Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Low-power interconnect for implant SoC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Λυράκης Αλέξιος

Επιβλέπων : Δημήτριος Σούντρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2016

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Low-power interconnect for implant SoC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Λυράκης Αλέξιος

Επιβλέπων : Δημήτριος Σούντρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Σεπτεμβρίου 2016.

Κιαμάλ Πεκμεστζή Καθηγητής Ε.Μ.Π.	Δημήτριος Σούντρης Αναπληρωτής Καθηγητής Ε.Μ.Π.	Γιώργος Οικονομάκος Επίκουρος Καθηγητής

Αθήνα, Σεπτέμβριος 2016

Λυράκης Αλέξιος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Abstract

Implanted medical devices (IMDs) are safety-critical devices that are currently used for the long-treatment of various medical conditions such as cardiac arrhythmias, Parkinson's disease and epilepsy. Modern IMDs employ a wide range of components (sensors, actuators, processors and memory blocks) which communicate with each other in a System-On-Chip (SoC) architecture. The design of IMDs is a challenging task because of the low-power and size constraints; however, little attention has been given so far to the design of the interconnect between the various components. Given the increasing complexity of IMD SoCs, the choice of the communication architecture can play a significant role in the total device's power consumption and size. The purpose of this thesis is to examine different communication architectures regarding their use for IMDs. After considering a range of interconnects and their use for IMDs, we designed and implemented a crossbar and point-to-point interconnect in VHDL and synthesized the designs in UMC 90nm technology. These designs were subsequently evaluated and compared in terms of throughput, latency, size and power consumption. Our results proved that the point-to-point may outperform the crossbar, but the crossbar has a significant lower power consumption than the point-to-point. Consequently, based on IMDs needs for ultra-low power consumption, crossbar should be preferred for implantable medical applications.

Key words

VHDL, IMD, Interconnect, Bus, Crossbar, Point-To-Point

Περίληψη

Οι Εμφυτεύσιμες Ιατρικές Συσκευές (ΕΙΣ) είναι συσκευές που χρησιμοποιούνται για θεραπεία μακράς διαρκείας διάφορων ασθενειών όπως η καρδιακή αρρυθμία, η Νόσος του Παρκινσον και η επιληψία. Οι μοντέρνες ΕΙΣ χρησιμοποιούν ένα ευρύ φάσμα από εξαρτήματα (αισθητήρες, ενεργοποιητές, επεξεργαστές και μπλοκ μνήμης) τα οποία επικοινωνούνε μεταξύ τους σε ένα SoC. Η σχεδίαση μιας εμφυτεύσιμης ιατρικής συσκευής είναι μια απαιτητική διαδικασία λόγω των χαμηλών ενεργεακών περιορισμών και τους περιορισμούς μεγέθους. Σκοπός της παρούσας διπλωματικής είναι να εξετάσεις ένα ευρύ φάσμα επικοινωνιακών αρχιτεκτονικών σχετικά με τη χρήση τους στις ΕΙΣ. Λαμβάνοντας υπ'όψιν μια σειρά από διάφορες αρχιτεκτονικές διασυνδέσης, επιλέξαμε να σχεδιάσουμε και να υλοποιήσουμε μια crossbar και μία point-to-point διασύνδεση σε VHDL και να τις υλοποιήσουμε με τη βοήθεια UMC 90nm τεχνολογίας. Έπειτα εξετάσσαμε τις υλοποιήσεις μας και τις συγκρίναμε με βάση την απόδοση, το μέγεθός και την κατανάλωση ενέργειας. Τα αποτελέσματά μας απέδειξαν ότι η point-to-point διασύνδεση ξεπερνάει σε απόδοση την crossbar, αλλά η τελευταία έχει σημαντικά μικρότερη κατανάλωση ενέργειας. Κατά συνέπεια, με βάση τις ανάγκες των ΕΙΣ για χαμηλή ενέργεια, η crossbar διασύνδεση θα πρέπει να προτιμάται στις εμφυτεύσιμες ιατρικές εφαρμογές.

Λέξεις κλειδιά

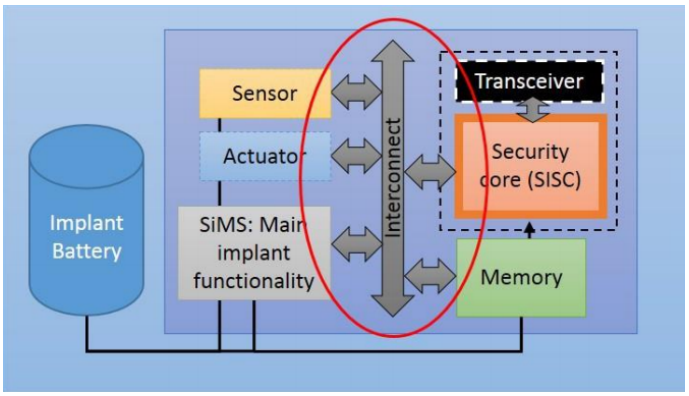
VHDL, Εμφυτεύσιμες Ιατρικές Συσκευές, Διασύνδεση, Bus, Crossbar, Point-To-Point

Εκτεταμένη περίληψη

Εμφυτεύσιμη Ιατρική Συσκευή

Οι εμφυτεύσιμες ιατρικές συσκευές (IMDs) αποτελούνται ουσιαστικά από έναν αριθμό διαφορετικών κυκλωμάτων που επιτελούν ξεχωριστές λειτουργίες (αισθητήρες, ενεργοποιητές, επεξεργαστές, μνήμη) και τα οποία είναι συνδεδεμένα μεταξύ τους έτσι ώστε να δημιουργήσουν ένα σύστημα (SoC). Αυτά τα κυκλώματα χρειάζεται να επικοινωνούν μεταξύ τους έτσι να μπορέσουν να επιτελέσουν τις διάφορες λειτουργείες τους. Ανάλογα με τον ρόλο τους στον τομέα της επικοινωνίας μπορούν να διακριθούν σε 2 κατηγορίες:

- Ενεργά επικοινωνιακά μέρη (αφέντες). Αυτά τα μέρη του συστήματος μπορούν να ξεκινήσουν μια επικοινωνία με άλλα μέρη του συστήματος.
- Παθητικά επικοινωνιακά μέρη (σκλάβοι). Αυτά τα μέρη του συστήματος δεν μπορούν να ξεκινήσουν μια επικοινωνία με άλλα μέρη του συστήματος αλλά είναι υποχρεωμένα να απαντούν στους αφέντες όποτε αυτοί θελήσουν να επικοινωνήσουν μαζί τους.



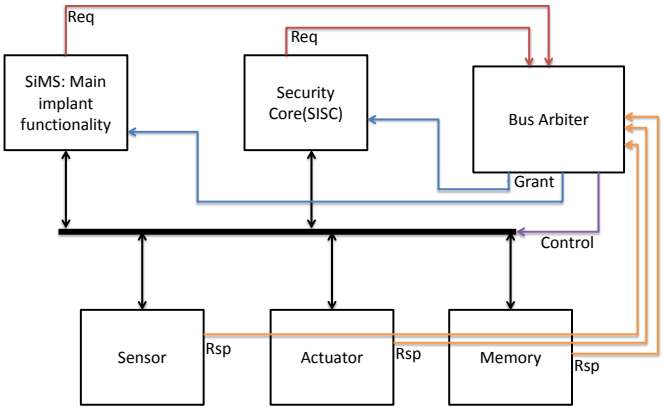
Εικόνα 1: Παράδειγμα ιατρικής εμφυτεύσιμης συσκευής SoC

Η περιγραφή όλων των μερών του συστήματος θα μας βοηθήσει να καταλάβουμε τις επικοινωνιακές τους ανάγκες.

Ο αισθητήρας είναι το κύκλωμα που ανιχνεύει αλλαγές και γεγονότα στο ανθρώπινο σώμα και είναι υπεύθυνο για την συλλογή δεδομένων από αυτό. Χρησιμοποιείται για να μετρήσει παραμέτρους όπως τα επίπεδα γλυκόζης στο αίμα, την θερμοκρασία και τον καρδιακό ρυθμό. Αυτές οι μετρήσεις θα βοηθήσουν την εμφυτεύσιμη ιατρική συσκευή να παρέχει στον ασθενή μια αποτελεσματική θεραπεία. Ανάλογα με την ιατρική εφαρμογή, οι ανάγκες του αισθητήρα ποικίλουν από χαμηλές σε υψηλές.

Ο ενεργοποιητής είναι υπεύθυνος να επιδρά στον ανθρώπινο οργανισμό. Ουσιαστικά ο ενεργοποιητής παρέχει στον ασθενή την θεραπεία που χρειάζεται. Γι αυτό τον λόγο υπάρχουν πολλά είδη ενεργοποιητών ανάλογα με την ασθένεια που αντιμετωπίζει ο ασθενής αν και κυρίων χρησιμοποιείται η ηλεκτρική διέγερση. Όπως και στην περίπτωση του αισθητήρα οι επικοινωνιακές ανάγκες ποικίλουν.

Ο κύριος επεξεργαστής αναλαμβάνει όλες τις λειτουργίες, τους υπολογισμούς και τις διαδικασίες που έχει να εκτελέσει η συσκευή και είναι γενικώς υπεύθυνος για τη σωστή λειτουργία του εμφυτεύσιμου.



Εικόνα 2: Παράδειγμα μιας bus τοπολογίας.

Η επικοινωνιακή του δραστηριότητα είναι πολύ σημαντική και περιλαμβάνει λήψη δεδομένων από τον αισθητήρα, αποστολή εντολών στον ενεργοποιητή και αποθήκευση δεδομένων στη μνήμη. Οι IMDs μπορεί να έχουν πολλούς επεξεργαστές, οι οποίοι είναι υπεύθυνοι για διαφορετικές υπολογιστικές λειτουργίες. Για να παρέχει στους επεξεργαστές ένα μέρος για την αποθήκευση δεδομένων, το SoC περιλαμβάνει και μνήμη.

Άλλο σημαντικό χαρακτηριστικό του εμφυτεύσιμου είναι η επικοινωνία ανθρώπου-συσκευής. Γι αυτό το λόγο ένα σπείρωμα χρησιμοποιείται από το χρήστη για να επικοινωνήσει με τη συσκευή. Όλα τα αναφερθέντα κυκλώματα ηλεκτροδοτούνται από μία κύρια μπαταρία.

Τοπολογία Διασύνδεσης

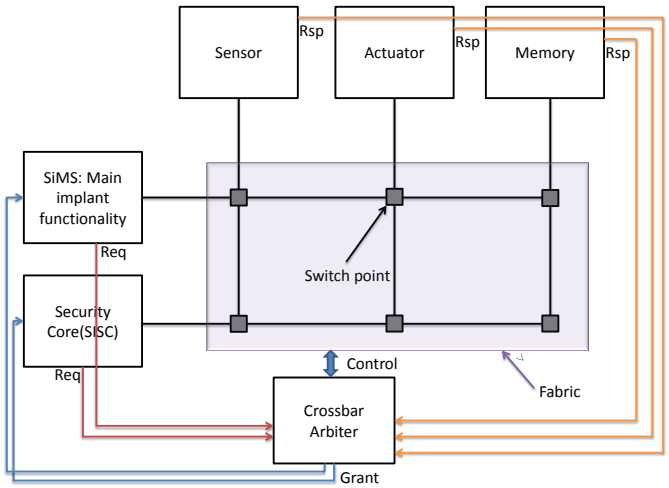
Η τοπολογία μιας SoC διασύνδεσης ορίζεται απ' τον τρόπο με τον οποίο οργανώνονται τα μέρη που επικοινωνούν. Με βάση τις επικοινωνιακές ανάγκες, τους ενεργειακούς περιορισμούς , τους περιορισμούς μεγέθους και την επιθυμία για υψηλή απόδοση περιγράφουμε παρακάτω τρεις τοπολογίες που πιστεύουμε ότι πιθανότατα ικανοποιούν τις IMDs.

Bus

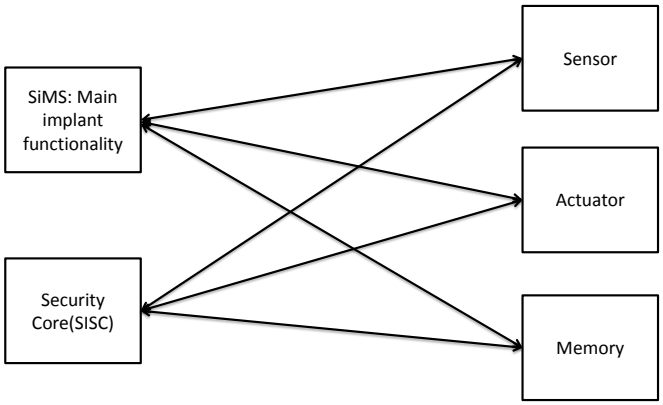
Η bus τοπολογία είναι ένα δίκτυο του οποίου τα μέρη που είναι συνδεδεμένα μ' αυτό επικοινωνούν με ένα κοινό μέσο. Καθένα απ αυτά τα μέρη είναι ικανό να τοποθετήσει τα δεδομένα του στο bus ή να αισθανθεί τα δεδομένα που υπάρχουν σ' αυτό. Κάθε φορά μόνο ένα απ αυτά τα μέρη μπορεί να τοποθετήσει το δεδομένα του στο bus, ενώ τα υπόλοιπα μέρη μπορούν να επιλέξουν είτε να τα αποδεχθούν είτε να τα απορρίψουν. Λόγω του ανταγωνισμού που υπάρχει μεταξύ αυτών των μερών, χρησιμοποιείται ένα πρωτόκολλο διατησίας με βάση το οποίο αποφασίζεται ποιο από αυτά μπορεί να τοποθετήσει το σήμα του. Τα πλεονεκτήματά του είναι ότι είναι απλό, έχει χαμηλό κόστος και επεκτασιμότητα. Το μειονέκτημά του είναι η χαμηλή απόδοση [1] [2].

Crossbar

Ένα NxM (N αφέντες, M σκλάβοι) crossbar προσφέρει ένα μονοπάτι από κάθε αφέντη σε κάθε σκλάβο. Σε κάθε διασταύρωση αυτών των μονοπατιών υπάρχει ένας διακόπτης ο οποίος εφόσον είναι ενεργοποιημένος συνδέει τον αφέντη με το σκλάβο. Για τη σωστή λειτουργία πρέπει το πολύ ένας αφέντης να είναι συνδεδεμένος σε κάθε σκλάβο κάθε στιγμή. Αυτός ο περιορισμός επιτυγχάνεται με τη χρησιμοποίηση ενός κεντρικού διαιτητή ο οποίος αποφασίζει ποιος αφέντης θα λάβει άδεια να επικοινωνήσει και ο οποίος ελέγχει τους διακόπτες. Το σύνολο των διακοπτών αυτής της τοπολογίας αναφέρεται και ως fabric. Το κύριο πλεονέκτημα αυτής της τοπολογίας είναι ότι πολλοί αφέντες μπορούν να εξυπηρετηθούν σε μια στιγμή όσο κάθε σκλάβος παραμένει συνδεδεμένος με το πολύ έναν αφέντη.



Εικόνα 3: Παράδειγμα μιας crossbar τοπολογίας.



Εικόνα 4: Παράδειγμα μιας full point-to-point τοπολογίας.

