letteratura gli algoritmi presentati vengono analizzati in contesti software e particolare attenzione viene dedicata alla complessità asintotica degli algoritmi. In particolare il periodo ha un limite inferiore al di sotto del quale l'algoritmo diventa O(N) invece che O(1).

In hardware, e in particolare nella struttura considerata, questo problema non sussiste. Infatti la limitazione viene effettuata in parallelo da N unità e l'algoritmo ha comportamento identico per ogni periodo.

3.2.3 Ri-assegnamento della Banda Inutilizzata

Quando un dispositivo non utilizza banda per periodi estesi di tempo l'algoritmo proposto la riassegna in modo proporzionale ai dispositivi attivi. Per poter riassegnare la banda, un dispositivo non deve effettuare alcuna transazione per un intero *periodo*, dove con periodo si intende il periodo generalizzato introdotto nella sezione precedente che possiamo assumere essere la somma dei budget dei dispositivi limitati in banda.

Per assicurarsi che l'interezza della banda sia sempre utilizzata la lunghezza del periodo deve essere sufficientemente piccola da poter permettere ai dispositivi non limitati in banda di rimanere inattivi per l'interezza di un periodo.

Non è difficile trovare configurazioni che non riescono a sfruttare completamente la banda disponibile, per esempio, in figura 12 è riportata la banda per dispositivo e la banda totale utilizzata in vari multipli della stessa configurazione.

Concludendo, la configurazione migliore, nella maggior parte dei casi, sarà la configurazione minima, senza fattori comuni tra i vari pesi. Rimane necessario una verifica, caso per caso, dell'effettiva utilizzazione totale.

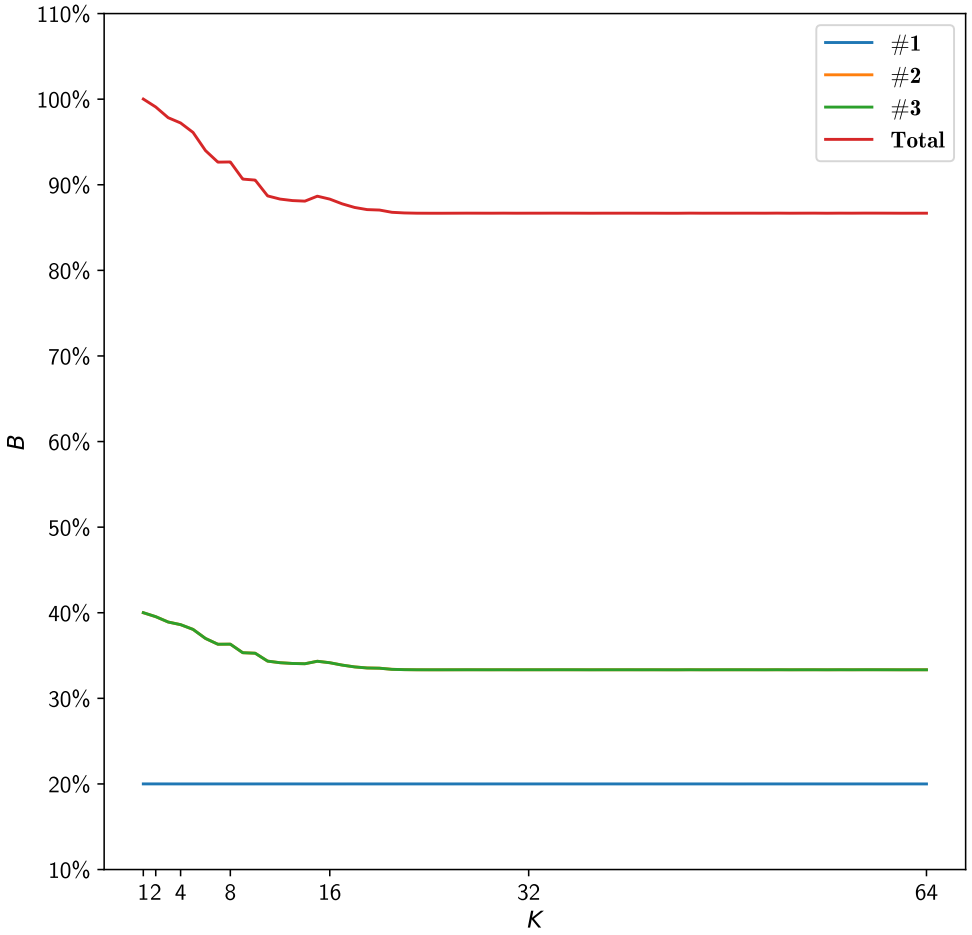

Figura 12: Il dispositivo #1 utilizza il 20% della banda mentre gli altri due sono limitati dall'unità AXI-REALM.
Per multipli maggiori di 1, l'utilizzazione rimane inferiore al 100%.

