İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖZEL DOĞRUDAN BELLEK ERİŞİM MODÜLÜ

YÜKSEK LİSANS TEZİ

Mustafa Mert ESEN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

ŞUBAT 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖZEL DOĞRUDAN BELLEK ERİŞİM MODÜLÜ

YÜKSEK LİSANS TEZİ

Mustafa Mert ESEN
(504191219)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

ŞUBAT 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

CUSTOM DIRECT MEMORY ACCESS MODULE

M.Sc. THESIS

Mustafa Mert ESEN
(504191219)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

FEBRUARY 2023

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 504191219 numaralı Yüksek Lisans Öğrencisi Mustafa Mert ESEN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "ÖZEL DOĞRUDAN BELLEK ERİŞİM MODÜLÜ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Sıddıka Berna ÖRS YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Nizamettin AYDIN** Yıldız Teknik Üniversitesi	.....
	Dr. Öğr. Üye. Ayşe YILMAZER METİN İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **30 Aralık 2022**
Savunma Tarihi : **27 Şubat 2023**

Eşime ve aileme,

ÖNSÖZ

Lisans ve yüksek lisans eğitimim boyunca bana yol gösteren, beni teşvik ve motive eden danışman hocam Prof. Dr. Sıddıka Berna ÖRS YALÇIN'a, bu süreç boyunca gösterdiği desteği ve anlayışı için eşime, her zaman arkamda durup bana inanan aileme, eğitime verdikleri önem ve sağladıkları imkanlar için PROCENNE'e, bu projeyi geliştirmemde katkı sağlayan ekip arkadaşlarıma teşekkür ederim.

Aralık 2022 Mustafa Mert ESEN

İÇİNDEKİLER

KISALTMALAR

AES : Advanced Encryption Standard
AHB : Advanced High Performance Bus Interface
BTÇ : Baytları Ters Çevir
CDMAM : Custom Direct Memory Access Module
CPU : Central Processing Unit
DMA : Direct Memory Access
FIFO : First In First Out
FPGA : Field Programmable Gate Array
FSM : Finite State Machine
HDL : Hardware Description Language
HP : Harici Port
IP : Intellectual Property
ISA : Instruction Set Architecture
İB : İşlem Birimi
LUT : Lookup Table
ÖDBEM : Özel Doğrudan Bellek Erişim Modülü
PCIE : Peripheral Component Interconnect Express
RAM : Random-Access Memory
RISC-V : Reduced Instruction Set Computer-V
ROM : Read Only Memory
RTL : Register-Transfer Level
SoC : System on Chip
SRAM : Static Random-Access Memory
UART : Universal Asynchronous Receiver-Transmitter
VD : Veri Dağıtıcı
VHDL : Very High-Speed Integrated Circuit Hardware Description Language
VS : Veri Seçici

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÖZEL DOĞRUDAN BELLEK ERİŞİM MODÜLÜ

ÖZET

Teknolojinin gelişimiyle birlikte elektronik sistemlerde ve cihazlarda performans beklentisi artmaktadır. Elektronik cihazlar üzerinde yapılan işlemler daha da karmaşıklaşırken kullanılan veri genişlikleri de artmaktadır. Bu sebeple yüksek performans beklentilerinin karşılanabilmesi için donanım kullanımı çokça tercih edilmektedir. Günümüz işlemcilerinin veri genişliğinden çok daha büyük genişliklerde giriş ve çıkış verilerine sahip karmaşık algoritmalar için algoritmaya özel donanımlar (Custom IP) tasarlanarak performansın arttırılması sağlanmaktadır.

Oluşturulan algoritmaya özel tasarımlar işlemlerin bir işlemciye kıyasla çok daha hızlı yapılmasına olanak sağlar. Fakat sadece algoritmaya özel tasarımlar ile giriş verilerinin sağlanması, çıkış verilerinin yönetilmesi, dış dünya ile örneğin Ethernet, UART, PCIE gibi standart arayüzlerle haberleşmenin sağlanması gibi süreçleri yönetmek tasarım sürecini zorlaştıracağı gibi performans kaybına da sebep olabilir. Bu sebeple algoritmaya özel donanımlardan oluşan bir sistem işlemciyle yönetilir. İşlemciye bağlanan Custom IP'ler ile kırmık üstü sistem (System on Chip, SoC) oluşturulur.

Oluşturulan kırmık üstü sistemde veri haberleşmesinin doğrudan işlemci üzerinden gerçekleşmesi algoritmalarda kullanılan verinin büyüklüğü göz önüne alındığında işlemcinin zamanının çoğunun veri haberleşmesi ile geçeceği anlamına gelir. Bu sebeple birden çok birimin belleğe erişebileceği doğrudan bellek erişim modülü (Direct Memory Access, DMA) yapısı kullanılır. DMA yapısı sayesinde işlemci veri haberleşmesinin içinde doğrudan yer almaktansa DMA'i yöneterek veri haberleşmesindeki yükünü hafifletmiş olur.

DMA kullanılarak oluşturulacak bir SoC tasarımında işlemcinin DMA arayüzü olması gerekir. Bu da işlemci tercihi konusunda bir kısıt oluşturur. Örneğin açık kaynak kodlu bir RISC-V çekirdeği işlemci olarak kullanılmak istenirse, DMA arayüzü eklenmesi gerekir. DMA arayüzünün eklenmesi işlemcinin alanını büyüterek çalışma frekansını düşürür ve performans kaybına neden olur.

Bu çalışmada işlemci seçiminde bir kısıt getirmeyen, işlem birimleri ve dış dünya ile haberleşmeyi sağlayacak, ölçeklenebilir ve geliştirilebilir bir Özel Doğrudan Bellek Erişim Modülü(ÖDBEM) tasarımı anlatılmaktadır. ÖDBEM tasarımı işlemci ile doğrudan bir bağlantı gerektirmez. İşlemci ile ÖDBEM arasındaki haberleşme bellek üzerinde belirlenmiş adresler üzerinden yapılır. İşlemci ile doğrudan bir bağlantı gerektirmediği için işlemcinin fazladan bir arayüz içermesi gerekmez. İşlemcinin belleğe erişebiliyor olması yeterlidir. Böylelikle işlemci tercihi konusunda bir kısıt getirmez.

ÖDBEM ölçeklenebilir bir tasarımdır. İşlem birimleri, harici portlar ve bellek arayüzlerine sahiptir. İşlem birimleri ve harici portların sayısı, bellek arayüzünün veri genişliği değiştirilebilir. Bellek çift portlu olmalıdır. ÖDBEM işlemciden aldığı komutlar ile yönetilir. ÖDBEM, işlemci, bellek, işlem birimleri ve harici portlardan oluşan bir kırmık üstü sistemde; harici portlardan okunan veri belleğe yazılabilir, işlem birimlerinde bir işlem başlatılabilir ve sonucu belleğe veya doğrudan bir harici porta yazılabilir, işlem birimlerinin sıfırlama(reset) girişleri kontrol edilebilir, belleğin bir adresinden başka bir adresine kopyalama yapılabilir, bellekteki veri harici portlara yazılabilir.