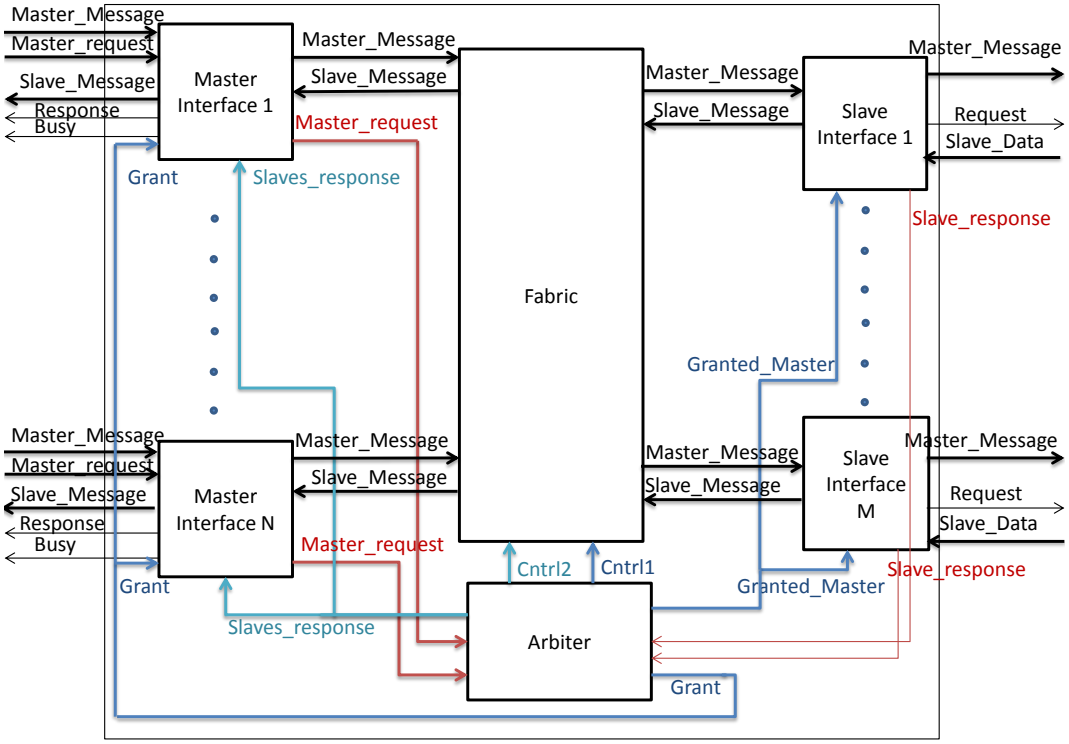
Συνεπώς, αυτή η τοπολογία παρέχει μεγαλύτερη απόδοση, από τη bus τοπολογία. Οπότε όσο η κίνηση μεταξύ 2 μερών του συστήματος αυξάνεται, δεν επηρεάζεται η κίνηση μεταξύ άλλων μερών. Το μεγαλύτερο μειονέκτημα είναι το κόστος υλοποίησης, το οποίο αυξάνεται αναλόγως του $*$ [2].

Point to point

Η point-to-point τοπολογία είναι μια απ τις πιο απλές, επειδή δύο μέρη τα οποία χρειάζεται να επικοινωνήσουν μεταξύ τους, συνδέονται απλά μεταξύ τους με ένα σύνδεσμο. Υπάρχουν 2 τύποι point-to-point: full και custom διασύνδεση. Ο full τύπος συνδέει κάθε αφέντη με κάθε σκλάβο, ενώ ο custom συνδέει τους αφέντες με τους σκλάβους, ανάλογα με τις ανάγκες επικοινωνίας. Το μεγαλύτερο πλεονέκτημα είναι ότι η απόδοση παραμένει σταθερή ανεξάρτητα από τον αριθμό των συνδεδεμένων μερών, ενώ το μειονέκτημά του είναι ότι οι ανάγκες συνδέσμων, ο αριθμός των διεπαφών και η πολυπλοκότητά του αυξάνονται ραγδαία καθώς ο αριθμός των μερών μεγαλώνει.

Σχεδιασμός του crossbar

Ο σχεδιασμός μας ενός NxM crossbar φαίνεται στην εικόνα 5. Τα σήματα εισόδου του crossbar είναι: N *Master_message*, N *Master_request* και M *Slave_data*. Τα σήματα εξόδου είναι: M *Master_message*, N *Busy*, N *Response*, N *Slave_message* και M *Request*. Ο σχεδιασμός μας είναι ευέλικτος και το crossbar μπορεί να επεκταθεί πάνω ή κάτω απλά αλλάζοντας τις ακόλουθες παραμέτρους: Τον



Εικόνα 5: Σχεδιασμός της NxM crossbar διασύνδεσης.

αριθμό των αφεντών, τον αριθμό των σκλάβων, το εύρος των δεδομένων και το εύρος της διεύθυνσης. Τα σήματα clock και reset είναι καθολικά. Μια απλή ανταλλαγή μεταξύ ενός αφέντη και ενός σκλάβου συμβαίνει ως εξής:

Φάση 1η: Διατησία Ο αφέντης ξεκινάει μια αίτηση για μεταφορά μηνύματος. Η διεπαφή του αφέντη αποθηκεύει το μήνυμα του αφέντη και ζητάει άδεια απ τον διαιτητή για να μεταδώσει το μήνυμα στον επιθυμητό σκλάβο.

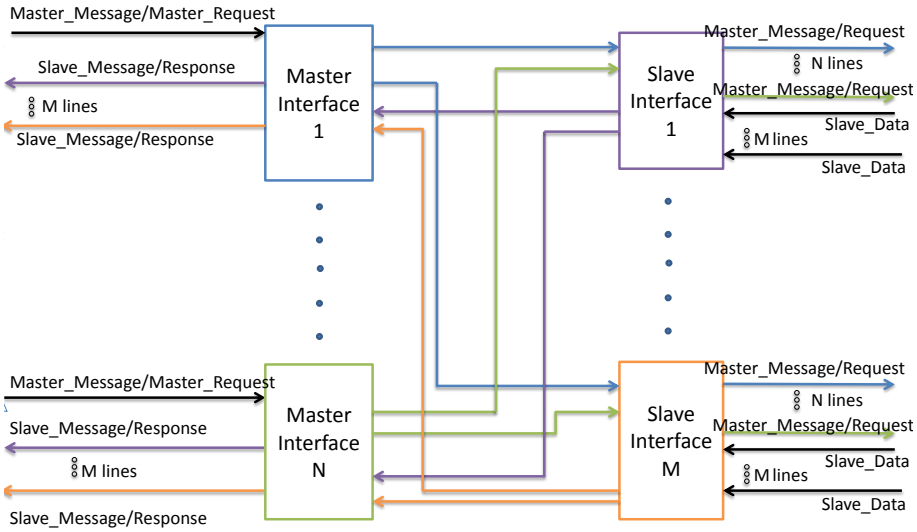
Φάση 2η: Μεταφορά Αν η διεπαφή του αφέντη λάβει άδεια απ τον διατητή, τότε το μήνυμα μεταφέρεται στον σκλάβο. Αν όχι τότε ξαναζητάει άδεια απ τον διατητή.

Φάση 3η: Απάντηση Αν το μήνυμα το οποίο παραδόθηκε στον σκλάβο ήταν τύπου read, τότε ο σκλάβος πρέπει να απαντήσει με δεδομένα. Θεωρούμε ότι αυτή η απάντηση λαμβάνει μέρος, αμέσως μετά τη λήψη του read τύπου μηνύματος.

Σχεδιασμός του point-to-point

Ο σχεδιασμός μας ενός NxM point-to-point φαίνεται στην εικόνα 6. Τα σήματα εισόδου του crossbar είναι: N *Master_Message*, N *Slave_request* και M*N *data*. Τα σήματα εξόδου είναι: N*M *Master_Message*, N*M *Slave_Message*, N*M *Request* και N*M *Response*. Ο σχεδιασμός μας είναι ευέλικτος και το point-to-point μπορεί να επεκταθεί πάνω ή κάτω απλά αλλάζοντας τις ακόλουθες παραμέτρους: Τον αριθμό των αφεντών, τον αριθμό των σκλάβων, το εύρος των δεδομένων και το εύρος της διεύθυνσης. Τα σήματα clock και reset είναι καθολικά. Μια απλή ανταλλαγή μεταξύ ενός αφέντη και ενός σκλάβου συμβαίνει ως εξής:

Φάση 1η: Μεταφορά Μόλις ο αφέντης αποφασίσει να στείλει ένα μήνυμα σ έναν σκλάβο, η διεπαφή του αφέντη αποδέχεται το μήνυμα και το παραδίδει στον επιθυμητό προορισμό.



Εικόνα 6: Σχεδιασμός της full Point-To-Point διασύνδεσης.

Φάση 2η: Απάντηση Αν το μήνυμα το οποίο παραδόθηκε στον σκλάβο ήταν τύπου read, τότε ο σκλάβος πρέπει να απαντήσει με δεδομένα. Θεωρούμε ότι αυτή η απάντηση λαμβάνει μέρος, αμέσως μετά τη λήψη του read τύπου μηνύματος.

Πειραματική διάταξη

Μέτρα αξιολόγησης

Throughput

Ορίζεται ως ο μέσος αριθμός των εξυπηρετούμενων αιτημάτων ανά κύκλο. Μετράται υπολογίζοντας το συνολικό αριθμό των αιτημάτων που η διασύνδεση εξυπηρετεί σ' ένα συγκεκριμένο πλαίσιο χρόνου και είναι ίσο με τη διαίρεση των συνολικών αιτημάτων που εξυπηρετήθηκαν προς το μετρούμενο χρόνο. Μετρήσαμε τον συνολικό αριθμό των εξυπηρετούμενων αιτημάτων μέσο της χρήσης του Modelsim. Η μετρούμενη throughput εξαρτάται από την κίνηση που εξυπηρετεί, ενώ η μέγιστη throughput από το σχεδιασμό.

Latency

Ορίζεται ως ο μέσος χρόνος που χρειάζεται η διασύνδεση για να ολοκληρώσει μια ανταλλαγή μηνυμάτων. Η latency μετράται υπολογίζοντας πόσους κύκλους χρειάζεται για να ολοκληρωθεί ένα αίτημα ανταλλαγής αφού πρώτα είχε αιτηθεί από τον αφέντη. Για μια crossbar διασύνδεση η μικρότερη latency είναι 3 κύκλοι. Μπορεί να χρειαστούν περισσότεροι από 3 κύκλοι σε περίπτωση συγκρούσεων των αιτημάτων. Από την άλλη μεριά μια point-to-point διασύνδεση έχει σταθερή latency ίση με 2 κύκλους.

Area

Ο χώρος που καταναλώνει η διασύνδεση εξαρτάται από τον αριθμό των κελιών και των τύπων των κελιών. Για να το μετρήσουμε χρησιμοποιήσαμε το εργαλείο "Design Compiler (DC)" και τη βιβλιοθήκη UMC 90nm.

Power

Ορίζεται ως η ενέργεια που δαπανάται απ τη συσκευή ανά μονάδα χρόνου. Η συνολική δαπανόμενη ισχύς στα CMOS VLSI μπορεί να κατηγοριοποιηθεί σε δύο βασικές κατηγορίες:

- Στατική ισχύς, η οποία δαπανάται όταν η συσκευή είναι σε σταθερή κατάσταση.
- Δυναμική ισχύς, η οποία δαπανάται όταν η συσκευή είναι ενεργή.

Η συνολική κατανάλωση ισχύς μπορεί να υπολογιστεί ως: $P = P + P$.

Παράμετροι αξιολόγησης

Όπως αναφέραμε, ο σχεδιασμός των 2 διαφορετικών τοπολογιών που υλοποιήσαμε είναι ευέλικτος. Μπορεί να προσαρμοστεί σε οποιοδήποτε αριθμό σκλάβων και αφεντών και οποιοδήποτε εύρος μεταφοράς απλά αλλάζοντας μερικές παραμέτρους στην VHDL. Σε σειρά με τα αναμενόμενα μεγέθη τωρινών και μελλοντικών εμφυτεύσιμων ιατρικών συσκευών επιλέξαμε να υλοποιήσουμε τα εξής μεγέθη: 2x2, 2x4, 2x8, 2x16, 2x32, 4x4, 4x8, 4x16 and 4x32. Η επιλογή του μεγέθους 2xM είναι λογική γιατί 2x2-2x8 μεγέθη είναι πιθανά σενάρια σήμερα (βλ. Παράρτημα Β'). Ενώ 32 σκλάβοι είναι πιθανότατα πολλοί για το κοντινό μέλλον, αξιολογούμε γενικώς την βιωσιμότιτητά τους για τα εμφυτεύσιμα. Επίσης υπάρχουν κάποιες μοντέρνες εμφυτεύσιμες ιατρικές συσκευές, οι οποίες χρησιμοποιούν περισσότερους από 2 επεξεργαστές (π.χ. ένα εμφυτεύσιμο σύστημα για τη θεραπεία νευρολογικών διαταραχών [3]) γι' αυτό πιστεύουμε ότι η επιλογή του 4xM μεγέθους είναι ρεαλιστική επίσης. Η προκαθορισμένη διασύνδεση υποθέτει 8-bit δεδομένα και 32-bit διεύθυνση. Για να καθορίσουμε όμως πως το σύστημά μας ανταποκρίνεται σε συνάρτηση με αυτές τις παραμέτρους, επιλέξαμε να υλοποιήσουμε και interconnects με 4-bit δεδομένα και 16-bit διεύθυνση. Σ αυτές τις τελευταίες διασυνδέσεις ένας έξτρα διαχωριστικός και ανακατασκευαστικός μηχανισμός απαιτείται να ενσωματωθεί στη διεπαφή του αφέντη και του σκλάβου. Αυτός ο μηχανισμός διαχωρίζει το μήνυμα ώστε να μπορεί να το μεταφέρει και το ανακατασκευάζει πριν το παραδώσει στο σκλάβο. Το ίδιο συμβαίνει και στην περίπτωση της μεταφοράς της απάντησης του σκλάβου.

Παράμετροι μοτίβων κίνησης

Τα μοτίβα κίνησης καθορίζουν τον προορισμό και τον ρυθμό γέννησης αιτημάτων από τους αφέντες. Βασισμένοι στην προσπάθειά μας να αξιολογήσουμε τις διασυνδέσεις για ένα μεγάλο εύρος επικοινωνιακών σεναρίων δημιουργήσαμε διάφορα τυχαία μοτίβα κίνησης. Οι επιλεγμένες τιμές των παραμέτρων τους φαίνονται στον πίνακα 1.

Πειραματικά αποτελέσματα

Throughput

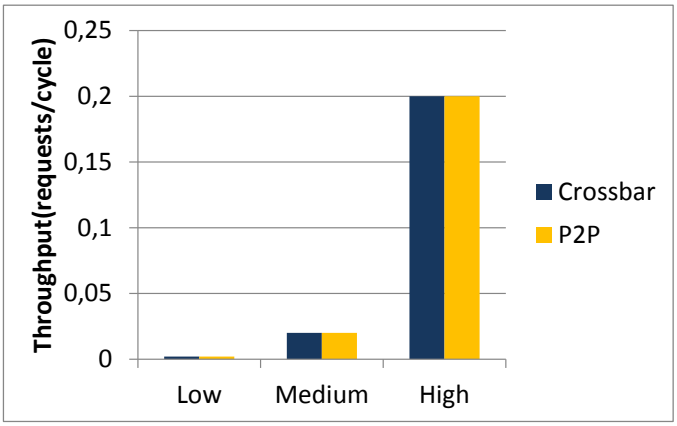
Για κάθε μοτίβο κίνησης και μέγεθος παρατηρούμε ότι μετράμε το ίδιο throughput και για τους 2 τύπους διασύνδεσης (βλ. εικόνα 7). Παρ' όλα αυτά αναμένουμε ότι η point-to-point διασύνδεση έχει μεγαλύτερο δυνατό throughput.