3.3 Note sull'Implementazione

3.3.1 Nuova Struttura a Blocchi

L'unità originale descritta in [10] prevedeva solamente[4] una numero di porte ognuna delle quali configurabile con il proprio budget e periodo. Seppur funzionale in molti casi, in presenza di un singolo dispositivo con più porte non era generalmente possibile realizzare una configurazione corretta senza limitare il dispositivo all'utilizzo di una singola porta.

A questo fine è stata realizzata una nuova struttura, modulare, dell'unità precedente suddividendola in due moduli assestanti. Il primo detto "device module" limita un dispositivo con un arbitrario numero di porte. Viene configurato direttamente con i limiti per gli splitter e dal secondo modulo con il budget. Il secondo detto "master module" coordina un arbitrario numero di "device module". Viene configurato dall'utente con i budget per ogni dispositivo.

3.3.2 Logica di Controllo

La comunicazione tra master e dispositivi avviene secondo un semplice protocollo: il master ha a disposizione un segnale (`REMAINING`), corrispondente al contatore di eccesso (`SC[i]`), e controlla i dispositivi tramite un segnale di incremento (`INCR`) e un valore (`BUDGET`).

Ogni dispositivo procede in modo autonomo permettendo l'invio fintanto che il budget a sua disposizione rimane positivo, una volta terminato il budget, attende l'arrivo di un segnale di incremento dal master. All'avvio il budget viene impostato a zero.

Il master mantiene al suo interno un contatore (`offset`) e il periodo attuale (`period`), i quali all'avvio vengono rispettivamente impostati a 0 e 1. Detti attivi quei dispositivi con `REMAINING` minore al loro budget, quando `offset` e `period` sono uguali, il master imposta `offset` a zero e `period` pari alla somma dei budget di tutti i dispositivi attivi. Contestualmente comunica ai dispositivi attivi la presenza di ulteriore budget tramite il segnale `INCR` e `BUDGET`. Quando nessun dispositivo è attivo `period` viene impostato a 1.

I dispositivi che ricevono un segnale `INCR` aumentano il proprio budget di `BUDGET` se il valore di `REMAINING` è negativo altrimenti impostano `REMAINING` a `BUDGET`.

[4]Non considerato in questa analisi, l'implementazione originale prevede budget suddiviso per area di memoria, concetto compatibile con l'analisi svolta ma rimosso per semplicità.