İşlem birimleri, giriş verisi alan ve çıkış üreten herhangi bir algoritmaya ait donanım gerçeklemesi olabilir. İşlem birimleri arayüzü FIFO'dur. Giriş ve çıkış verilerini yazmak için birer FIFO bulundurur. Tasarlanacak basit bir üst modül ile algoritmaya ait donanımın giriş verileri giriş FIFO'sundan okunarak algoritmaya verilir. Oluşan sonuç ise çıkış FIFO'suna yazılır. İşlem birimlerinin çıkış FIFO'sundaki veri daha önceden işlemci tarafından belirlenmiş bir adrese yazılır. Bunun için işlemcinin tekrar bir komut göndermesine gerek olmaz.

Harici portlar kırmık üstü sistemin dış dünya ile bağlantısını sağlayacak Ethernet, UART, PCIE gibi fziksel portlara karşılık gelir. Harici portların arayüzü FIFO'dur. Bir giriş ve bir çıkış FIFO'su içerir. Tasarlanan bir üst modül ile harici portun giriş FIFO'sundan okunan veri fiziksel portlara yazılır. Benzer şekilde fiziksel portlardan okunan veri harici portlar çıkış FIFO'suna yazılır.

Çalışma kapsamında ÖDBEM'in tasarım detayları verilmiş, donanım tanımlama dilleri ile ölçeklenebilir ve geliştirilebilir bir tasarım yapılmış, yapılan tasarım davranışsal benzetim yoluyla doğrulanmış, sentez ve gerçekleme yapılarak kaynak tüketim verileri paylaşılmış, sentez sonrası benzetim yapılmış ve doğru çalıştığı gözlemlenmiştir.

CUSTOM DIRECT MEMORY ACCESS MODULE

SUMMARY

As technology advances, performance expectations for electronic systems and devices have increased. While the operations conducted on electronic devices become more complex, the data widths used have also increased. Therefore, hardware usage is preferred in order to meet high performance expectations. To improve performance for complex algorithms with inputs and outputs of data widths much larger than current processors, custom IPs are designed specifically for the algorithm.

Algorithm-specific designs enable processes to be carried out much faster than on a processor. However, managing processes such as providing input data, managing output data and communicating with the outside world through standard interfaces such as Ethernet, UART and PCIE, only using algorithm-specific designs can complicate the design process and cause performance loss. Therefore, a system consisting of algorithm-specific hardware is managed by a processor. Custom IPs connected to the processor are used to create a system-on-chip (SoC).

If data communication in the system on chip (SoC) design is via the processor, most of the processor time is spent in data communication due to the use of high data widths in algorithms. Therefore, a direct memory access (DMA) structure is used, where multiple units can access memory. With the help of the DMA structure, the processor does not take part directly in the data communication, but rather manages the DMA to reduce the amount of time spent on data communication.

In a SoC design created using DMA, the processor must have a DMA interface. This creates a constraint on the processor choice. For example, if an open-source RISC-V processor is to be used, the DMA interface must be added. Adding the DMA interface increases the area of the processor, reducing the operating frequency and resulting in a performance loss.

In this study, a Custom Direct Memory Access Module (CDMAM) design is presented that does not impose a constraint on processor selection, provides communication with processing units and external world, is scalable and can be developed. The CDMAM design does not require a direct connection with the processor. The communication between the processor and the CDMAM is done through predetermined addresses on the memory. Since it does not require a direct connection with the processor, the processor does not need to have an additional interface. It is enaugh the processor to be able to access the memory. Thus, it does not impose any constraint on processor selection.

CDMAM is a scalable design. It has processing elements, external ports and memory interfaces. The number of processing elements and external ports can be changed. The width of the data of the memory interface can be changed.

The memory should have two ports. One port should be connected to the processor, and the other port should be connected to the CDMAM module. The data widths of the processor and the CDMAM module can be different. In such cases, the memory should be designed to support this difference.

In the presented work, the design is described where the data width of the CDMAM module is twice that of the processor. However, with small modifications to the design, solutions can be achieved for different data width combinations as well. There doesn't need to be a direct interface between the processor and the CDMAM module.

Communication between the processor and the CDMAM module can be done through the memory. Specific addresses are used on the memory for this purpose. These addresses enable the transmission of commands from the processor to the CDMAM module and allow the CDMAM module to transmit its status signals to the processor. The address values are designed to be configurable during the design. When writing an application on the processor, these addresses should be determined to match those of the CDMAM module. A command written by the processor to the predetermined command address on the memory is read and processed by the CDMAM module. Once the command is processed and completed, the CDMAM module writes the information about the completion to the command addresses on the memory, which is then read by the processor.

In a system composed of a processor, memory, processing elements and external ports, CDMAM can write data read from external ports to memory, start a process in a processing element and write the result to memory or directly to an external port, control the reset inputs of processing elements, copy from one address to another in memory, write data in memory to external ports.

Processing elements can be hardware implementations of any algorithm that takes input data and produces output. The interface of the processing element is FIFO. It has one input and one output FIFO. The input data is provided to the algorithm by a simple top module that reads the input data from the input FIFO. The resulting output is written to the output FIFO. The data in the output FIFO of the processing element is written to a predetermined address by the processor without the need to send another command from the processor.

External ports correspond to physical ports such as Ethernet, UART, PCIE which provide the system's connection to the outside world. The interface of the external ports is FIFO. It contains one input and one output FIFO. Data read from the input FIFO of the external port is written to the physical ports by a top module designed. Similarly, data read from the physical ports is written to the output FIFO of the external ports.

The CDMAM design has a simple and effective command structure. It can perform all the operations described in the 4 basic commands. These commands are send, copy, get, and reset.

CDMAM commands are written by the processor to the address specified as the command address in the memory. This address is continuously read by CDMAM, and when a command is written by the processor, it is read and processed. CDMAM commands consist of four components: operation code, source, destination, and length.

The operation code includes validity, operation type, and parameters. Validity determines whether the code written by the processor is valid or not. This allows CDMAM to determine if there is a command that needs to be processed. The operation type determines which of the operations to perform: send, copy, receive, or reset.

The source determines the source of the data transfer, which can be a memory address or a connection number. The destination determines the destination of the data transfer, which can also be a memory address or a connection number. The length represents the size of the data transfer in bytes.

The send command allows sending data to a processing unit or an external port. In this command, the source specifies the memory address, and the destination specifies the connection number. Sending data to processing units implies performing an operation within a processing unit. If this operation results in an output, information about the size of the output and the memory address to write it to or the external port to send it to should be sent to the processing unit initially.Thus, there is no need to send a new command to CDMAM to read the output generated in the processing units. When a processing unit produces an output, CDMAM reads it and writes it to the necessary location.

The copy command allows copying data from one memory address to another. In this command, the source and destination represent the memory addresses. With the help of CDMAM, a data specified by the processor with its start address and size can be quickly copied to the specified destination address.

The get command enables reading data from external ports and writing it to memory. In this command, the source represents the connection number of the external port from which the data will be read, and the destination represents the memory address where the data will be written. The length parameter is not used in this command. The size of the data is determined by reading the first word of the data from the external port.

The reset command allows managing the reset inputs of processing units. With this command, one or more processing units can be reset. In this command, the terms source, destination, and size are not used. Instead, it utilizes another designated address in memory called the reset address. Each bit of the data written to this address represents the reset input for a specific processing unit. The processor sets the bit to one for the processing unit or units it wants to reset and sends a reset request to CDMAM. To complete the reset process, the relevant value at the reset address is set to zero, and another reset request is sent.

In the scope of this work, the design details of CDMAM have been given, a scalable and developable design has been made with hardware description languages, the design has been verified by behavioral simulation, resource consumption has been shared by synthesis and implementation, post synthesis simulations have been made and it has been observed that it works correctly.