Latency

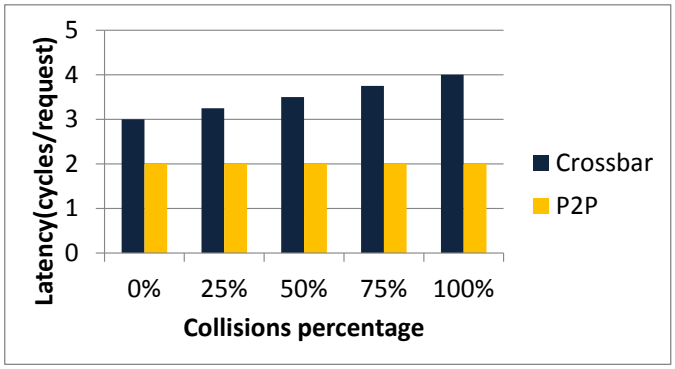
Η point-to-point διασύνδεση έχει μικρότερη μέση latency από το crossbar. Αυτό οφείλεται στις ανάγκες του crossbar για διαιτησία. Η διαιτησία καθυστερεί την μεταφορά ενώ σε περίπτωση σύγκρουσης αιτημάτων διαφορετικών αφεντών η latency του crossbar μεγαλώνει (βλ. εικόνα 8).

Πίνακας 1: Τυχαία μοτίβα κίνησης

Pattern names	Generation rate	Potential Collisions
L0	Low($\frac{2\ requests}{1000\ cycles}$)	0%
L25		25%
L50		50%
L75		75%
L100		100%
M0	Medium($\frac{2\ requests}{100\ cycles}$)	0%
M25		25%
M50		50%
M75		75%
M100		100%
H0	High($\frac{2\ requests}{10\ cycles}$)	0%
H25		25%
H50		50%
H75		75%
H100		100%



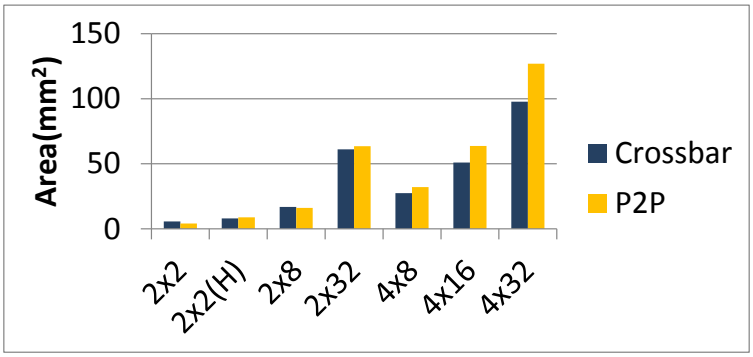
Εικόνα 7: Παράδειγμα ενός 2x16 crossbar and ενός 2x16 point-to-point τα οποία εξυπηρετούνε τα μοτίβα κίνησης L0,M0 και H0.



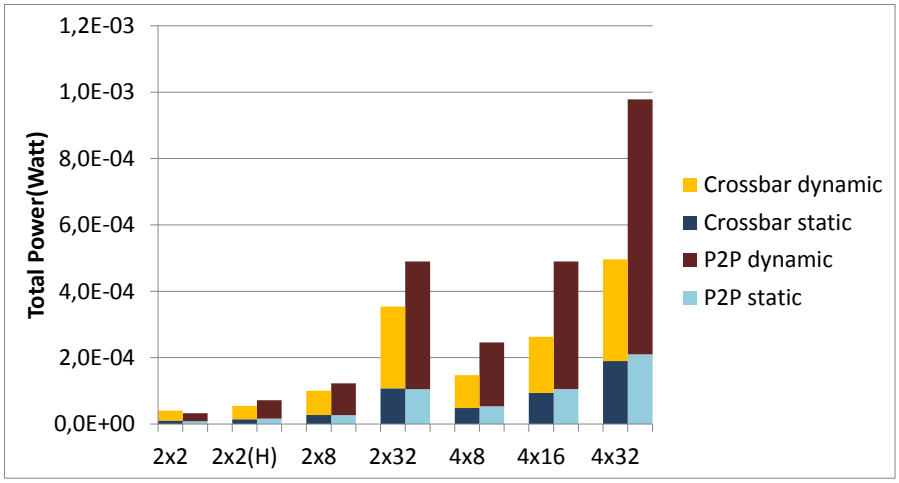
Εικόνα 8: Παράδειγμα ενός 2x16 crossbar κι ενός 2x16 point-to-point διασύνδεσης που εξυπηρετούν τα μοτίβα κίνησης L0-L100.

Area

Βρίσκουμε ότι η point-to-point υλοποίηση απαιτεί λιγότερο χώρο για υλοποιήσεις μικρών διασυνδέσεων. Αυτό συμβαίνει γιατί η point-to-point υλοποίηση δεν απαιτεί διαιτητή και ο σχεδιασμός της διεπαφής



Εικόνα 9: Σύγκριση της κατανάλωσης χώρου crossbar και P2P υλοποιήσεων.



εικόνα 10: Η μέση κατανάλωση ισχύς των crossbar των point-to-point διασυνδέσεων κατά την εεξυπηρέτηση του M25 μοτίβου κίνησης.

αφέντη είναι απλούστερος. Αντίστροφα, το crossbar εκμεταλεύται την προσαρμοστικότητα του fabric (λιγότεροι σύνδεσμοι) και τον απλούστερο σχεδιασμό της διεπαφής σκλάβου (έχει να χειριστεί μόνο ένα αίτημα και απάντηση κάθε στιγμή) για μεγαλύτερα μεγέθη κι αυτό οδηγεί σε μικρότερο κόστος χώρου. Το Crossbar, έτσι, προσαρμόζεται καλύτερα στην αύξηση του μεγέθους της διασύνδεσης (βλ. εικόνα 9).

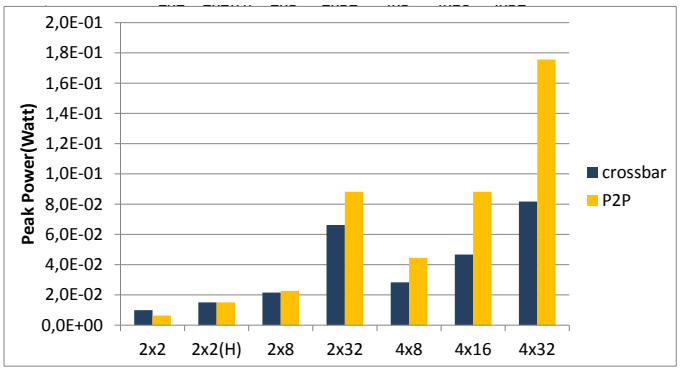
Power

Σε όλες τις περιπτώσεις η Point-to-Point διασύνδεση καταναλώνει περισσότερη ενέργεια από τη crossbar. Αυτό οφείλεται στο ότι καταναλώνει περισσότερη δυναμική ενέργεια από το crossbar.

Συμπεράσματα και προτάσεις για έρευνα

Με βάση τα παραπάνω αποτελέσματα, πιστεύουμε ότι η crossbar διασύνδεση πρέπει να προτιμάται στις παρόντες και μελλοντικές εμφυτεύσιμες ιατρικές συσκευές έναντι μιας point-to-point διασύνδεσης, με βάση τα αναμενόμενα μοτίβα κίνησης και του χαμηλούς ενεργειακούς περιορισμούς. Μερικές ιδέες που θα μπορούσαν να υλοποιηθούν σε μελλοντική έρευνα είναι:

- Υλοποίηση διαφορετικών τοπολογιών πέρα από crossbar και point-to-point. Η bus αρχιτεκτονική είναι η επόμενη υποσχόμενη εναλλακτική λόγω της πιθανής χαμηλότερης κατανάλωσης ενέργειας που μπορεί να προσφέρει.



εικόνα 11: Η μέγιστη κατανάλωση ισχύς των crossbar των point-to-point διασυνδέσεων κατά την εεξυπηρέτηση του M25 μοτίβου κίνησης.

- Μέθοδοι μείωσης της κατανάλωσης ισχύς μπορούν να υλοποιηθούν. Δύο από αυτές τις μεθόδους περιγράφονται στο παράρτημα Α. Παρ'ολα αυτά δεν είναι σίγουρη η αποτελεσματικότητα της υλοποίησης τους καθώς αυτή εξαρτάται από τα χαρακτηρηστικά του συστήματος.

- Fault-tolerant τεχνικές μπορούν να υλοποιηθούν επίσης έτσι ώστε να διορθώνουν ελλατωματική συμπεριφορά των μερών του συστήματος και για να επιτευχθεί υψηλή αξιοπιστία. Αυτές οι τεχνικές είναι πολύ σημαντικές γιατί οποιαδήποτε περίπτωση βλάβης μπορεί να οδηγήσει σε τραυματισμό του ασθενή ή ακόμα και στον θάνατο. Φυσικά η υλοποίηση μιας τέτοιας μεθόδου θα ανεβάσει και το κόστος χώρου και την κατανάλωση ισχύς.

Acknowledgements

I would like to thank Professor Dimitrios Soudris for giving me the opportunity to carry out my diploma thesis under his supervision. His teaching approach and his insightful and extensive research have strongly increased my motivation and commitment to work on this project.

I also would like to thank Professor Christos Strydis, from Neuroscience department of the Erasmus Medical Center, for the great interest shown in our common project from the beginning of our work together. Our constructive and definitely creative meetings greatly encouraged me during my work.

The completion of this thesis however would be impossible without the enormous support and cooperation with Robert Seepers, PhD student at Erasmus Medical Center. His experience and constant assistance made it possible for me to accomplish this work.

Finally and last but not least, I would like to thank my family for their support and help to accomplish my goals, as well as my friends for listening and supporting me.

Lyrakis Alexios,

Athens, September 26, 2016

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

IMDs made their first appearance in 1958 when the first Implantable Pacemaker was created. Despite the significance of the moment, the first implantable device functioned for only 3 hours and had to be replaced by a second the same day [4]. Thanks to technological advances, IMDs have increased in complexity since then and they have expanded to an ever wider range of medical applications. For example, in electrical neuromodulation, they can be used to treat Parkinson's disease, epilepsy, bladder control, gastrointestinal disorders and numerous psychological disorders such as obsessive-compulsive disorder [5]. Nowadays they have become a crucial part in many different medical applications and with this increasing need for implantable devices comes a set of technical challenges.

The design of an IMD is a challenging task and demands many general requirements such as low power and minimal size. The low power consumption is important as IMDs are battery-powered devices that are intended for long-term treatment, yet, can often not be recharged while in operation. Also, the power source and encapsulation components remain the major contributors to the overall weight and size of the device [6]. The size is important because smaller and lighter devices will be less invasive to the patient's body and will cause less pain and discomfort, while also improving the healing process.

In this thesis, we focus on the way the IMD components are communicating with each other. Different ways of communication lead to different levels of performance, area cost and power consumption. There is a wide range of various interconnect types that someone can choose from; bus, point-to-point, crossbar, network-on-chip etc. The question that we try to address in this work is: Which interconnect type suits IMDs better, given their needs and constraints? A theoretical analysis may give us an approximate answer but is not adequate due to the various factors that have to be taken under consideration(number of components, communication activity, interconnect interface differs for each type etc.). For our thesis purpose, we chose to design and implement a crossbar and a point-to-point interconnect, based on their effectiveness for small-medium size systems. We measured their performance, area cost and power consumption through the execution of different traffic patterns for different IMD-SoC sizes.

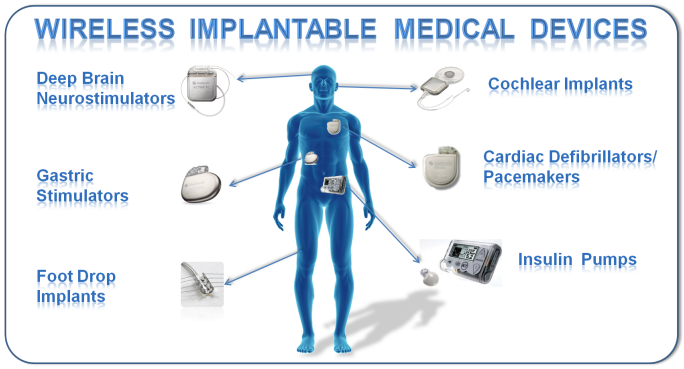


Figure 1.1: Examples of wireless implantable medical devices.

The remainder of this thesis is structured as follows: In Chapter 2, a brief overview of an Implantable Medical Device and of the basic interconnect topologies is presented. Some basic interconnect topologies are presented in order to help the reader understand their basic characteristics. Related work progress on the field is also included, to highlight the contribution of our work and to draw some meaningful results from other experiments. In Chapter 3, we present our implementation of a crossbar and a point-to-point interconnect. In Chapter 4, we evaluate our interconnects implementations in terms of performance, area cost and power consumption and compare the results of our two interconnects. At the end a comparison between these two types is attempted. We conclude this thesis in Chapter 5 by summarizing our main findings and highlighting interesting research directions for future work.

CHAPTER 2

Background and related work

This chapter contains the essential information that the reader should be familiar with in order to better understand our thesis results. We begin with a brief analysis of the structure of the IMD and review some basic interconnect topologies (bus, crossbar, point-to-point). Afterwards, previous work concerning the above SoC interconnect topologies is presented.

2.1 Implantable Medical Device components role

From this thesis abstract point of view, an IMD is an assemblage of a number of different types of functional components (sensor, actuator, processors, memory) which are connected to form a system. The systems whose therapy is delivered based on the measurements of the sensor are defined as ”closed-loop” while those whose therapy is delivered based on the patient’s doctor programming (with no real-time closed loop feedback) are defined as ”open-loop”. An IMD’s components, in order to perform their computations, need to communicate with each other. Depending on their role in communication activity, these peripherals can be classified in two categories:

- Active communication components (processors). These components can initialize a transfer between them and other components. From now on, we refer to these modules as ”masters”.
- Passive communication components. These components cannot initialize any transfer (sensor, actuator, memory). They are able only to receive messages and to respond with data. From now on, we refer to these modules as ”slaves”.

Below we describe the function of each of these peripherals. This will help the reader conceive the communication needs of an IMD.

The sensor is the module that detects events and changes in human body and is responsible for the acquisition of data from it. Sensors are used to measure parameters such as blood-glucose levels, temperature or cardiac rhythm. These measurements are necessary to help the IMD deliver an efficient therapy and to make decisions based on them. Depending on the medical application, the sensor’s communication needs range from low to high. For example, a glucose sensor (i.e. used by an ”artificial pancreas”) that is detecting the glucose in the bloodstream may provide 8-12 data bits per minute, while an ECG sensor (i.e. used by a combined artificial pacemaker)that is monitoring the heart rate of the patient may provide 2.400-3.600 data bits per second.