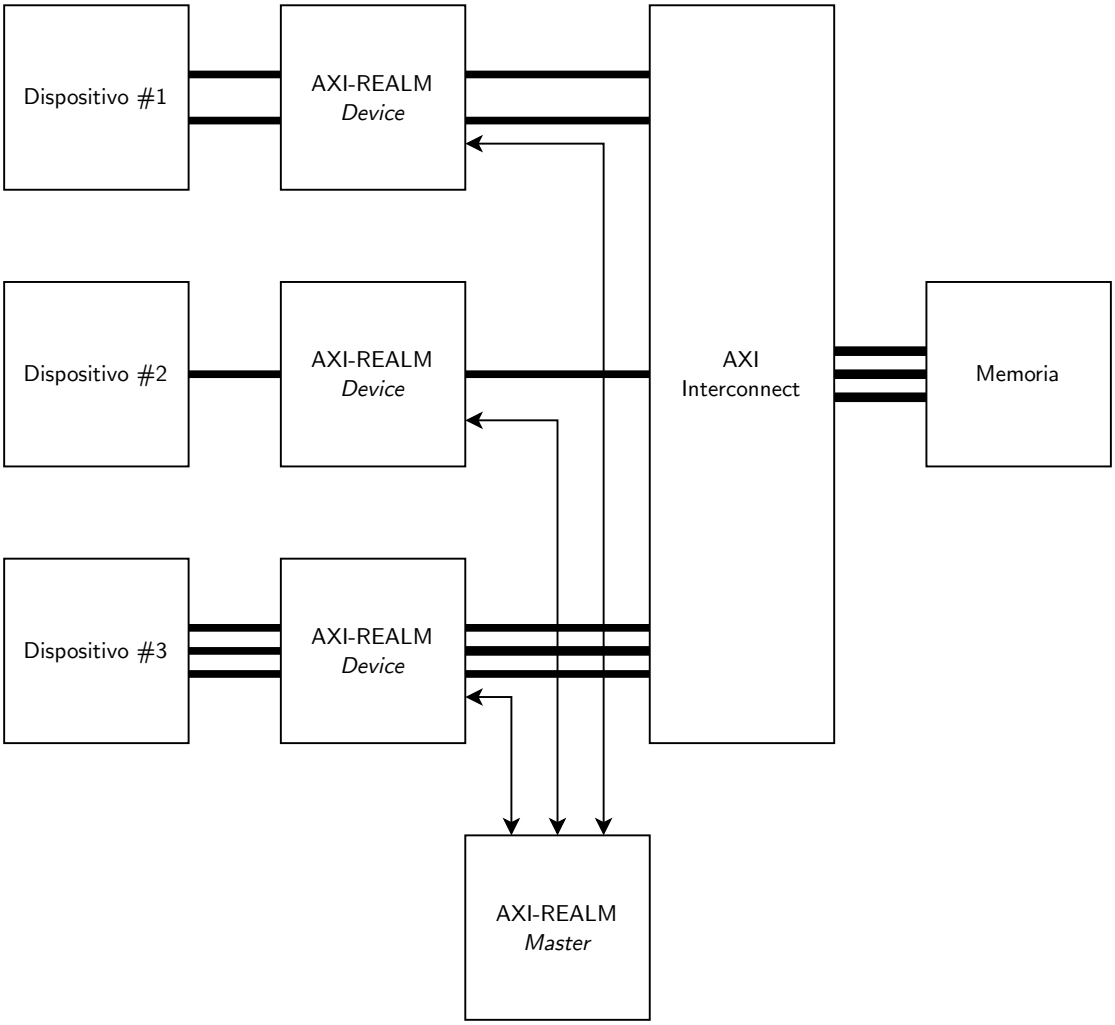

Figura 13: Nuova struttura a blocchi dell'unità AXI-REALM.

4 Conclusioni e Sviluppi Futuri

L'implementazione alternativa dell'Isolate risolve i problemi rilevati nell'analisi e permette una configurazione semplice nella maggior parte dei casi. Nonostante ciò, l'unità non risulta completamente priva di difetti, rimane, seppur considerevolmente ridimensionato, il problema della banda inutilizzata che però può essere ridotto modificando la configurazione.

Rimozione del Buffer Con l'introduzione della nuova logica, la necessità di Buffer e Splitter deve essere nuovamente analizzata. Generalmente sono necessari, dato che non assumiamo nulla sul funzionamento dell'interconnect, ma utilizzando un modello più preciso di interconnect, il risultato potrebbe non essere lo stesso.

Per esempio, la presenza di un buffer all'interno dell'interconnect dovrebbe risultare quasi obbligatoria nella maggior parte delle implementazioni, pertanto manterne due andrebbe semplicemente a introdurre ulteriore ritardo nel sistema.

Integrazione con l'interconnect Per migliorare ancora l'unità, sarebbe necessario riconsiderare l'interfaccia della stessa: integrando anche l'interconnect e dunque implementando il Round Robin dovrebbe essere possibile semplificare ulteriormente la logica. Inoltre, andrebbero prese in considerazione altre varianti dello stesso Deficit Round Robin come quelle analizzate in [8, 3, 4]. Infine, l'integrazione potrebbe sfruttare a pieno le potenzialità dello Splitter, suddividendo dinamicamente le richieste in base allo stato dell'interconnect stesso.

L'unità risultante sarebbe un vero e proprio interconnect fair e configurabile per garantire performance a componenti real-time critici.

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutte le persone che, in un modo o nell'altro, hanno contribuito alla realizzazione di questo lavoro.

In primis, un sentito ringraziamento va al professor Stea dell'Università di Pisa, che mi ha seguito con grande disponibilità e pazienza durante tutto il percorso, nonostante i tempi stretti e le difficoltà incontrate. La sua guida e il suo supporto sono stati fondamentali per il completamento della tesi.

Un ringraziamento speciale va anche al mio tutor, il professor Biondi dell'istituto TeCIP, la cui competenza e il cui entusiasmo hanno reso l'esperienza di ricerca ancora più stimolante. Ringrazio, inoltre, il TeCIP e tutti coloro con cui ho collaborato al progetto di tesi: il vostro contributo è stato prezioso e determinante.