1. GİRİŞ

Doğrudan bellek erişim modülü (Direct Memory Access, DMA) işlemci dışında diğer bileşenlerin de doğrudan belleğe erişmesini sağlayan bir mekanizmadır [1]. İşlemcinin sistem içerisinde bulunan işlem birimlerinin veri transferlerinde ve fiziksel portlar aracılığı ile dış dünyadan gelen verilerin transferinde doğrudan rol almasının önüne geçerek işlemcinin iş yükünü hafifletmeyi amaçlar.

Günümüzde yapılan hemen her tasarım için hız beklentilerinin artması ve yapılan işlemlerde büyük veri kullanımının yaygınlaşması yüksek performanslara erişmek için donanım kullanımını daha önemli hale getirmektedir [2]. Bu sebeple özellikle kriptografi gibi bir işlemci veri genişliğinin 4 ila 128 katı genişliğinde verilerle işlemlerin yapıldığı sistemler için algoritmaya özel donanımlar (Custom IP) tasarlanarak, standart bir işlemci yapısına göre çok daha yüksek performans hedefleri karşılanabilmektedir.

Algoritmaya özel tasarlanan ve FPGA üzerinde gerçeklenen Custom IP'ler her ne kadar işlemlerin işlemciye göre çok daha hızlı gerçeklenmesini sağlasa da bu işlemlerin başlatılması, sonuçların okunması, dış dünyadan Ethernet, UART, PCI gibi standard arayüzler ile veri transfer işlemleri performans hedefinin önünde bir engel oluşturmaktadır. Bu sebeple kontrol ve dış dünya ile veri haberleşmesi işlerinden sorumlu bir işlemci kullanılmakta ve algoritmaya özel donanımlar bu işlemciye bağlanarak bir kırmık üstü sistem (system on chip – SoC) oluşturulmaktadır.

Custom IP'ler ile gerçeklenen algoritmaların giriş ve çıkış veri genişlikleri günümüzde var olan işlemcilerin veri yolu uzunluklarından kat ve kat fazla olduğu için, her ne kadar algoritmaya özel donanımlar tasarlanarak işlemlerin hızlı yapılması sağlanıyor olsa da işlemciye doğrudan bağlanan Custom IP'lerin veri giriş çıkışının kontrolü işlemciye ağır işlem yükü getirerek veri transferi bir darboğaz oluşturmakta ve sistemin verimli çalışmasının önünde bir engel olmaktadır.

İşlemcinin veri transferi yükünün hafifletilmesi için DMA içeren bir tasarım yapılabilir. Ancak, işlemci olarak bir açık kaynak kodlu (soft) işlemci kullanıldığında, bu işlemcinin bir DMA arayüzü olmasını zorunlu hale getirmekte, bu zorunluluk ise performans kaybına sebep olduğu gibi DMA arayüzüne sahip olmayan işlemcilerin tercihi konusunda bir kısıt oluşturmaktadır. Örneğin, işlemci olarak bir açık kaynak kodlu bir RISC-V çekirdeği kullanılmak istenirse DMA arayüzü eklemek zorunluluğu işlemci çekirdeğinin alanının büyümesine ve çalışabileceği maksimum saat frekansının düşmesine sebep olmaktadır.

Dolayısıyla tüm bu olası sistemlerin eksikliklerini tamamlayacak, işlemci tercihinde bir kısıt oluşturmayacak, yüksek performans hedeflerine ulaşılmasına olanak sağlayan, kolay uygulanabilir ve ölçeklenebilir bir özel doğrudan bellek erişimi (Custom DMA) birimi tasarımına ihtiyaç duyulmaktadır.

2. ÖNBİLGİLER

2.1 Doğrudan Bellek Erişimi

Doğrudan bellek Erişimi (Direct Memory Access, DMA), bir sistemde bellek ve diğer bileşenler arasında veri iletimini sağlayan bir yapıdır [3]. Birden fazla bileşenin belleğe erişiminin gerektiği sistemlerde bellek erişimini yönetir. Böylelikle işlemcinin, diğer bileşenlerin bellek ile yapacağı veri alışverişi için sarfettiği vakitten tasarruf sağlanır.

İşlemci dışındaki diğer bileşenler, işlem birimleri ve harici portlar olarak sınıflandırılabilir. İşlem birimleri sistem içerisinde bulunan, bellek ile veri alışverişi yapan, bir giriş verisi alıp bir çıkış verisi üreten herhangi bir yapı olabilir. Harici portlar ise sistemin dış dünya ile haberleşmesini sağlayan fiziksel portlardır. Fiziksel portlar DMA aracılığı ile belleğe erişerek veri yazabilir veya okuyabilir. DMA yapısı sayesinde işlemci bu veri alışverişlerinde her ne kadar işlemi başlatan olsa da veri transferini kendisi yapmaz. Böylelikle özellikle yüksek performans hedeflenen sistemlerde işlemci büyük bir işlem yükünden kurtulmuş olur.

2.2 RISC-V

RISC-V açık kaynak kodlu bir komut seti mimarisidir (Instruction Set Architecture, ISA) [4]. Tasarım özgürlüğünü ve esnekliği temel ilke edindiğinden geniş kitleler tarafından destek bulmuştur. RISC-V ISA, bir uygulamanın gerçeklenebilmesi için en basit komut setini sunmakla birlikte daha karmaşık uygulamalar için gerekli olabilecek ek komut setlerini de tanımlar. Bir gerçeklemenin ek komut setlerinden hangsini içereceği ve nasıl uygulanacağı konusunda tasarımcı özgürdür. Böylelikle her tasarımcı kendi ihtiyaçları doğrultusunda ve tasarım yöntemleriyle ihtiyacına en uygun işlemci gerçeklemesini yapabilir. Bununla birlikte RISCV organizasyonu tarafından sağlanan gerçeklemeleri kullanabilir ve bunlar üzerinde ihtiyacı olan geliştirmeleri

yapabilir. RISCV ISA işlemciler, ulaşılabilirliği ve esnekliği sayesinde birçok sistemde kullanılmakta ve kullanımı her geçen gün artmaktadır.

2.3 Bellek

Bellek bir sistemde verinin saklanması ve ihtiyaç duyulan kısmına erişilmesine olanak sağlar [5]. Bir sayısal tasarımda verinin saklanması için bulunan en temel eleman yazmaçlardır. Yazmaçlarda veri ikilik sistemde tutulur. Yazma kontrol işareti geldiğinde girişinde bulunan veri kaydedilir. Kaydedilen veri aynı zamanda çıkışıdır. Sistemlerde kullanılan verinin büyümesiyle yazmaç kullanımı performans gereksinimlerini karşılayamaz. Bu sebeple adreslenmiş yazmaçlardan oluşan bellek yapısı kullanılır. Bellek, sahip olduğu adres girişi ile verinin belirli bir parçasına erişilerek okunmasını veya değiştirilmesini sağlar. Birden çok verinin aynı bellek içerisinde tutulabilmesini ve kolay bir şekilde yönetilmesini sağlar.

Bellek yapısı temelde 3 giriş ve 1 çıkıştan oluşur. Girişler veri girişi, adres girişi ve yaz / oku seçim girişidir. Çıkışı ise veri çıkışıdır. Bu üç giriş ve bir çıkış bir bellek portu olarak adlandırılır. İki portlu bellek iki bellek portu içerir. Böylelikle iki farklı bileşen tarafından bellek içerisindeki veri değiştirilebilir veya okunabilir.