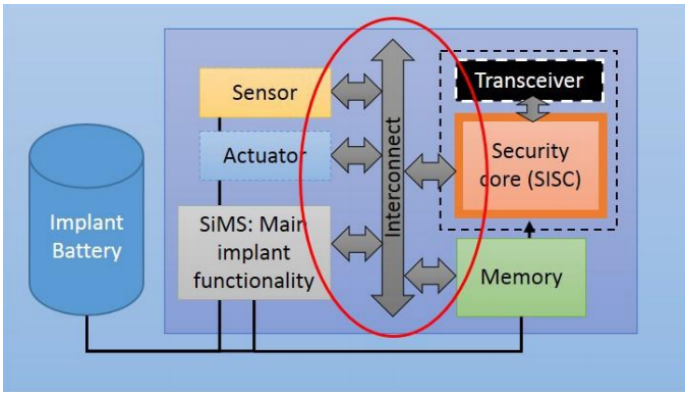


Figure 2.1: SiMS System-on-Chip for implants.

The actuator is responsible for the intervention to the human body. It is in charge of delivering the therapy based on the control system. The kind of therapy delivered by an IMD depends on the specific functional requirements while the most common is the electrical stimulation. As in the case of the sensor, the communication needs of the actuator differ from application to application. For example, an insulin pump (actuator that release glucose to the blood when the glucose-levels are low) may consume 8-12 data bits per minute, while the actuator of an implantable pacemaker (if the heart rate is too low, the pacemaker will send electric stimulation to the heart to keep it beating) may consume 8-12 data bits per second. Considering future technology, an actuator of a future IMD that can suppress epileptic seizures may consume up to 12.800-19.200 data bits per second.

The main processor takes care of all the processes/calculations/functions the device has to execute and in general is responsible for the correct main implant functionality. Its communication activity is paramount and includes requesting data from the sensor, sending commands to the actuator and storing sensor data and actuator commands to memory. IMDs may employ multiple processor for different computing jobs (e.g., combination of DSPs and normal processing units). In order to provide the processors with a storage place, the implant SoC contains a memory module.

Another necessary IMD feature is the device-human communication. An external user should be able to monitor and control the IMD, so he should be able to send commands to the IMD and the IMD transmit physiological data back to the external user. A coil is used by an external reader to communicate with the IMD.

All the above components are powered by the main IMD battery. Sometimes, the same coil used for information broadcasting is also used for power transfer through living tissue [7].

2.2 System On Chip Interconnect Topology

The topology of the SoC interconnect is defined by the arrangement of the communication elements and links (bundle of wires that carries a signal). Based on the communication needs of IMDs for low power consumption, low area cost and high performance we next describe 3 basic interconnect topologies that we believe to most likely serve the communication needs of an IMD: A Bus, crossbar and point-to-point architecture. Unfortunately, time limitations have prevented us from evaluating all 3 interconnect types in this thesis and we have excluded the bus architecture from our analysis. The evaluation of the bus architecture is, therefore, reserved for future work.

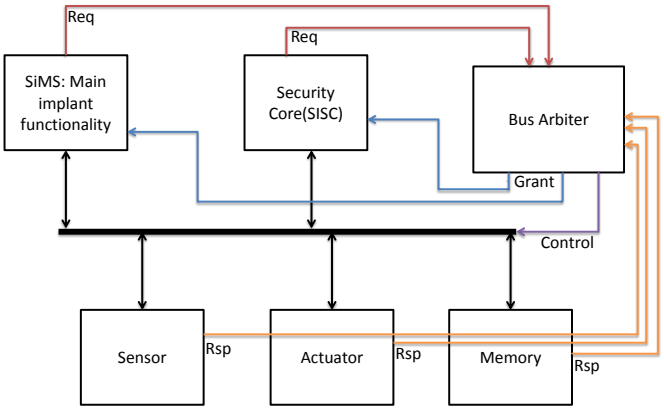


Figure 2.2: An example of the bus topology.

Bus

The bus topology is a network setup in which each component is connected to a shared medium called bus (see Figure 2.2). Each of these components are able either to place signals on the bus or sense the signals already present on it. Each time only one of these entities can place signals on the bus, while every other entity senses these signals and, depending on the content of the signals, may choose to accept or discard them. Since a bus is a shared medium that allows only a single entity to use it at a time, an arbitration protocol is required to decide which of these entities will be granted. The most common arbitration protocol employs a central arbiter. The entities must send their bus requests to the arbiter and the arbiter is defining which of these will been granted based on the arbitration protocol.

The advantages of a shared bus architecture include simple topology, low area cost and extensibility [1]. However, as the number of entities attached to the bus increases, the bus speed decreases because of the higher capacitive load that it has to drive to a longer distance [2]. Furthermore, the more components are connected with it, the more will grow the wait-time for each of them because of their competition and that limits their maximum throughput. Despite the above timing limitations, we believe that a bus would be a good alternative for IMDs with limited number of components because of its low power consumption.

Crossbar

An NxM crossbar (N masters x M slaves) provides a direct path from every master to every slave. At every intersection of these paths a switch serves as a crosspoint connector, if enabled, connecting the master to the slave. Any slave can be connected to any master by simply closing the switch connecting the master and the slave (see Figure 2.3). However, for correct operation, each slave must be connected to at most one master. This restriction is accomplished by a central arbiter which decides which master will be granted with permission and configures the switches respectively. We refer to the collection of these switches at cross points as the ”fabric” of the crossbar.

The main advantage of the crossbar is that it can potentially serve multiple masters at a time (using switches) as soon as each slave remains connected with maximum one master. Consequently, it provides a higher throughput than the bus environment. Moreover, as the traffic between any two devices increases, it does not affect traffic between other devices. The biggest disadvantage of a crossbar is its

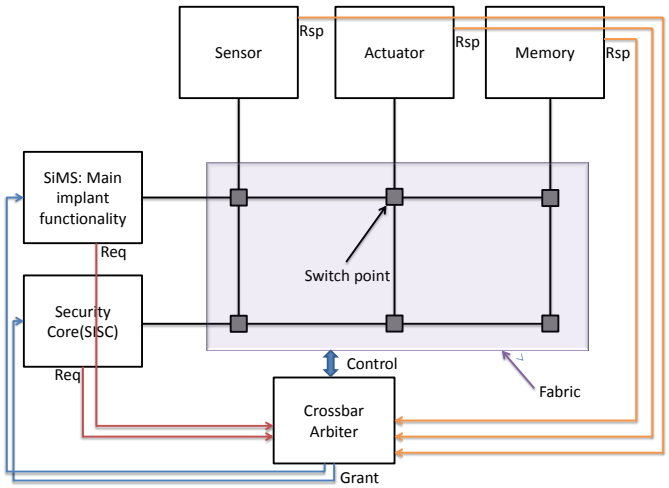


Figure 2.3: An example of the crossbar topology.

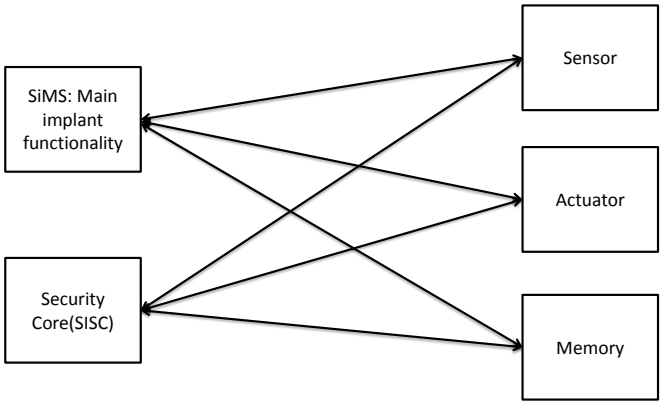


Figure 2.4: An example of a fully connected point-to-point topology.

cost, which grows proportional to $N * M$, where N+M is the number of components [2]. However, cost may be acceptable for a few number of components.

Point to point

The point-to-point architecture is one of the simplest interconnects, in which two components that need to communicate with each other have a dedicated link (Figure 2.4). There are two types of point-to-point: fully connected interconnect and custom interconnect. The fully connected type connects every master with every slave while the custom one is designed according to the needs of the masters. The main advantage of the point-to-point is that its throughput remains stable no matter how many components it connects because the different modules do not compete each other. However, the interconnect needs a set of wires to connect different entities together and as many interfaces as the number of entities that need to communicate. As a result, the use of wires, the number of interfaces and complexity thereof increase rapidly as the number of components increases.

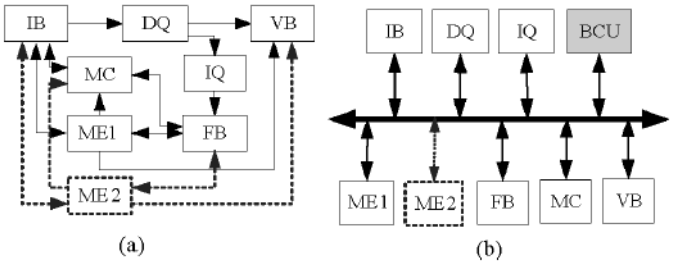


Figure 2.5: Implementation of MPEG-2 encoder with 2 MEs using: (a) P2P and (b) bus communication architectures

2.3 Related work

To the best of our knowledge, there is not any related work concerning the use of different interconnect topologies that specifically target Implantable Medical Devices or implant SoCs. Nevertheless, we next discuss research concerning the comparison of crossbar with bus and point-to-point with bus implemented in different low-power SoC applications. We believe that the main topologies' tendencies, extracted from these projects, apply to implant SoCs too. Despite the fact that our thesis does not involve a bus interconnect implementation, the information below can offer us a first-order view of the consequences of replacing the point-to-point or crossbar with a bus interconnect.

2.3.1 Bus versus Point-to-point

For the bus and point-to-point comparison, we follow a study that is implementing bus-based and P2P-based architectures of a real multimedia application (i.e. MPEG-2 encoder) using an FPGA prototype (Xilinx Virtex2 3000) and actual video clips [8].

The implemenation of a basic MPEG-2 encoder consists of 7 modules/cores: (1) input buffer (IB); (2) DCT and quantization(DQ); (3) variable length encoder and output buffer (VE); (4) motion compensation (MC); (5) inverse quantization and inverse DCT (IQ); (6) motion estimation (ME); and (7) reconstructed frame buffer (FB) (see Figure 2.5). In a P2P-based implementation these modules are communicating with each other through dedicated channels while in a bus-based through a shared link. The bus-based implementation requires one extra more module, the BCU (arbiter that determines which module has the right to access the shared bus).

The study concludes that the point-to-point throughput is bigger than the bus one because in point-to-point architecture each component use its own dedicated channel so there is no contention. If we add one extra ME module this difference gets even bigger. On the other hand, the point-to-point scales worse than the bus in terms of area because of the extra interfaces and links it requires when the number of components increases. In contrast, the bus is only local affected since only one more link and one more port from the newly inserted module to the bus/router is added. Lastly, the point-to-point has a larger energy consumption than the bus implementation concerning an MPEG-2 encoder with only one ME module. However, as we add more ME modules and thus the degree of parallelism is increased, this fact reverses because point-to-point may have more links and interfaces however the bus needs longer time to encode real data. On the other hand, the power consumption of the P2P implementation is larger than the power consumed by the bus. The power consumption of the P2P architecture scales

poorly as the degree of parallelism increases, since the P2P implementation requires a significantly larger number of additional links and network interfaces to accommodate the addition of extra ME modules than the crossbar.

Based on the above research we believe that a bus-based implementation suits better for IMDs than a P2P-based because of its smaller size and lower power and energy consumption. However, a P2P implementation may be justified if the medical application demands higher throughput.

2.3.2 Crossbar versus Bus

We could find two projects that compare the bus with the crossbar architecture [9][10].

The first project [9] synthesized different bus and crossbar architectures with 64-bit width to STMicroelectronics 0.18 µm. They are implemented utilizing a few number of components. The interface between the agent (master) and the interconnect is a common input/output block that contains two FIFO buffers (one from agent to interconnection and one from interconnection to agent) and a state machine controlling the data transmissions. The transmissions are done with data packets containing one data valid bit, an address field and a data field.

For the performance measurement, uniform traffic patterns have been used, meaning that all components transmit the same amount of data to all other components in a round robin way. As expected, it has been proven that the crossbar area is bigger than the bus and as the number of components increases this difference is getting bigger. As it matters the throughput, it seems that the bus is efficient only for small length transmissions (8 transfers) and a few number of agents (< 5 agents). Thus, when the number of agents increases (approximately above 4 agents) the crossbar outperforms the bus.

The second project [10] synthesized crossbar and bus designs in 0.35 micron CMOS technology. The bus and the crossbar designs are build both based on MUXes in order to make them more comparable. This paper does not include a scheduler in either design, the data transfer preprocessing is done manually.

The experiments are based on two different sets of data. The first set is artificially generated. With this dataset, the crossbar can operate in full parallelism and is called “perfect data” set. The second data set is pseudo randomly generated, which we call “random data”. The simulation of the perfect data set proved that the crossbar consumes more energy per cycle than the bus with the same number of components (simulations have taken place with 4, 8 and 16 components). However, crossbar consumes less energy per data because of its high throughput and because it can operate in full parallelism. With the second data set, the “random” one, in contrast with the previous results, the crossbar consumes more energy both per data and per cycle than the bus.

We believe that for IMDs that can operate in full parallelism and demand medium-high throughput the hardware complexity of the crossbar is justified.

CHAPTER 3

Implementation

We previously identified three types of interconnect that are well suited for the needs and constraints of IMDS: A bus, P2P and crossbar architecture. Due to time limitations, we have only implemented the latter two (the implementation and comparison to a bus is reserved for related work). We next describe the design of both interconnects. We must highlight that their design is generic and can be adjusted to any number of slaves, number of masters, data-width and address-width with minor effort.

3.1 Crossbar design

In this section we describe the design of the crossbar. The overall view of our N x M bidirectional crossbar design is given in Figure 3.1. The input signals to the crossbar interconnect are N *Master_message*, N *Master_request* and M *Slave_data* signals. The output signals are: M *Master_message*, N *Busy*, N *Response*, N *Slave_message* and M *Request* signals. Our crossbar interconnect is adjustable and flexible and can be scaled up or down by simply changing the following parameters: number of Masters,number of Slaves, data width and address width.

The *clock* and *reset* inputs are global. The *reset* input reset all the flip flops used in the design and it is asynchronous low because it requires less gates to implement. The *Master_message*, *Slave_message* and *Slave_data* signals are used for message transmission while the rest of them are control signals. We next discuss the typical procedure followed to exchange a message between a master and slave, after which we describe the involved components in more detail. A single message exchange between a master and slave occurs as follows:

Phase 1 - Arbitration The master initiates a new message/request transfer. The master interface stores the master message and is requesting permission from the arbiter to transmit the message to the desired slave/slaves (signal *slave_request*).

Phase 2 - Transfer If the master interface is granted access to exchange a message by the arbiter (based on the arbitration protocol) then the message is transmitted through the fabric and the slave interface/interfaces to the desired slave/slaves. If the master interface is not granted access, then the master interface requests again permission from the arbiter.