Infine, non posso non ringraziare la mia famiglia, che mi ha sempre permesso di studiare e di perseguire i miei obiettivi, e la mia fidanzata, la cui presenza e incoraggiamento mi hanno spinto a mettermi in gioco e a puntare sempre più in alto, fino ad arrivare a perseguire l'ambizioso traguardo di entrare nella Scuola Superiore Sant'Anna.

A tutti voi, grazie.

Bibliografia

[1] Hari Adiseshu, Guru Parulkar e George Varghese. «A reliable and scalable striping protocol». In: *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '96. New York, NY, USA: Association for Computing Machinery, ago. 1996, pp. 131–141. ISBN: 9780897917902. DOI: 10.1145/248156.248169. URL: https://dl.acm.org/doi/10.1145/248156.248169 (visitato il giorno 16/02/2025).

[2] ARM. *AMBA AXI Specification*. URL: https://developer.arm.com/documentation/ihi0022/hc/?lang=en (visitato il giorno 16/02/2025).

[3] Moris Behnam et al. «Overrun and Skipping in Hierarchically Scheduled Real-Time Systems». In: *2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 2009, pp. 519–526. DOI: 10.1109/RTCSA.2009.62.

[4] Moris Behnam et al. «Overrun Methods and Resource Holding Times for Hierarchical Scheduling of Semi-Independent Real-Time Systems». In: *IEEE Transactions on Industrial Informatics* 6.1 (2010), pp. 93–104. DOI: 10.1109/TII.2009.2037918.

[5] Thomas Benz et al. «AXI-REALM: A Lightweight and Modular Interconnect Extension for Traffic Regulation and Monitoring of Heterogeneous Real-Time SoCs». eng. In: USA, 2024. DOI: 10.23919/date58400.2024.10546511. URL: https://www.iris.sssup.it/handle/11382/571875?mode=simple (visitato il giorno 16/02/2025).

[6] Marc Boyer, Giovanni Stea e William Mangoua Sofack. «Deficit Round Robin with network calculus». In: *6th International ICST Conference on Performance Evaluation Methodologies and Tools*. Ott. 2012, pp. 138–147. DOI: 10.4108/valuetools.2012.250202. URL: https://ieeexplore.ieee.org/document/6376315 (visitato il giorno 16/02/2025).

[7] Luciano Lenzini, Enzo Mingozzi e Giovanni Stea. *Full exploitation of the Deficit Round-Robin capabilities by efficient implementation and parameter tuning*. Gen. 2003.

[8] Luciano Lenzini, Enzo Mingozzi e Giovanni Stea. «Performance analysis of Modified Deficit Round Robin schedulers». en. In: *Journal of High Speed Networks* 16.4 (nov. 2007), pp. 399–422. ISSN: 0926-6801, 1875-8940. DOI: 10.3233/HSN-2007-325. URL: https://journals.sagepub.com/doi/10.3233/HSN-2007-325 (visitato il giorno 16/02/2025).

[9] D. Nikolova e C. Blondia. «Evaluation of surplus round robin scheduling algorithm». In: dic. 2006. URL: https://www.semanticscholar.org/paper/Evaluation-of-surplus-round-robin-scheduling-Nikolova-Blondia/03827c9ac06199219efa50cecc7b9c0c582d2310 (visitato il giorno 16/02/2025).

[10] Francesco Restuccia et al. «Is Your Bus Arbiter Really Fair? Restoring Fairness in AXI Interconnects for FPGA SoCs». In: *ACM Trans. Embed. Comput. Syst.* 18.5s (ott. 2019), 51:1–51:22. ISSN: 1539-9087. DOI: 10.1145/3358183. URL: https://dl.acm.org/doi/10.1145/3358183 (visitato il giorno 16/02/2025).

[11] M. Shreedhar e George Varghese. «Efficient fair queueing using deficit round-robin». In: *IEEE/ACM Trans. Netw.* 4.3 (giu. 1996), pp. 375–385. ISSN: 1063-6692.

[12] Seyed Mohammadhossein Tabatabaee e Jean-Yves Le Boudec. «Deficit Round-Robin: A Second Network Calculus Analysis». In: *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. ISSN: 2642-7346. Mag. 2021, pp. 171–183. DOI: 10.1109/RTAS52030.2021.00022. URL: https://ieeexplore.ieee.org/document/9470448/ (visitato il giorno 16/02/2025).