2.4 İlk Giren İlk Çıkar

İlk giren ilk çıkar (First In First Out, FIFO) bir veri yönetimi yöntemidir [6]. Bu yöntemde bir belleğe yazılan ardışık verilerden ilk yazılan ilk olarak okunacaktır. Uygulamaya bağlı olarak farklı tasarımlar yapılabilir. Bir FIFO'nun sayısal tasarımında tasarım yöntemine göre farklılıklar olsa da en az içermesi gereken üç giriş ve üç çıkış bulunur. Girişler veri girişi, yaz kontrol işareti, oku kontrol işaretidir. Çıkışlar ise veri çıkışı, dolu durum işareti ve boş durum işaretidir. Tasarıma bağlı olarak veri girişi genişliği ve veri çıkışı genişliği farklı olabilir. Bu giriş ve çıkışlar kullanılarak veri, yaz kontrol işareti ile birlikte gönderilir. Eğer FIFO dolu değil ise veri FIFO'ya yazılır. FIFO'nun kapasitesi kadar veri kelimesi FIFO'ya yazılabilir. Yazılan veri, ilk yazılandan başlanarak FIFO'nun veri çıkışından okunur. Veri çıkışındaki verinin okunmasıyla sonraki verinin veri çıkışına yazılması için oku kontrol işareti

gönderilir. Boş durum işaretinin değerinin ’1’ olması FIFO içerisinde okunabilecek bir verinin olmadığı anlamına gelir.

3. LİTERATÜR

Bu başlık altında doğrudan bellek erişimi ile ilgili makaleler ve patentlere yer verilmiştir. Mevcut doğrudan bellek erişimi tasarımları ile tezin konusu olan ÖDBEM tasarımının farklılıkları verilmiştir.

[7] numaralı makalede düşük güç tüketimini hedefleyen bir DMA tasarımı verilmiş. Belleğin ve RISC-V işlemcisinin gelişmiş yüksek performanslı veriyolu arayüzüne (Advanced High Performance Bus Interface, AHB) sahip olması gerektiği belirtilmiş. Ayrıca verilen tasarımda işlem birimleri ile haberleşmeye yer verilmemiş. USB ve QSPI gibi fiziksel portlarla haberleşmenin düşük güç tüketimiyle yapılması hedeflenmiştir. Bir N-bit veri uzunluğu ve M kanalları için FIFO kelime uzunluğunu tanımlar. Ortak mimariler olmasına rağmen, FIFO her işlem için adresleri saklamaz. FIFO, yapı parametre bayraklarına bağlı olarak verileri çift kanallı bir RAM veya yazmaçlar kullanarak depolar. Her çevre birimi, bellek erişimini gerçekleştirmek için tek veya birden çok kanal kullanarak okuma ve yazma işlemleri yapabilir. FIFO, onaylanacak kanalı, işlem türünü ve yazma işlemleri için yazma verilerini tanımlamak için bir tanımlayıcı numara saklar. Bir arbiter (yeşil), işlemleri eşit önceliklere sahip tüm kanallar arasında çoklar. Kanal arayüzü, alandan tasarruf etmek için birkaç sinyal portunu çalıştırır.

[8] numaralı makalede DMA veri yolu, kaynak adres jeneratörünü kullanarak kaynak bağlantı noktasından veri alır ve hedef adres jeneratörü tarafından oluşturulan adresi kullanarak verileri hedef bağlantı noktasına depolar. Farklı veri hızlarına ve formatlarına sahip verileri işlemek için kaynak kod çözme modülüne ihtiyaç vardır. DMA kontrol yolu, Sonlu Durum Makinesinden oluşur. DSP çekirdeği bir kanalın konfigürasyonunu talep edebilir. FSM, veri yolunun kontrolünden sorumludur. Aktarım tamamlandığında FSM, DSP çekirdeğini kapatacaktır.

[9] numaralı makalede bir DMA tasarımı verilmiş. Verilen tasarımda DMA modülünün ve işlemcinin AHB arayüzüne sahip olması gerektiği belirtilmiştir.

Önerilen DMA yapsı, dört iletim kanalı bulundurur. Eşzamanlı donanım istekleri için, DMA önceden ayarlanmış önceliğe göre karar verecek ve her seferinde yalnızca bir isteğe yanıt verecektir. İki çalışma modunu döngü modunu ve tekli modu destekler. Bayt, yarım kelime ve tam kelime aktarımı yapılabilir. Güç kontrolü amacıyla bir bütün olarak devre dışı bırakılabilir.

[10] numaralı patentte işlemcinin yüksek verimli veriyolu arayüzü olduğu varsayımı yapılmıştır. Patentte bahsi geçen DMA'in 2 arayüzü olduğu söylenmiş. Biri işlemci diğeri çevresel. Bu tasarım kendi içerisinde çevresellerin komutlarını tutan bir bellek içerir. İşlemcinin komut belleğinde yer alan program parçası bu belleğe kopyalanır. Buna karşılık bizim tasarımımızda işlemci sadece basit bellek erişim arayüzüne sahiptir. Ek bir veri yolu arayüzüne ihtiyacı yoktur. Bütün çevresellerin kontrol işaretleri ÖDBEM üzerinden sadece basit bellek erişim arayüzü ile yapılır. ÖDBEM tasarımında DMA'in bellek, çevresel ve harici portlar olmak üzere 3 tip arayüzü vardır. ÖDBEM çevresel veri yolunu (peripherial bus) da içermektedir. Ayrıca harici portlar için de bir veri yolu içerir. İçerisinde herhangi bir bellek yapısı bulunmaz. İşlemci sadece bellekte ÖDBEM için ayrılmış gözlere ÖDBEM komutunu yazar. Bu komut ÖDBEM içerisinde yer alan komut yazmacına alınarak kod çözme işlemi yapılır. Bu komut ile sadece verinin kaynağı ve hedefine karar verilir. Çevresellere gönderilecek komut veya veri ÖDBEM davranışı üzerinde bir farklılık oluşturmaz. ÖDBEM yalnızca kaynak ve hedef arasında veri yolunu seçme görevi yapar.

[11] numaralı patentte bir doğrudan bellek erişim veri kanalı oluşturan ve bir belleğe bağlantı için bir birinci arabirim içeren bir doğrudan bellek erişim denetleyicisinden oluşan bir doğrudan bellek erişim sistemini açıklamaktadır. Buluşta çok sayıda düğüme bağlanmak için ikinci bir arayüz bulunmaktadır ve bir işlemci, doğrudan bellek erişim denetleyicisine ve ikinci arabirime birleştirilmiştir. IPCM (Virtual DMA) içinde RISC işlemcisi mevcuttur. RISC işlemcisinin çalışabilmesi için komut belleği (ROM) ve veri belleği (SRAM) bulunmaktadır. Bu sebeple bu tasarım hem yazılım hem donanım parçaları içermektedir. Bu patentte host işlemcinin host bus üzerinden IPCM'e ve belleğe erişebildiği görülmektedir. Buna karşılık ÖDBEM bir işlemci içermez. Sadece donanım ile gerçeklenmiştir. Kontrol bir sonlu durum makinesi(Finite

State Machine, FSM) ile sağlanır. Host işlemci sadece basit bellek erişim arayüzüne sahiptir. Ek bir veri yolu arayüzüne ihtiyacı yoktur.