Phase 3 - Response If the message received by the slave was ”read” type, then the slave responds with data and the slave message/response is transmitted through the fabric and the master interface to

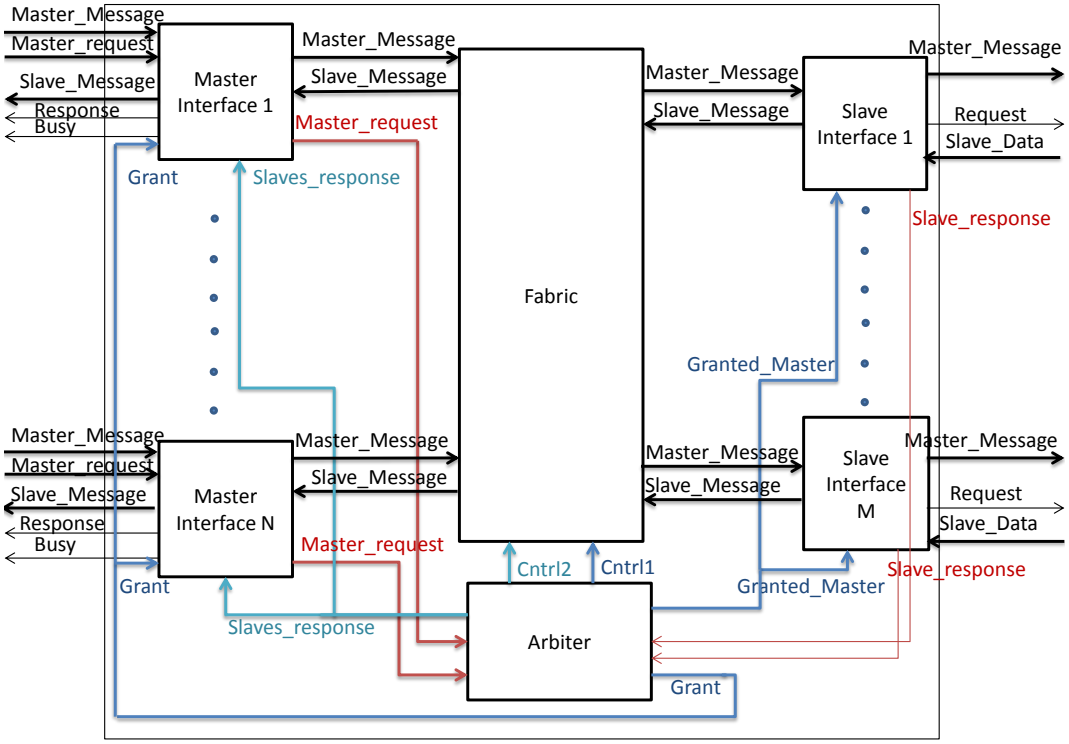


Figure 3.1: Crossbar switch NxM.

the master. The slave message contains the data received by the slave and the address of the initial request. This response takes place immediately after the reception of the ”read” type message.

3.1.1 Master interface

Between each master and the main crossbar (fabric and arbiter) we have the ”Master Interface” component. The master interface is responsible to inform the master about the availability of the interconnect, store the message of the master, deliver slaves messages (response from any slave) to the master and in general manage all the communication between the master and the interconnect. The overview of master interface is shown at Figure 3.2.

The masters→slaves communication is based on the following logical operations. If the master interface is available (*Busy*=0) then he accepts and stores the next master’s message (*Data*, *Address*, *RW*) and requests the arbiter for permission (*Master_request*) to transmit the message to the desired slave/slaves. The *Busy* signal comes from a comparison between the requested slave/slaves (*master_request_stored*) and the slave/slaves that he receives permission to transmit the message (*granted_master*). If they are equal that means the master interface is granted for all the slaves he requested. Otherwise, the master holds his request until master interface is available.

Regarding the slaves→masters communication, the master interface is notified about any responses (*Slaves_responses*) by the crossbar arbiter. If any slave has to deliver a response to the master then the master is notified (*Response*=1).

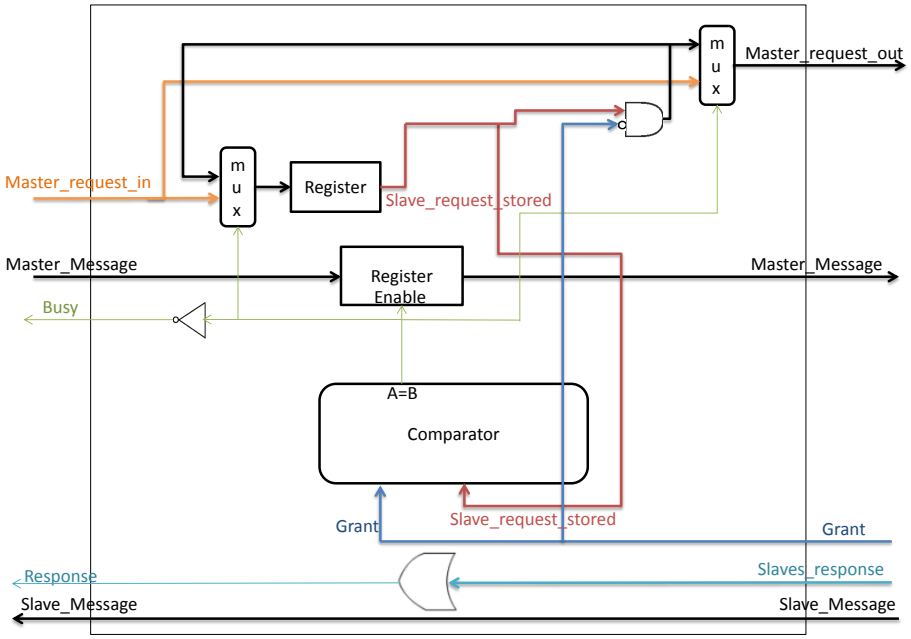


Figure 3.2: The Master interface module.

3.1.2 Crossbar arbiter

In the crossbar interconnect, there is a case that multiple masters would request the same slave. However, this collision cannot be permitted because every slave should be connected with only one master at a time. The component that prevents these collisions from taking place is called crossbar "arbiter" or "scheduler". This component considers the requests from all masters and grants access to them. In order to perform the arbitration, the crossbar arbiter emploies a scheduling algorithm that decides which Masters will be granted with permission. We chose a round robin algorithm for our implementation, i.e., the arbiter gives permission to the masters which are requesting the same slave in a circular order.

In effect, the crossbar arbiter consists of M basic arbiter modules, one for each slave. As a result each slave has each own "personal" arbiter and if a master requests more that one slaves then he may be granted access only to some of them, because each basic arbiter module is independent. The architecture of the basic arbiter module is shown in Figure 3.3. The "logic" box of this module's architecture is implementing the round robin arbitration following the logical operations bellow. It has only two inputs: the master that desire to communicate with this slave (*Request*) and the (*Last_grant*) last master granted with permission.

$$Gnt <= req\,and\,(not(req)) + 1; \tag{1}$$

$$reqs <= req\,and\,not((Last\_grant–1)\,or\,Last\_grant); \tag{2}$$

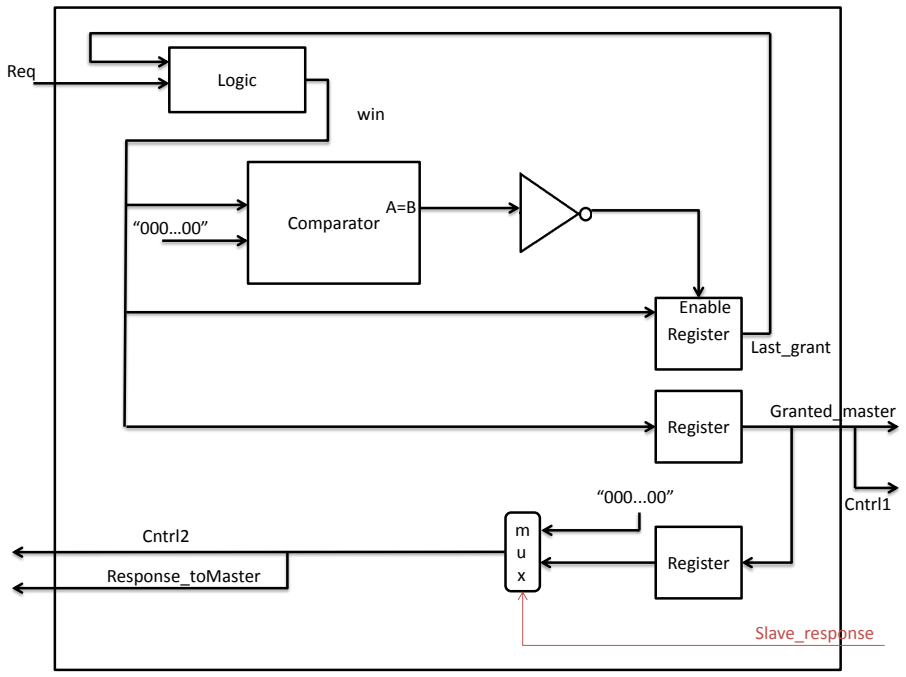


Figure 3.3: The basic arbiter module architecture of our Crossbar arbiter.It is using a round-robin protocol.

$$gnts <= reqs\, and(not(reqs)) + 1; \tag{3}$$

$$Grant <= gnts\, when\, reqs/ = 0\, else\, gnt; \tag{4}$$

3.1.3 Fabric

The crossbar fabric component in the design (shown in figure 6.1) is responsible for physically connecting a master to its destined slave and vice versa, based on the grants issued by the scheduler. Our bidirectional crossbar fabric consists of two fabric modules, one for each way of communication (masters→slaves, slaves→masters). The fabric structure is an assembly of individual switches between N inputs and M outputs. The switches are arranged in a matrix, i.e., an NxM bidirectional crossbar fabric is implemented by NxM switches for the masters→slaves communication and by MxN switches for the slaves→masters communication. The crossbar fabric's function is pretty simple: if a master is granted or a slave has to respond, then it uses set of logical gates to set corresponding connection between master and slave, as controlled by the arbiter (*Cntrl* signals).

3.1.4 The slave interface

Between each slave and the fabric we have a "slave interface". The slave interface is responsible for delivering the master's message to the slave, handle the data response (if the request was "read" type) and in general manage all the communication between the slave and the interconnect. The overview of

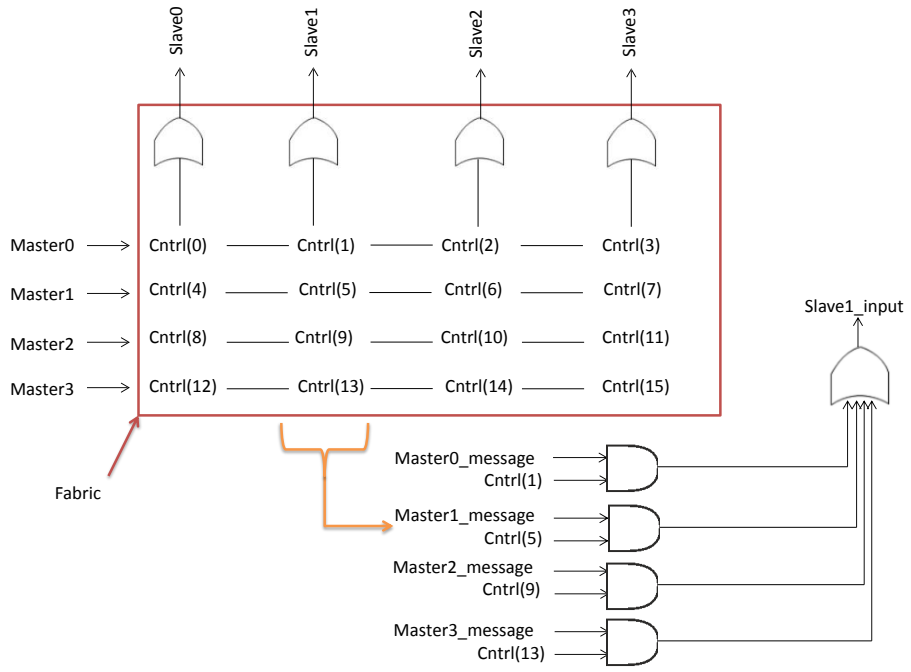


Figure 3.4: An example of a 4x4 fabric module. Each bit of *cntrl* input corresponds to a certain switch. Each of the crossbar's fabric output is the logical sum (OR) of inputs 0 to 3 AND'd with the cntrl lines of that output's column.

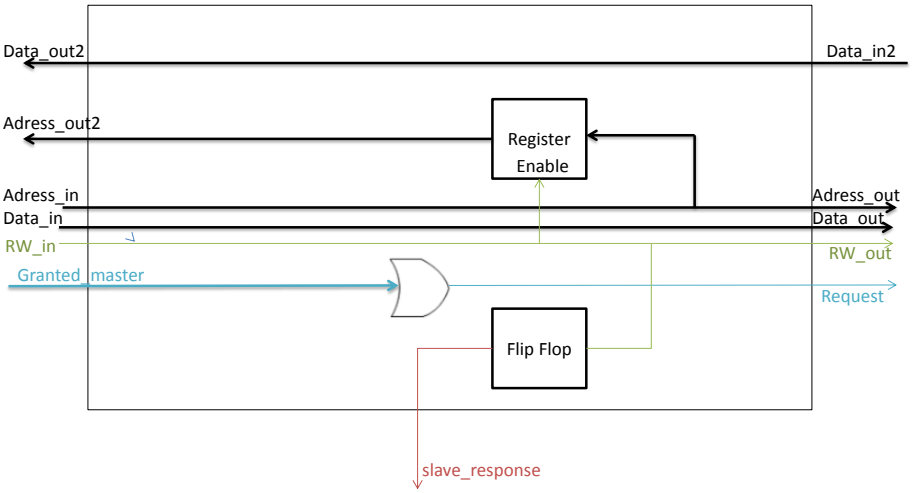


Figure 3.5: The Slave interface module.

a slave interface is shown at Figure 3.5.

The slave interface function is based on the following procedure. The slave interface has to take the master's message from the fabric (*Data*, *Address* and *RW*) and deliver it to its corresponding slave. The slave interface may be notified about an upcoming message by the crossbar arbiter (*Granted_master*), if so, then it notifies the slave *Request*). In the next cycle, if the request was of "read" type, the slave responds with data and the slave interface forwards the response to the fabric.

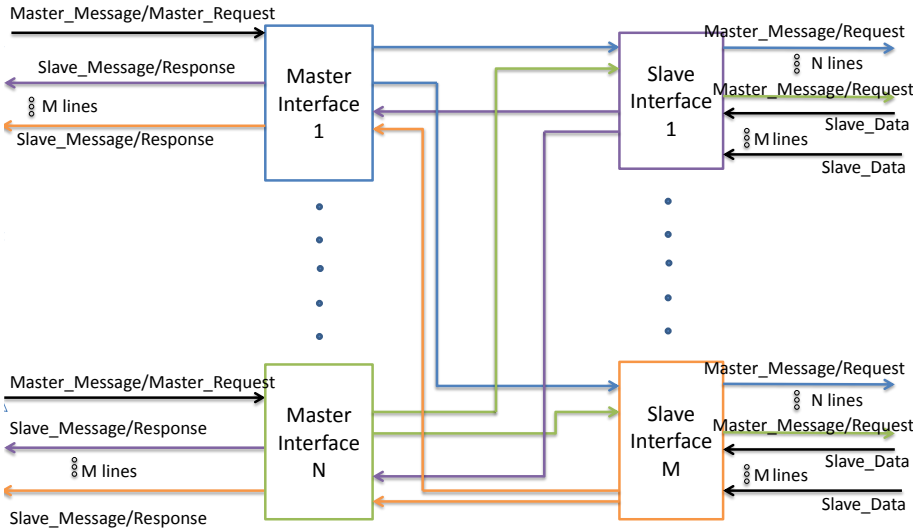


Figure 3.6: Point to point interconnect design.

3.2 Point-to-point Design

Similar to the analysis followed before, we first give an overall view of what is implemented and then we move on to a further analysis of each component. The overall view of our NxM point-to-point design is given in Figure 3.6 . The input signals of the point-to-point interconnect are N *Master_Message*, N *Slave_request* and M*N *data* signals. The output signals of the point-to-point are: N*M *Master_Message*, N*M *Slave_Message*, N*M *Request* and N*M *Response* signals. Our point-to-point interconnect is adjustable and flexible and can be scaled up or down by simply changing the following parameters: number of Masters,number of Slaves, data width and address width.