[12] numaralı patentte doğrudan bellek erişimi için tasarlanmış bir yöntem ve sistem açıklanmaktadır. Bu patentte DMA devresinin kontrolü üzerine bir metot ve sistem önerisi yapılmıştır. DMA devresi hazır olarak kullanılmıştır. ÖDBEM tasarımı bir DMA devresidir.

[13] numaralı patentte harici olarak tetiklenebilen ve ayrıca programlanabilir bir sırayla farklı sıralı olmayan bellek konumlarına atlayabilen programlanabilir bir rastgele dizili doğrudan bellek erişimi (DMA) denetleyicisi açıklanmaktadır. Bu patentte konfigüre edilmeye çalışılan çevreseller işlemciden bağımsız çalışmakta ve DMA sadece konfigürasyon bilgisini göndermek için kullanılmaktadır. Buna karşılık, ÖDBEM tasarımında çevreseller işlemci tarafından kontrol edilmelidirler. Ayrıca ÖDBEM tasarımı hem çevresellerin konfigürasyonları hem de bellekten çevresellere veri aktarmak için kullanılan veri yolunun konfigürasyonunu sağlar.

[14] numaralı patentte daha esnek ve verimli transfer yeteneklerine sahip bir DMA kontrolörü sağlanması için tanımlayıcıların sıralı ve sıralı olmayan şekilde işlenmesi arasında verimli geçişe izin verecek şekilde iki tip tanımlayıcının kullanıldığı, Scatter-Gather doğrudan bellek erişim kontrolcüsü açıklanmaktadır. [15] numaralı patentte de aynı tasarım üzerinde uygulanan farklı yöntemleri içermektedir. Bu iki patentte CPU'nun veri yoluna erişimi olduğu görülmektedir ve DMA kontrolü için CPU ile DMA Controller arasında özel bağlantı kullanılmaktadır. Buna karşılık ÖDBEM CPU'nun sadece bellek erişiminin olması sistemin çalışması için yeterlidir. DMA kontrolü tasarım sırasında belirlenen bellek gözlerine işlemci tarafından yazılıp ÖDBEM tarafından komutlar ile sağlanır. Özel bir bağlantıya ihtiyaç duyulmamaktadır.

[16] numaralı patentte aynı öncelik seviyesine sahip DMA kanallarının çoklu DMA kanalı işlemleri arasında geçiş yapmak üzere çalıştırılabilir özellikte bir DMA kontrolörü açıklanmaktadır. Bu patentte CPU'nun bir veriyolu arayüzü (bus matrix) olduğu görülmektedir. Yine DMA kontrolü için de CPU ile DMA Controller arasında Bus Matrix kullanılmaktadır. Buna karşılık ÖDBEM tasarımında CPU'nun sadece

bellek erişiminin olması patent konusu olan sistemin çalışması için yeterlidir. DMA kontrolü tasarım sırasında belirlenen bellek gözlerine CPU tarafından yazılıp ÖDBEM tarafından komutlar ile sağlanır.

ÖDBEM ile diğer tasarımların işlemcinin bir ek arayüz gerektirmemesi, işlem birimleri ile veri alışverişi yapabilmesi, harici portlar ile veri alışverişi yapabilmesi, işlem birimlerine komut iletimi yapabilmesi ve harici bir veri yolu gereksinimi olmaması özellikleri açısından karşılaştırılması Çizelge 3.1'de verilmiştir.

Çizelge 3.1 : Tasarımların karşılaştırılması.

Tasarım	İşlemci Arayüzü	İşlem B.	Harici P.	Komut İletimi	Veri Yolu
[7]	-	-	+	-	-
[8]	-	+	+	+	-
[9]	-	+	+	+	-
[10]	-	-	+	+	-
[11]	-	-	+	-	-
[12]	-	+	-	-	-
[13]	-	+	-	+	-
[14]	-	+	+	+	-
[15]	-	+	+	+	-
[16]	-	+	+	+	-
ÖDBEM	+	+	+	+	+

Literatür ve patent araştırmalarının ardından bugüne kadar yapılan DMA tasarımlarında genel olarak DMA bellek ile haberleşme arayüz kullanılarak sağlanırken üzerinde çalışılan tasarımda ek bir arayüz kullanılmayacaktır. İşlemcinin sadece bellek arayüzü kullanılarak komut ve çekirdeklerin veri iletimleri gerçeklenecektir. Benzer konularda birçok çalışma yapılmış olmasına rağmen yapılacak olan çalışmayla birebir bağlantılı bir çalışmaya rastlanmamıştır.

4. TASARIM

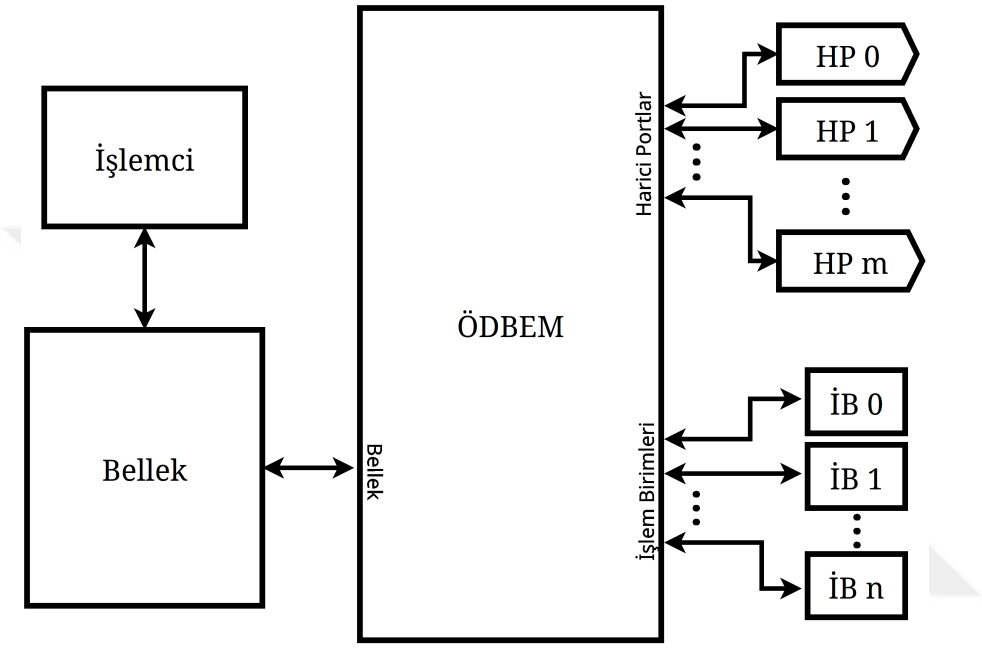
ÖDBEM, işlemci, harici portlar ve işlem birimleri arasında veri ve komut haberleşmesi için kullanılacak olan bir doğrudan bellek erişim modülüdür. Doğrudan bellek erişim yapısı gereği veri transferi işlemci üzerinden yapılmaz ve böylelikle işlemci veri transferi için vakit harcamaz. ÖDBEM harici bir veriyoluna ihtiyaç duymaz, gerekli veri yolunu kendi içerisinde bulundurur. İşlemci ile doğrudan bir bağlantı gerektirmez. Bu sebeple işlemcinin ekstra bir arayüz gerekliliğini ortadan kaldırarak işlemci seçiminde esneklik sağlar. ÖDBEM ile işlemci arasındaki haberleşme bellek içerisindeki belirli adresler üzerinden gerçekleşir. Performans kaybı olmaması adına işlemci ve ÖDBEM'in belleğe eşzamanlı erişimi olması gerekir. Bu sebeple ÖDBEM kullanılarak oluşturulacak olan bir sistemde çift portlu bir belleğe ihtiyaç duyulur. Bu bellek aynı zamanda işlemcinin veri belleğidir. İşlemcinin yapısına göre, veri belleği olmasıyla birlikte komut belleği olarak da kullanılabilir.