The clock and reset inputs are global. The reset input reset all the flip flops used in the design and it is asynchronous low because it requires less gates to implement. The *Master_message*, *Slave_message* and *Slave_data* signals are used for message transmission while the rest of them are control signals. As in the crossbar case, we describe first the typical procedure followed to exchange a message between a master and slave before we describe the involved components in more detail. A single message exchange between a master and slave occurs as follows:

Phase 1 - Transfer The master initiates a new message/request transfer. The master interface accepts the master request and delivers the message to the desired slave through the slave interface.

Phase 2 - Response If the message received by the slave was ”read” type, then the slave responds with data and the slave interface forwards the response to the master through the master interface. We assume that this response takes place immediately after the reception of the ”read” type message.

3.2.1 Master / slave interface

The master/slave interface components are effectively implemented by wires while the slave interface has also some extra flip flops to buffer the request (if the request was of ”read” type). They are responsible to transmit the master message to the desired slave and the slave response to the master

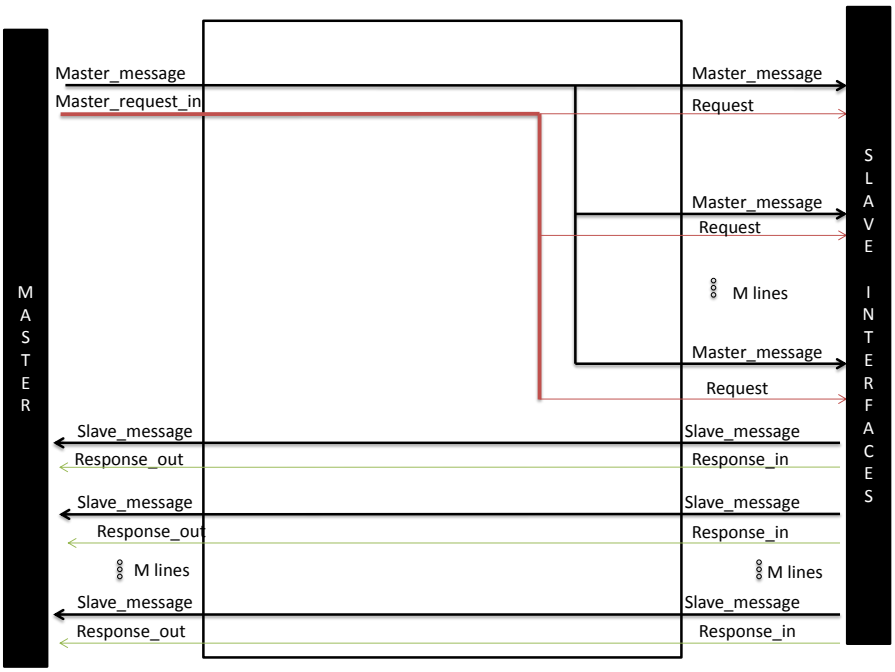


Figure 3.7: The master interface module.

respectively. The overview of the master interface is shown at Figure 3.7. As in the crossbar case, we need also some more ”control” signals to notify the slave about upcoming messages (*request*) and to notify the master about upcoming responses (*Response_in* and *response_out*).

CHAPTER 4

Evaluation

We are next evaluating our interconnects based on various quality measures and traffic patterns. First, the experimental setup is described in 4.1, after which our evaluation ensues in 4.2.

4.1 Experimental Setup

In this section we first describe the figures of merit we considered for the designs evaluation, after which we describe what and how parameters we vary as part of our evaluation.

4.1.1 Figures of merit

For the evaluation of our implementations, we decided to measure 4 different characteristics: throughput, latency, area cost and power consumption, discussed next in order.

Throughput

Throughput is defined as the average number of served requests per cycle. It is measured by counting the total number of requests that the interconnect serves in a specific amount of time and dividing the total requests served by the amount of time we measured. We count the total number of served requests using ”Modelsim” simulations. The measured throughput of the interconnect depends on the traffic it serves while the maximum throughput depends on the interconnect design.

Latency

Latency is defined as the average time the interconnect requires to complete a message exchange. The latency of a message exchange is measured by seeing how many cycles it takes to be completed after it was first requested by a master using ”Modelsim” simulations. For a crossbar interconnect, the minimum latency of a message exchange is 3 cycles (it has at least to complete the 3 phases). It may need more than 3 cycles in case that the request conflicts with other requests. On the other side, for a point-to-point interconnect the latency is always stable and equals to 2 cycles.

Area

The area factor defines the space consumed by the implemented interconnect and it depends on the number of cells used and the type of the cells. In order to measure it we used the ”Synopsys De-

sign Compiler (DC)" tool to perform hardware synthesis. This synthesis tool takes an RTL hardware description and a standard cell library as input (we chose the UMC 90nm library due to availability) and produces a gate-level netlist as output. The resulting gate-level netlist is a completely structural description with standard cells. The area metric is in units specific to the standard cell library (which is μm^2 in our case).

Power

Power is the energy dissipated in a device per unit of time. The total power dissipated in a CMOS VLSI can be classified in two major categories:

- Static or leakage power which is dissipated when the interconnect is at steady state.
- Dynamic power which is dissipated when the device is switching.

The total power consumption can be calculated as: $P_{total} = P_{static} + P_{dynamic}$. In order to measure the average power consumption of our interconnects we used the "Synopsys PrimeTime" tool. For the power analysis, the design data and the switching activity are required. The design data are provided by the synthesis of our design and the switching activity by the vcd file, which is produced from the simulation of the traffic pattern. For our measurements we set the clock frequency at 20MHz since this is commonly assumed suitable for IMDs.

4.1.2 Evaluation parameters

As we mentioned in the previous chapter, the design of both of our crossbar and point-to-point interconnect is flexible. It can be adjusted to any number of masters and slaves and any transfer width (data and address fields) by changing a few parameters in VHDL.

Inline with the expected sizes of current and future IMDs, we choose to implement interconnects of sizes: 2x2, 2x4, 2x8, 2x16, 2x32, 4x4, 4x8, 4x16 and 4x32. A 2xM interconnect size is a logic hypothesis since 2x2-2x8 interconnect sizes are likely realistic scenarios nowdays (see IMDs described in the Appendix B). While 32 slaves are likely too many for IMDs in the near future, we evaluate the effect to assess its viability for Implants. Moreover, there are some modern IMDs which are using more than two processors (e.g. a responsive implantable system for the treatment of neurological disorders [3] is using 1 processor and 2 sub-processors for the event detection and the stimulation of the system), thus we believe that a 4xM evaluation is a realistic choice too. The above sizes contribute to a consistent view of the interconnects' scalability in terms of performance, area cost and power consumption.

Our default interconnect assumes an 8-bit data width and a 32-bit address. To determine how our system overheads scale as a function of these parameters, we also implemented interconnects with 4-bit data and 16-bit address. In these latter interconnects (4-bit data,16-bit address) an extra split and reconstruct mechanism is implemented within the master and slave interfaces. This mechanism split the request message in order to transfer it and reconstruct it before delivering it to the slave. The same occurs for a response.

Table 4.1: Random traffic patterns

Pattern names	Generation rate	Potential Collisions
L0	Low($\frac{2\ requests}{1000\ cycles}$)	0%
L25		25%
L50		50%
L75		75%
L100		100%
M0	Medium($\frac{2\ requests}{100\ cycles}$)	0%
M25		25%
M50		50%
M75		75%
M100		100%
H0	High($\frac{2\ requests}{10\ cycles}$)	0%
H25		25%
H50		50%
H75		75%
H100		100%

4.1.3 Traffic patterns

The traffic pattern indicates the destination and the generation rate (the bigger it is the more requests our interconnect has to serve) of masters' requests. Based on our attempt to evaluate our interconnects for a wide range of communication scenarios, we created different "random" traffic patterns. Our traffic patterns are defined by their generation rate (low, medium or high) and their "potential collisions" percentage, which defines the percentage of collissions expected. We are interested in the above specifications because they clearly indicate the influence of many different communication scenarios to the interconnect's performance and power consumption. The type of masters' messages/requests are half "read" and half "write" type.

4.2 Experimental results

In the following sections we present and analyse the results of our experiments. We start by showing the results of each individual interconnect type which were produced by executing the mentioned traffic patterns. Afterwards, we compare the results for both interconnects.

4.2.1 Crossbar

Throughput

Figures 4.1 and 4.2 depict the throughout results for different traffic patterns and SoC sizes. If the number of requests per second does not exceed the saturation point of the interconnect, the throughput (accepted traffic) equals the demand (offered traffic). We observe that none of our 15 "random" traffic patterns exceeds the saturation point of each implemented crossbar. As a result, the measured throughput of our implemented crossbars depends neither on the percentage of collisions, nor the size of them, but only on the traffic density they have to serve. For example, a 2x16 has the same measured throughput for every of the L0-L100 random traffic patterns (the same applies to the M0-M100 and H0-

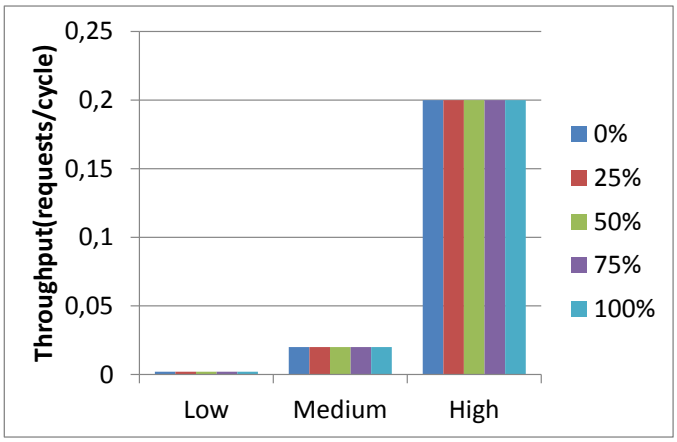


Figure 4.1: An example of a 2x16 crossbar with all the random traffic patterns.

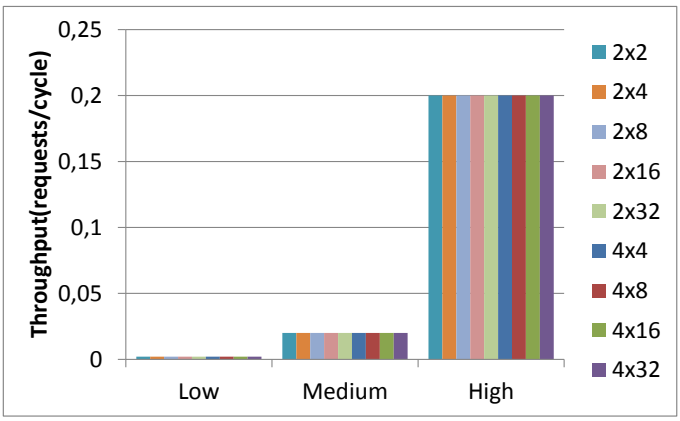


Figure 4.2: Different kind of density traffic patterns (L0, M0, H0) served by every crossbar size.

H100 traffic patterns, see Figure 4.1). Moreover, for all the implemented crossbar sizes we got the same measured throughput when the same random traffic pattern was served (see Figure 4.2). It is expected that for traffic patterns with very high regeneration rate, the throughput may depend on the collisions percentage and may not equal the demand (traffic), e.g., a traffic pattern with $\frac{2\ requests}{cycle}$. Finally, we note that by reducing the transfer capability of the crossbars to 4-bits data and 16-bits address width, we get the same measurements. However, we expect that the maximum potential throughput is decreased (as it depend on the design).

Latency

Figure 4.3 depict the latency results for different traffic patterns and SoC sizes. Our results proves that latency is affected by conflicts, while it is irrespective of traffic density (see Figure 4.3). Moreover, the latency is stable for a given collision rate regardless of SoC size because the procedure followed for a request service is not affected by the number of components. Finally, by halving transfer width we observe that the latency is increased by two cycles because only half of the initial request can be transferred through the fabric at a time.

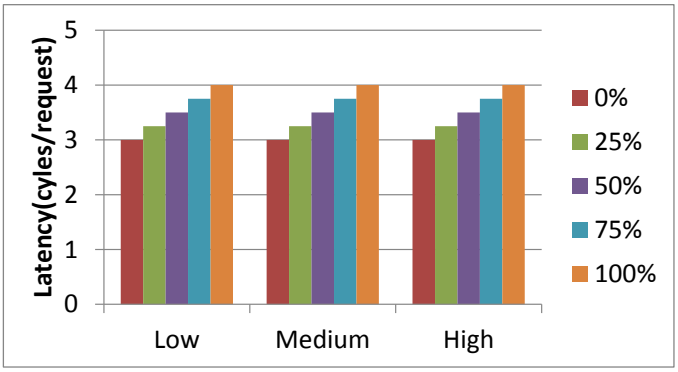


Figure 4.3: An example of a 2x16 crossbar with all the random traffic patterns.

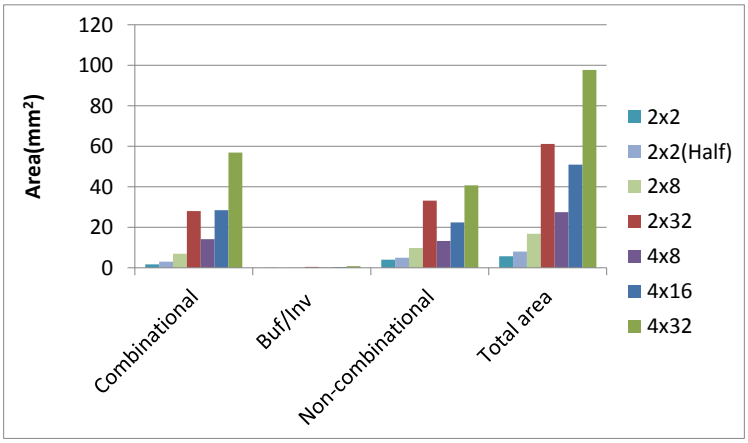


Figure 4.4: Crossbar utilization area of the different implemented sizes.

Area

Figure 4.4 depicts the area results for various SoC designs. By implementing the 2x2 - 4x32 crossbars we could get a clear and complete picture of the crossbar's area scalability. We observe that the crossbar's needs of area tend to be increased when the number of the slaves increases. For example, moving on from a 2x2 to 2x4 crossbar results to a 64% increase in area, while moving on from a 2x16 to a 2x32 results to a 93,4% increase (see Figure 4.4). The same applies to the 4xM sizes. Therefore crossbar loses its scalability as size increases. Primarily combinational logic is increased because the fabric is increased proportional to $N * M$. Finally, we note that by halving the crossbar's transfer width, the total area is increased. This is due to the extra logic needed for the implementation of the split and reconstruct function. For example, the 2x2 crossbar's area is 30,4% bigger than our previous crossbar implementation.