İşlem birimleri, bir işleme özel olarak tasarlanmış giriş verisini işleyerek bir çıkış verisi oluşturabilen herhangi bir tasarıma karşılık gelir. Giriş verisi komut, boyut veya işlem birimine özel parametreleri içerebilir.İşlem birimleri belleğe ÖDBEM üzerinden erişir. Bellekten okunan veri işlem birimlerine ÖDBEM ile gönderilir. İşlem birimlerinde üretilen çıkış verisi yine ÖDBEM tarafından okunarak belleğe yazılır. İşlem birimleri sayısı değiştirilebilir. Harici portlar sistemin, sistem dışına açılan kapısıdır. UART, PCIE gibi fiziksel portlardan veri alınmasını ve gönderilmesini sağlar. Harici portların sayısı değiştirilebilir.

ÖDBEM kullanılarak oluşturulabilecek bir kırmık üstü sistemin tasarımı Şekil 4.1 de verilmiştir. Şekil 4.1'e benzer bir sistem kurulduğunda sistemin haberleşmesi ÖDBEM ile sağlanabilir. Bu bağlamda,

- Dış dünyaya bağlı harici portlardan veri okunarak belleğe yazılabilir.
- Bellekteki veri harici portlara gönderilebilir.

- İşlem birimlerinde işlem başlatılabilir ve sonucu belleğe veya doğrudan harici porta yazılabilir.
- İşlem birimlerimlerinin sıfırlama girişi kontrol edilebilir.
- Belleğin bir adresinden başka bir adresine kopyalama yapılabilir.

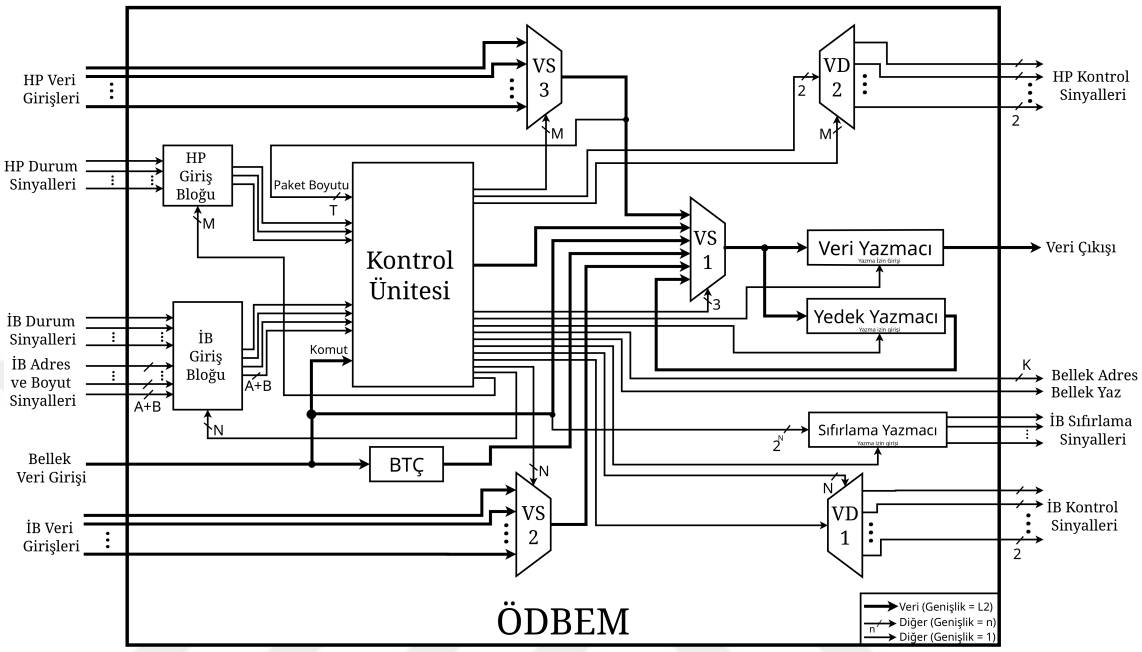


Şekil 4.1 : ÖDBEM kullanılarak oluşturulabilecek kırmık üstü sistem tasarımı.

4.1 Özel Doğrudan Bellek Erişim Modülü (ÖDBEM)

ÖDBEM bir doğrudan bellek erişim modülüdür ve işlemciden aldığı komutlar ile Şekil 4.1 de verilen sistemin haberleşmesini sağlar. Bellek, Harici Portlar ve İşlem Birimleri arayüzleri bulunur. Bellek arayüzü basit bellek arayüzüdür. Veri girişi, veri çıkışı, yazma emir çıkışı ve adres çıkışından oluşur. İşlem birimleri ve harici portların sayısı değişebilir. Harici portlar arayüzü FIFO arayüzüdür. Veri girişi, veri çıkışı, dolu ve geçerli durum işaretleri, yaz ve oku kontrol işaretlerinden oluşur. Birden fazla harici port olması durumunda her bir harici port için bu arayüzden bir adet bulunur. İşlem birimleri arayüzü FIFO arayüzüne ek olarak sıfırla kontrol işareti, adres ve boyut işaretleri de bulunur. Birden fazla işlem birimi olması durumunda bu arayüzden her bir işlem birimi için bir adet bulunur.

ÖDBEM işlemciden aldığı komutlar ile haberleşmeye konu olan kaynak ve hedef arasında veri yolunu konfigüre ederek veri transferini gerçekleştirir. ÖDBEM tasarımı Şekil 4.2'de verilmiştir.



Şekil 4.2 : ÖDBEM tasarımı.

ÖDBEM tasarımı kontrol ünitesi, işlem birimleri giriş bloğu, harici portlar giriş bloğu, baytları ters çevir bloğu, yazmaçlar, veri seçiciler ve veri dağıtıcılardan oluşur. Kontrol ünitesi tarafından kontrol edilir.