Power

Figures 4.5 and 4.6 depict the power-consumption results for all the crossbar sizes. We first observe that total power is primarily affected by size. This is because both static power and dynamic power (consumed for the service of a request) are increased. The same applies to peak power (see Figure 4.6). Moreover, we observe that total power is hardly affected by collisions because static power remains the same while the dynamic power may slightly increase. For every traffic pattern and crossbar size,

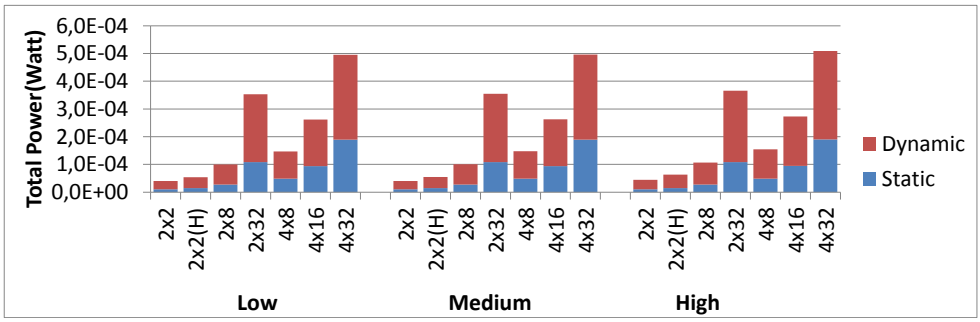


Figure 4.5: The average power consumption of different kind of density traffic patterns with 25% collisions.

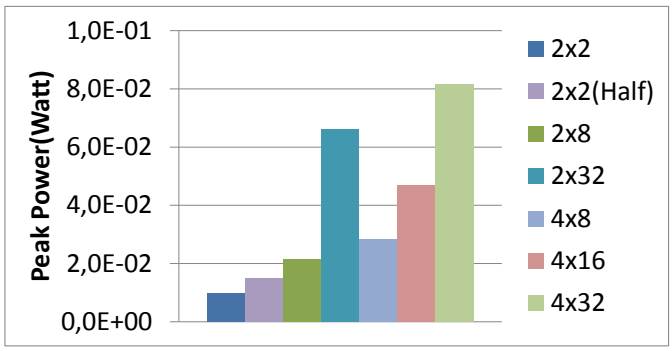


Figure 4.6: The peak power consumption for different crossbar sizes that serve the M25 traffic pattern.

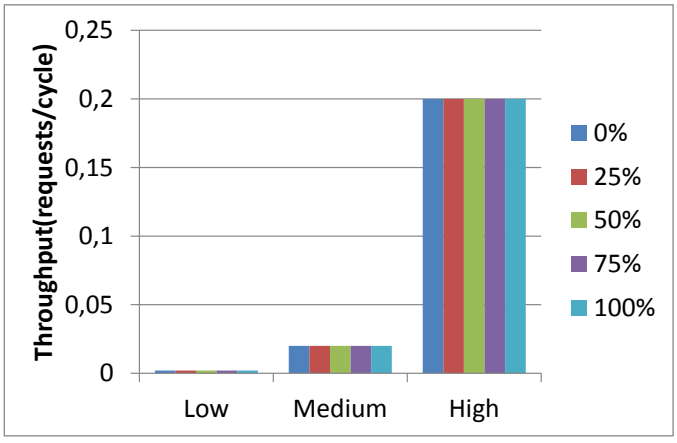


Figure 4.7: An example of a 2x16 point-to-point with all the random traffic patterns.

we notice that dynamic power is always a lot higher than static, even if the communication activity is low. Finally, we notice that by reducing by half the crossbar's width the total power consumption is increased due to the the split-reconstruct overheads. For example, a 2x2 crossbar that is simulating the L25 traffic pattern is consuming 35,73% more power than our previous crossbar.

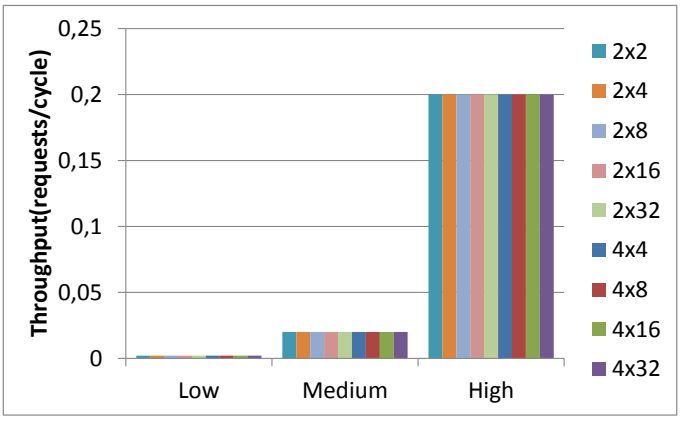


Figure 4.8: Different kind of density traffic patterns (L0, M0, H0) served by every point-to-point size.

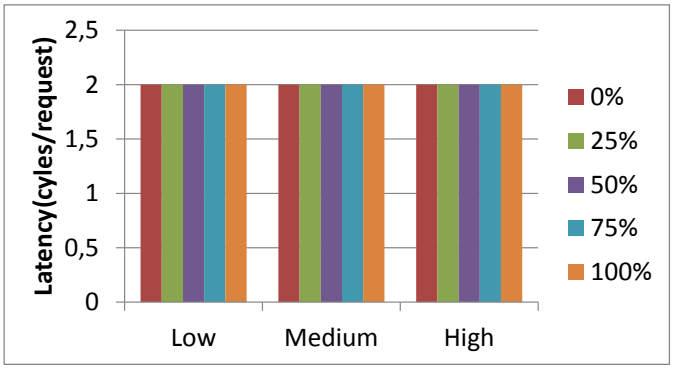


Figure 4.9: An example of a 2x16 point-to-point with all the random traffic patterns.

4.2.2 Point-to-Point

Throughput

Figures 4.7 and 4.8 depict the throughout results for different traffic patterns and SoC sizes. We observe that none of our 15 ”random” traffic patterns exceeds the saturation point of each implemented P2P. As a result, the measured throughput of our implemented P2Ps depends neither on the percentage of collisions, nor the size of them, but only on the traffic density they have to serve. For example, a 2x16 point-to-point has the same measured throughput for every of the L0-L100 random traffic patterns (the same applies to the M0-M100 and H0-H100 traffic patterns, see Figure 4.7). Moreover, for all the implemented P2P sizes we got the same measured throughput when the same random traffic pattern was served (see Figure 4.8). It is expected that even for traffic patterns with very high regeneration rate, the throughput would equal the demand. Finally, we note that by reducing the transfer capability of P2P to 4-bits data and 16-bits address width, we get the same measurements. However, we expect that the maximum potential throughput is decreased (as it depends on the design).

Latency

Figure 4.9 depicts the latency results for different traffic patterns and SoC sizes. Our results prove that conflicts do not affect latency nor the traffic density (see Figure 4.9). Moreover, the latency is stable regardless of SoC size because the procedure followed for a request service is not affected by

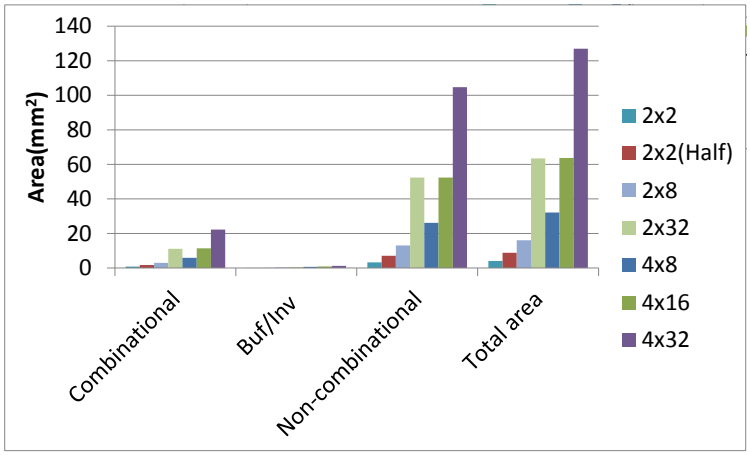


Figure 4.10: Point-to-point utilization area of the different implemented sizes.

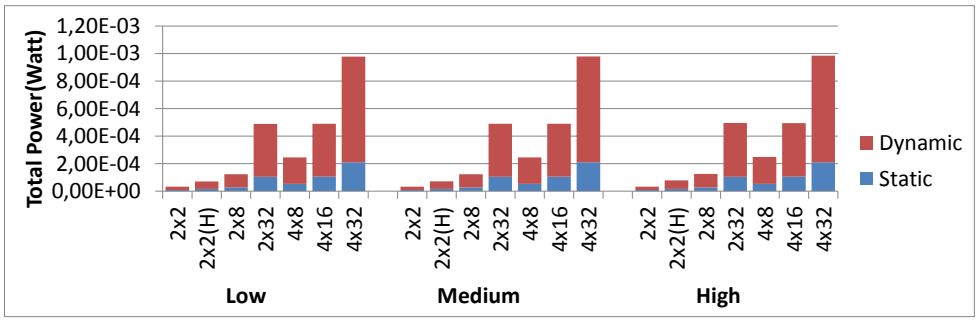


Figure 4.11: The average power consumption of different kind of density traffic patterns with 25% collisions.

the number of components. Finally, by halving transfer width we observe that the latency is increased by two cycles because only half of the initial request can be transferred at a time.

Area

Figure 4.10 depicts the area results for various SoC designs. By implementing the 2x2 - 4x32 P2Ps we could get a clear and complete picture of the P2P's area scalability. We observe that the P2P's needs of area tend to be increased when the number of the slaves increases. For example, moving on from a 2x2 to 2x4 P2P results to a 98,6% increase in area, while moving on from a 2x16 to a 2x32 P2P results to a 99,2% increase (see Figure 4.10). The same applies to the 4xM sizes. Therefore P2P loses its scalability as size increases. Primarily non-combinational logic is increased because of the many registers it requires to handle many request/responses at a time. Finally, we note that by halving the P2P's transfer width, the total area is increased. This is due to the extra logic needed for the implementation of the split and reconstruct function. For example, the 2x2 P2P's area is 110% bigger than our previous P2P implementation.

Power

Figures 4.11 and 4.12 depict the power-consumption results for all the P2P sizes. We first observe that total power is primarily affected by size. This is because both static power and dynamic power

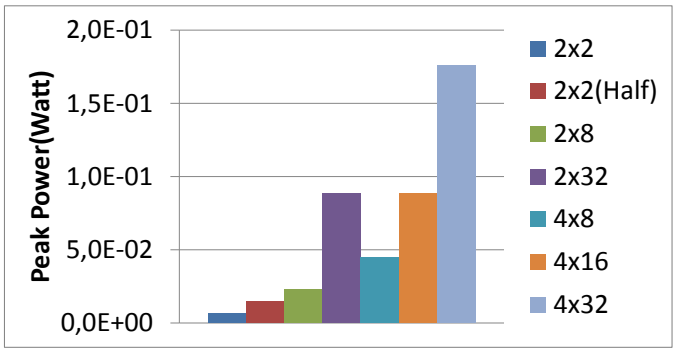


Figure 4.12: The peak power consumption for different point-to-point sizes that serve the M25 traffic pattern.

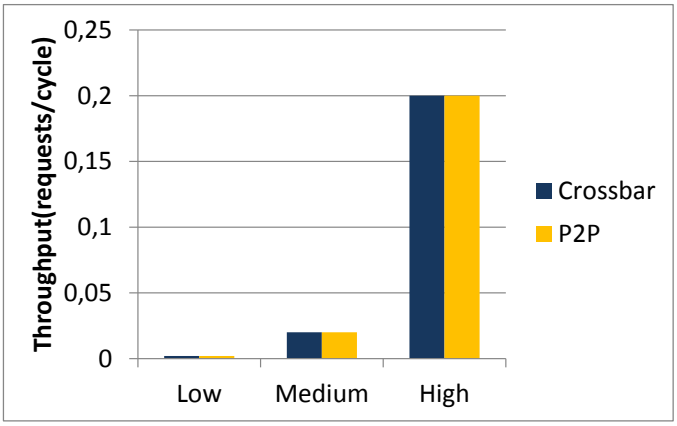


Figure 4.13: An example of a 2x16 crossbar and a 2x16 point-to-point serving the L0,M0 and H0 traffic patterns.

(consumed for the service of a request) are increased. The same applies to peak power (see Figure 4.12). Moreover, we observe that total power is not affected by collisions because both static and dynamic power remain the same. For every traffic pattern and P2P size, we notice that dynamic power is always a lot higher than static, even if the communication activity is low. Finally, we notice that by reducing by half the P2P's width the total power consumption is increased due to the the split-reconstruct overheads.

4.2.3 Comparison

Here we attempt a comparison of the crossbar and point-to-point interconnects based on the above measurements.

Throughput

For every traffic pattern and size the measured throughput of both interconnects is the same (see Figure 4.15). As we already mentioned, this occurs because none of our traffic patterns exceeds the saturation points of these two interconnects. However, we expect that the point-to-point's saturation point is bigger than the crossbar's.

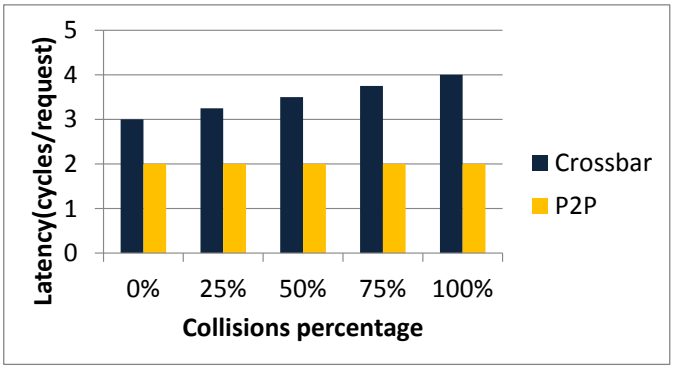


Figure 4.14: An example of a 2x16 crossbar and a 2x16 point-to-point serving the L0-L100 traffic patterns.

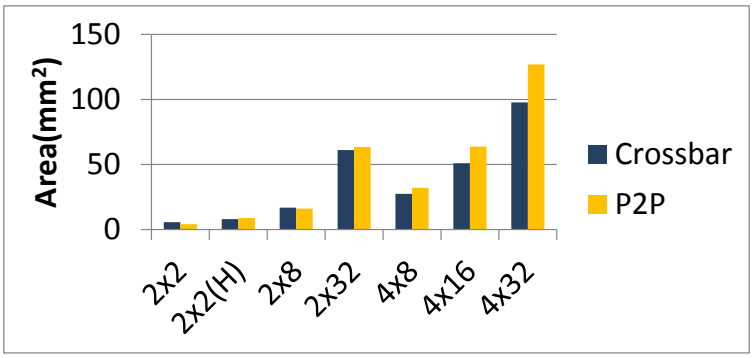


Figure 4.15: Comparison of crossbar and point-to-point area scalability.

Latency

The point-to-point has lower average latency than the crossbar. This is due to crossbar's need for arbitration. Crossbar requires one more cycle for arbitration and its latency may also increase due to collisions (See Figure 4.14).

Area

In terms of area, we find that the point-to-point implementation requires less area for small inter-connect sizes. This is because point-to-point's implementation does not require an arbiter and because its master interface is simpler. Conversely, a crossbar takes advantage of the fabric (less links) and the simpler slave interface (it has to handle only 1 request or response at a time) for larger sizes, resulting in a reduced area. The Crossbar, thus, scales better in terms of area (See figure 4.15).

Power

In all cases, we find that the crossbar consumes considerably less power than the point to point. This is primarily because P2P consumes more dynamic power than the crossbar.