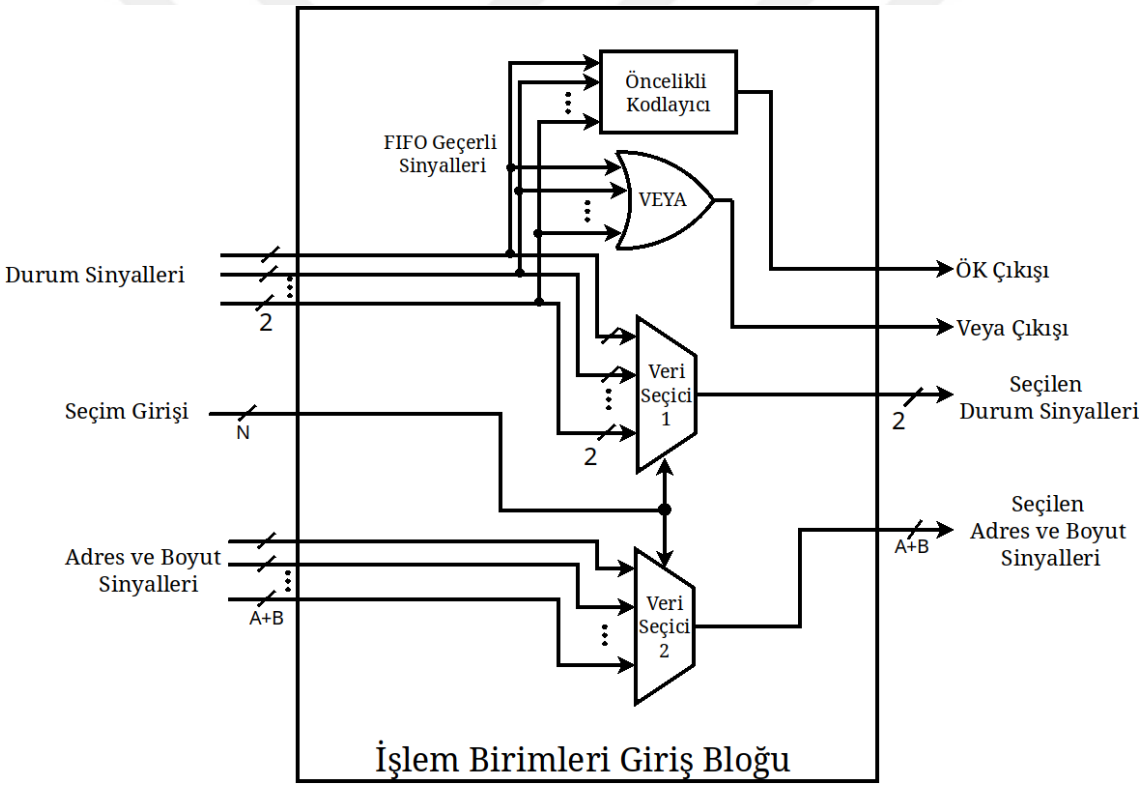
Çizelge 4.1 : ÖDBEM Tasarım Parametreleri

Parametre	Açıklama
L1	Belleğin işlemciye bağlı portunun veri genişliği
L2	Belleğin ÖDBEM'e bağlı portunun veri genişiliği
K	Belleğin adres işaretinin veri genişliği
n	İşlem birimleri sayısı
N	İşlem birimleri sayısının ifade edilebileceği en küçük bit sayısı
m	Harici port sayısı
M	Harici port sayısının ifade edilebileceği en küçük bit sayısı
A	İşlem birimleri adres işaretinin veri genişliği
B	İşlem birimleri boyut işaretinin veri genişliği
T	Harici portlar paket boyutunun veri genişliği

ÖDBEM tasarımı ölçeklenebilir bir yapıdadır. İşlem birimi sayısı, harici port sayısı ve veri genişlikleri gibi birçok parametre değiştirilebilir. Bu yüzden tasarım üzerinde bazı değerler parametrik olarak verilmiştir. Çizelge 4.1de tasarım parametreleri ve açıklamaları verilmiştir.

4.1.1 İşlem birimleri giriş bloğu

İşlem birimleri giriş bloğu işlem birimlerinin durum işaretleri ve adres ve boyut işaretlerinin kontrol ünitesi tarafından değerlendirilmesinde ve seçilmesinde kullanılır. Öncelikli kodlayıcı, veya kapısı ve iki adet veri seçiciden oluşur. İşlem birimleri giriş bloğunun iç yapısı Şekil 4.3de verilmiştir.



Şekil 4.3 : İşlem Birimleri Giriş Bloğu.

İşlem birimleri giriş bloğunun girişleri; işlem birimlerinin durum işaretleri, adres ve boyut işaretleridir. Bir işlem biriminin durum işaretleri, bir bitlik dolu durum işareti ve bir bitlik geçerli durum işareti olmak üzere 2 bitten oluşur. Bu iki bitten sistemde bulunan işlem birimi sayısınca(n) bulunur.

Dolu durum işareti işlem birimleri giriş FIFO'sunun dolu olup olmadığını gösterir. Değeri '1' olduğunda dolu olduğu anlamına gelir. Geçerli durum işareti, işlem biriminin çıkış FIFO'sunun veri çıkışındaki verinin geçerli olup olmadığını gösterir. Değeri '1' olduğunda veri çıkışının geçerli olduğu anlamına gelir. Adres ve boyut işaretleri işlem biriminde üretilecek çıkışın yazılacağı bellek adresini ve boyutunu ifade eder. Bu değerler, işlemci tarafından belirlenerek ÖDBEM vasıtası ile işlem birimlerine gönderilir.

Öncelikli kodlayıcı aktif giriş işaretlerinden önceliği en yüksek olan girişin, giriş sırasını çıkış olarak belirler. Girişleri işlem birimlerinin geçerli durum işaretleridir. Öncelik değerce küçük olan bağlantı numarasındadır. Örneğin birinci ve ikinci işlem birimlerinin geçerli durum işareti '1' olduğunda öncelikli kodlayıcının çıkış değeri '01' olur. İkinci ve üçüncü işlem birimlerinin geçerli durum işareti '1' olduğunda öncelikli kodlayıcı çıkışı '10' olur. Öncelikli kodlayıcının girişi n bit çıkışı N bittir.

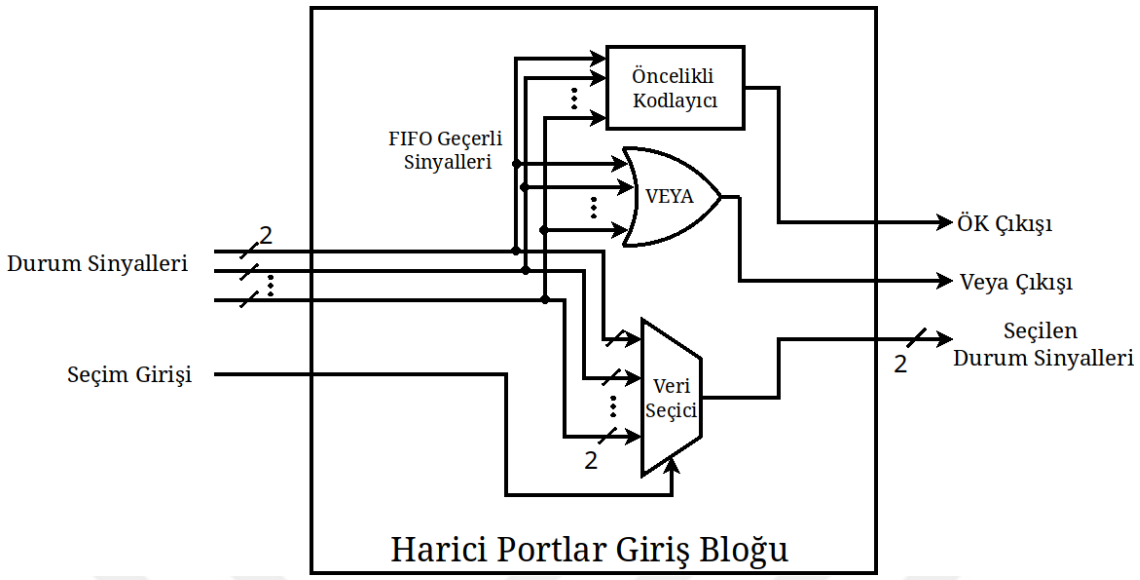
Veya kapısının girişleri işlem birimlerinin geçerli işaretleridir. Girişi n bit ve çıkışı '1' bittir. İşlem birimlerinin geçerli işaretlerinden herhangi biri '1' olduğunda çıkışı '1' olur. Girişlerinin hepsi '0' olduğunda çıkışı '0' olur.