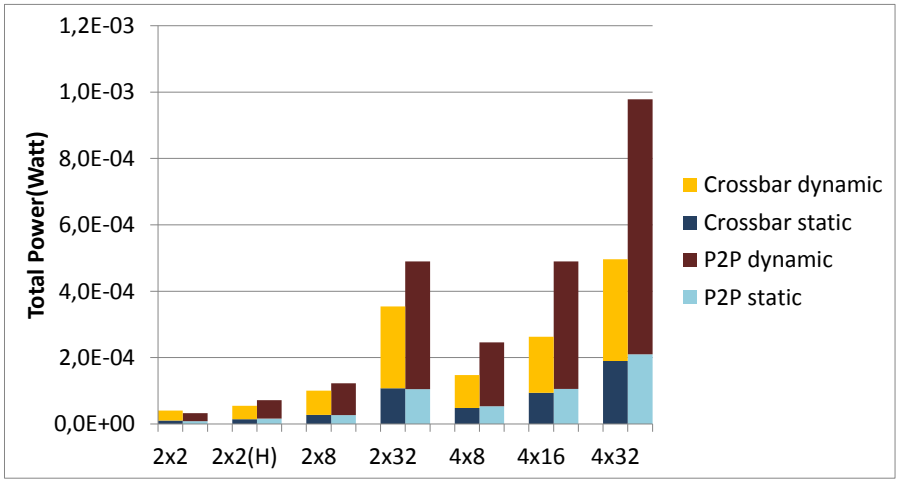


Figure 4.16: The average power consumption of crossbar and point-to-point interconnects during the simulation of the M25 traffic pattern.

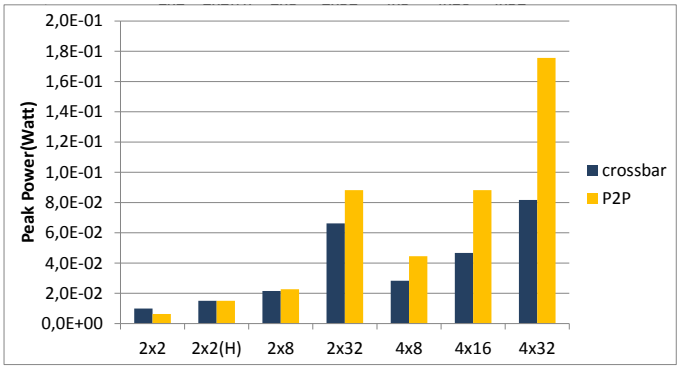


Figure 4.17: The peak power consumption for crossbar and point-to-point sizes that serve the M25 traffic pattern.

CHAPTER 5

Conclusion

5.1 Summary

In this thesis, we were involved with the problem of communication architecture on IMD System-On-Chip. In chapter 2, we presented the structure and communication activity of an IMD, three possible good alternative communication architectures and related work regarding these interconnect types. In chapter 3, we subsequently implemented the two out of the three interconnect types: A crossbar and a point-to-point interconnect. In chapter 4, we evaluated our implementations in terms of throughput, latency, area cost and power consumption. Our experiments show that crossbar has lower performance than the point-to-point, it is smaller for medium-large sizes, it consumes less power and scales better. To answer our question, we conclude that the crossbar type should be used for present and future IMDs instead of a point-to-point, given both their expected traffic patterns and low-power requirements, while the bus seems also a good alternative and thus its implementation is reserved for future work.

5.2 Future Work

We already stated that there is not previous work that examines the impact of different communication architectures designed specifically for Implantable Medical Devices. From this point of view, our work is innovative and also it is only a beginning. There is still much work that can be done, so we present some of our thoughts that may contribute to future research.

- Different topologies can be implemented instead of a crossbar and a point-to-point ones. The bus architecture is the next most promising one because of its potential less power consumption. We expect that, compared with point-to-point and crossbar architectures, a bus architecture would increase the latency of the requests and the maximum throughput would be reduced, but it would decrease the area cost and the total power consumption (based on the related work). The expected increase in latency and the reduction in throughput can likely be tolerated by current and future IMDs.

- Low power methods can be implemented to our existing communication architectures. Two of such methods are already described in Appendix A, their implementation can potentially reduce the dynamic power consumption. However, the effectiveness of these methods depends highly

from the application specifics, so unless we implement them we cannot be sure if the additional circuitry that must be added leads to a total power consumption reduction.

- Fault tolerant techniques should be adopted in order to correct faulty behaviour of components and achieve high reliability. The fault tolerance is of high importance for our IMDs because any communication failure or malfunction may result to serious injury to people or even death. However this feature doesn't come without cost. The adoption of a fault tolerant mechanism will raise both the area cost and the power consumption. This additional cost depends from the communication architecture that is implemented.

Bibliography

[1] R. KAMAL and N. YADAV, "Noc and bus architecture: A comparison," *International Journal of Engineering Science and Technology (IJEST)*, 2012.

[2] B. Moyer, *Real World Multicore Embedded Systems*. Elsevier / Newnes, 2013.

[3] A. R. M. U. Robert E.Fischell, David R. Fischell, Oct. 17, 2000.

[4] N. H. M. Catherine Ward, Susannah Henderson, "A short history on pacemakers," *International Journal of Cardiology*, 2013.

[5] A. Demosthenous, "Advances in microelectronics for implantable medical devices," *Advances in Electronics*, 2014.

[6] K. Bazaka and M. V. Jacob, "Implantable devices: Issues and challenges," *Electronics*, 2013.

[7] C. Strydis, *Universal Processor Architecture for Biomedical Implants*. PhD thesis, Delft University of Technology, 2011.

[8] U. Y. O. Hyung Gyu Lee, Naehyuck Chang and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACMTransactions onDesign Automation of Electronic Systems,Vol. 12, No. 3,Article 23*, August 2007.

[9] K. K. T. H. V. Lahtinen, E. Salminen, "Comparison of synthesized bus and crossbar interconnection architecures," *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, May 2003.

[10] Y. Zhang and M. J. Irwin, "Power and performance comparison of crossbars and buses as on-chip interconnect structures," *Signals, Systems, and Computers, 1999. Conference Record of the Thirty-Third Asilomar Conference on (Volume:1)*, 24 Oct 1999.

[11] L. B. G. D. M. E. M. D. Sciuto and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," *VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on*, Mar 1997.

[12] T. L. Enric Musoll and J. Cortadella, "Working-zone encoding for reducing the energy in microprocessor address buses," *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 6, NO. 4,*, December 1998.

[13] S. P. Hui Guo, "Shifted gray encoding to reduce instruction memory address bus switching for low-power embedded systems," *Journal of Systems Architecture*, 2010.

[14] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power i/o," *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 3, NO. I*, March 1995.

[15] J. Natesan and D. Radhakrishnan, "Shift invert coding (sinv) for low power vlsi," *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, September 2004.

Appendix A´

Low power encoding techniques for interconnects

The power dissipated in a CMOS Vlsi design can be classified in two categories: static power and dynamic power. The static power is the power consumed when the transistors are not switching while the dynamic power is the power consumed by the switching logic states. As it concerns the dynamic power, the dynamic power consumed by the charging or discharging of the output load external to the cell is referred as ”switching power” while the power dissipated within the cell is referred as ”internal power”.

There are many encoding techniques that have been applied to a bus to reduce the total power dissipation in a CMOS Vlsi. However, many of them apply only to address busses(Asymptotic Zero-Transition Activity Encoding, Working Zone encoding, Grey encoding etc.) [11, 12, 13], so they are not part of our concern. In the following lines we present two of the basic encoding methods to reduce the power dissipation of an interconnect, regardless the kind of transmitted data(memory access, sensor data,command to actuator etc). Despite the fact that they have been implemented only for buses, all of them are general and apply both for crossbar and point-to-point architectures.

A´.1 Invert Coding

One of the most popular and simple methods to reduce the dynamic power of the interconnect is the “Invert Coding” method. We must highlight that this method does not affect the internal power of the circuit but it is designed to reduce the power dissipated cause of the I/O activity. The power dissipated at the I/O can range between a really low percentage to a high enough so as to be the main cause of dynamic power dissipation. It depends on the communication needs of the components of our system.

The target of this method is to reduce the number of transactions in the input/output nodes and slightly increase the number of transactions in the internal nodes[14] . The power penalty for increasing the switching activity in internal nodes is much less significant than what we gain by decreasing the switching activity in external nodes. Now we move on with the description of this method.

The purpose of “invert coding” method is to reduce the number of transitions per cycle but without reducing the amount of transferred data. Here is the procedure to implement the coding[14]: 1) We compute the hamming distance(the number of bits in which they differ) between the present interconnect value (also counting the present invert line) and the next interconnect value. 2) If the Hamming distance is larger than n/2, set invert =1 and make the next interconnect value equal to the inverted next data value. 3) Otherwise let invert = 0 and let the next interconnect value equal to the next data value. 4) At the receiver side the contents of the interconnect must be conditionally inverted according to the invert line, unless the data is not stored encoded as it is (e.g., in a RAM).In any case the value of invert must be transmitted over the bus (the method increases the number of bus lines from n to n + 1).

With the above method the maximum number of transitions per time slot are reduced by half and we have also a smaller reduction at the average number of transitions does not reduce really much. That means that we have a large decrease in the peak power dissipation and a less decrease in the average power dissipation. It turns also out that as the width of the interconnect increase the decrease of the

average power dissipation gets even smaller while the reduction of peak power dissipation remains the same. So if we want to apply this method to interconnects with large-width we should partition them to a number of narrower interconnects. The only disadvantage of this coding is that it requires one extra bus line, the invert line, in order to know at each time if the transmitted data are inverted or not.

A´.2 Shifting Coding

Another simple coding method that we can use to reduce the dynamic power dissipation is the shifting coding method[15]. The concept of this method is, similar with the previous invert coding method, to shift left(or right) the data bits that have to be transferred. Let's consider an example with left shift encoding that will help us understand the privileges of this method. We suppose that the data transmitted at cycle k are 01001101 and the new data arrive at k+1 are 10010010. If we choose to transmit the data as they are then the number of transitions are 6, meaning that 6 bits must toggle. Now, if we had used a right shifting coding method the data at cycle k+1 would be modified to 01001001 and we would have only 1 transition. The methodology of this coding is like the previous one except that instead of inverting we are shifting the data.

Appendix B´

Implantable medical devices

Below we describe current and future IMDs. We give details about the device function, the components and the traffic pattern that simulates its communication activity.

system 1: Artificial pancreas

In one form of diabetes, patients no longer produce enough insulin to lower their blood glucose. Normally, patients have to periodically (prior to eating) inject insulin to keep their blood-glucose levels from spiking. Alternatively, they may rely on "pancreas pumps", which periodically release insulin (even when not needed).

An improvement to treatment can be made using closed-loop control, which only injects insulin when needed. This system models a pancreas stimulator for diabetics, specifically the "Biostator II". The pancreas stimulator (or "smart pump") actively monitors the blood glucose levels. If the glucose level is too high, it emits insulin to lower this glucose.

Components:

- P1: Processor – main implant functionality (SiMS)
- P2: Processor – security processor (SISC).
- S1: Sensor: Glucose
- A1: Actuator: Insulin pump
- M1: Shared memory

Table B´.1: Artificial pancreas traffic patterns

Traffic pattern name	Communication activity	Period
Artificial Pancreas(1)	P1 is reading from glucose sensor,writing to insulin pump and writing to memory the previous values.	1 minute
Artificial Pancreas(2)	case 1 + P2 is setting treatment parameters	1 week
Artificial Pancreas(3)	case 1 + P2 data log	3 times per day
Artificial Pancreas(4)	case 1 + P2 read-log	4 hours

system 2: Artificial pancreas with vital signs

It is not unlikely that any future IMD tracks a patient's four vital signs: Oxygen saturation, heart rate, body temperature and blood pressure. This systems models the "Biostator II", with an additional 4 sensory outputs to monitor these signs.

It has the same traffic pattern as system 1, except that it reads and stores the vital signs too. We refer to the traffic patterns based on the communication activity of this device as ”Artificial Pancreas with vital signs(1)”, ”Artificial Pancreas with vital signs(2)”, ”Artificial Pancreas with vital signs(3)” and ”Artificial Pancreas with vital signs(4)”.

Components:

- P1: Processor – main implant functionality (SiMS)
- P2: Processor – security processor (SISC).
- S1: Sensor: Glucose
- S2: Average Oxygen saturation
- S3: Average heart rate
- S4: Average body temperature
- S5: Average blood pressure
- A1: Actuator: Insulin pump
- M1: Shared memory

system 3/4: Combined Artificial pacemaker & ICD without/with vital signs

An artificial pacemaker actively monitors the heart rate of a patient. If the heart rate is too low, the pacemaker will send electric stimulation to the heart to keep it beating. If the patient is experiencing fibrillation, the ICD-part will send a high voltage electric stimulation to end the fibrillation.

These systems are similar with systems 1 / 2 (without / with vital signs), with the following changes:

- S1 is an ECG sensor, rather than a glucose sensor
- A1 is an electrode (for pacing the heart), rather than an insulin pump.

Table B´.2: Combined Artificial pacemaker & ICD without vital signs

Traffic pattern name	**Communication activity**	**Period**
Combined artificial pacemaker(1)	P1 is reading from sensor ECG, writing to actuator, stores average heart rate+actuator value, keep reading from sensor ECG every 1/300 seconds .	1 second
Combined artificial pacemaker(2)	case 1 + P2 is setting new treatment parameters	1 week
Combined artificial pacemaker(3)	case 1 + P2 data log	3 times per day
Combined artificial pacemaker(4)	case 1 + P2 read-log	4 hours

system 5/6: Epileptic-seizure detection without/with vital signs

Epileptic seizures cause a variety of symptoms, such as involuntary movement, loss of control and absence of mind. Closed-loop seizure control actively monitors the brain activity. If seizure-like activity is detected, an electric (/optogenetic) stimulation is executed, resulting in seizure suppression. These systems are similar with systems 1 / 2 (without / with vital signs), with the following changes:

- S1 and A1 are EEG multi-electrode arrays.

Table B´.3: Epileptic-seizure detection without vital signs traffic pattern

Traffic pattern name	Traffic pattern name	Period
Epileptic-seizure detection(1)	P1 reads from sensor EEG(5 times cause of 5 sensor channels),writes to actuator (channels*electrodes),stores actuator value, keep reading from sensor EEG every 1/100 seconds .	1 second
Epileptic-seizure detection(2)	case 1 + P2 is setting new treatment parameters	1 week
Epileptic-seizure detection(3)	case 1 + P2 data log	3 times per day
Epileptic-seizure detection(4)	case 1 + P2 read log	4 hours

system 7/8: Tinnitus treatment without/with vital signs

Tinnitus is a condition where patients experience phantom sounds. In severe cases, this has substantial affects on the mental health of patients. We are working on a system which trains the brain to ignore these phantom sounds. These systems are similar with systems 1 / 2 (without / with vital signs), with the following changes:

- S1 Is an EEG multi-electrode array.
- A1 is also an EEG multi-electrode array.

Compared to Systems 5/6, A1 requires a substantially higher data throughput: In system 5/6, the output is merely “stimulate” or “don’t stimulate”. In this system, a (complex) waveform is send to the actuator.

Table B´.4: Tinnitus treatment without vital signs traffic pattern

	Communication activity	Period
Tinnitus treatment(1)	P1 reads from sensor EEG(5 times cause of 5 channels),writes to actuator (channels*electrodes),stores actuator value,keep reading from sensor EEG and writing to actuator every 1/100 seconds.	1 second
Tinnitus treatment(2)	case 1 + P2 is setting treatment parameters	1 week
Tinnitus treatment(3)	case 1 + P2 data log	3 times per day
Tinnitus treatment(4)	case 1 + P2 read log	4 hours

The systems 1,3,5 and 7 are using a 2x3 interconnect,the system 6 is using a 2x6 and the rest a 2x7.