Veri seçici 1, üzerinde işlem yapılacak olan işlem biriminin durum işaretlerinin seçilmesini sağlar. Veri girişleri işlem birimlerinin durum işaretleridir. İki bitlik n adet veri girişi vardır. Seçim girişi N bitliktir ve kontrol ünitesi tarafından gönderilir. Çıkışı seçilen işlem biriminin durum işaretleridir ve iki bittir.

Veri seçici 2, üzerinde işlem yapılacak olan işlem biriminin adres ve boyut işaretlerinin seçilmesini sağlar. Veri girişleri işlem birimlerinin adres ve boyut işaretleridir. A + B bitlik n adet veri girişi vardır. Çıkışı seçilen işlem biriminin adres ve boyut işaretleridir ve A + B bittir.

4.1.2 Harici portlar giriş bloğu

Harici portlar giriş bloğu harici portların durum işaretlerinin kontrol ünitesi tarafından değerlendirilmesinde ve seçilmesinde kullanılır. Öncelikli kodlayıcı, veya kapısı ve veri seçiciden oluşur. Harici portlar giriş bloğunun iç yapısı Şekil 4.4de verilmiştir.



Şekil 4.4 : Harici Portlar Giriş Bloğu.

Harici portlar giriş bloğunun girişleri; harici portların durum işaretleridir. Bir harici portun durum işaretleri, bir bitlik dolu durum işareti ve bir bitlik geçerli durum işareti olmak üzere 2 bitten oluşur. Bu iki bitten sistemde bulunan harici port sayısınca(m) bulunur. Dolu durum işareti harici port giriş FIFO'sunun dolu olup olmadığını gösterir. Değeri '1' olduğunda dolu olduğu anlamına gelir. Geçerli durum işareti, harici port çıkış FIFO'sunun veri çıkışındaki verinin geçerli olup olmadığını gösterir. Değeri '1' olduğunda veri çıkışının geçerli olduğu anlamına gelir.

Öncelikli kodlayıcı aktif giriş işaretlerinden önceliği en yüksek olan girişin, giriş sırasını çıkış olarak belirler. Girişleri harici portların geçerli durum işaretleridir. Öncelik değerce küçük olan bağlantı numarasındadır. Öncelikli kodlayıcının girişi m bit çıkışı M bittir.

Veya kapısının girişleri harici portların geçerli işaretleridir. Girişi m bit ve çıkışı '1' bittir. Harici portların geçerli işaretlerinden herhangi biri '1' olduğunda çıkışı '1' olur. Girişlerinin hepsi '0' olduğunda çıkışı '0' olur.

Veri seçici, veri alınacak veya veri gönderilecek harici portun durum işaretlerinin seçilmesini sağlar. Veri girişleri harici portların durum işaretleridir. İki bitlik m adet veri girişi vardır. Seçim girişi M bitliktir ve kontrol ünitesi tarafından gönderilir. Çıkışı seçilen harici portun durum işaretleridir ve iki bittir.

4.1.3 Baytları ters çevir bloğu(BTÇ)

Baytları ters çevir bloğu bellekten okunan verilerin bayt sıralamasını tam ters olacak şekilde değiştirilmesini sağlar. Örneğin giriş baytları sırasıyla "abcd" ise çıkışı "dcba" olacaktır. Girişi bellek veri girişidir. Giriş ve çıkış veri genişliği L2 bittir.

4.1.4 Yazmaçlar

ÖDBEM tasarımı içerisinde 3 adet yazmaç bulunur. Bu yazmaçlar; veri yazmacı, yedek veri yazmacı ve sıfırlama yazmacıdır. Veri girişi, veri çıkışı ve yazma emir girişleri bulunur. Yazma emir girişleri kontrol ünitesi tarafından kontrol edilir.

4.1.4.1 Veri yazmacı

Veri yazmacı veri transferi sırasında veriyi tutan ana yazmaçtır. Veri transferinde kaynaktan okunan veri, veri yazmacına kaydedilir ve hedefe yazılır. Veri genişliği L2 bittir. Girişi Şekil4.2de görülen veri seçici 1 çıkışıdır. Çıkışı ÖDBEM veri çıkışıdır.

4.1.4.2 Yedek veri yazmacı

Yedek veri yazmacı, veri transferi esnasında transferin kesintiye uğraması halinde kaynağın verisini tutarak olası veri kaybını engeller. Girişi veri yazmacıdır. Veri genişliği L2 bittir.

4.1.4.3 Sıfırlama yazmacı

Sıfırlama yazmacı işlem birimlerinin sıfırlama girişlerini yönetir. Girişi bellek veri girişidir. Boyutu n bittir. Her bir biti bir işlem biriminin sıfırlama girişine bağlıdır.

4.1.5 Veri seçiciler

ÖDBEM tasarımı içerisinde 3 adet veri seçici bulunur. Seçim girişleri kontrol ünitesi tarafından yönetilir.

4.1.5.1 Veri seçici 1

Veri seçici 1, veri transferinin kaynağını belirler. Girişleri, veri seçici 2 veri çıkışı, veri seçici 3 veri çıkışı, kontrol ünitesi, bellek veri girişi, baytları ters çevir bloğu ve yedek veri yazmacı olmak üzere 6 tanedir. Girişlerin genişliği L2 bittir. Seçim girişi 3 bitliktir ve kontrol ünitesi tarafından kontrol edilir.

4.1.5.2 Veri seçici 2

Veri seçici 2'nin veri girişleri işlem birimleri veri girişleridir. L2 bitlik n tane veri girişi ve 1 tane veri çıkışı bulunur. Veri transferinin kaynağı işlem birimlerinden biri olduğunda ilgili işlem biriminin veri girişinin seçilmesini sağlar. Veri girişleri L2 bittir. Seçim girişi N bittir ve kontrol ünitesi tarafından yönetilir.

4.1.5.3 Veri seçici 3

Veri seçici 3'ün veri girişleri harici portlar veri girişleridir. L2 bitlik m tane veri girişi ve 1 tane veri çıkışı bulunur. Veri transferinin kaynağı harici portlardan biri olduğunda ilgili işlem biriminin veri girişinin seçilmesini sağlar. Veri girişleri L2 bittir. Seçim girişi M bittir ve kontrol ünitesi tarafından yönetilir.

4.1.6 Veri dağıtıcılar

Ödbem tasarımı içerisinde iki adet veri dağıtıcı bulunur. Seçim girişleri kontrol ünitesi tarafından yönetilir.

4.1.6.1 Veri dağıtıcı 1

Veri dağıtıcı 1 veri girişi 2 bitliktir. Kontrol ünitesi tarafından yönetilen kontrol işaretlerinin ilgili işlem birimine gönderilmesini sağlar. 2 bitlik n adet çıkışı vardır. Seçim girişi kontrol ünitesi tarafından yönetilir.

4.1.6.2 Veri dağıtıcı 2

Veri dağıtıcı 2 veri girişi 2 bitliktir. Kontrol ünitesi tarafından yönetilen kontrol işaretlerinin ilgili harici porta gönderilmesini sağlar. 2 bitlik n adet çıkışı vardır. Seçim girişi kontrol ünitesi tarafından yönetilir.

4.1.7 Kontrol ünitesi

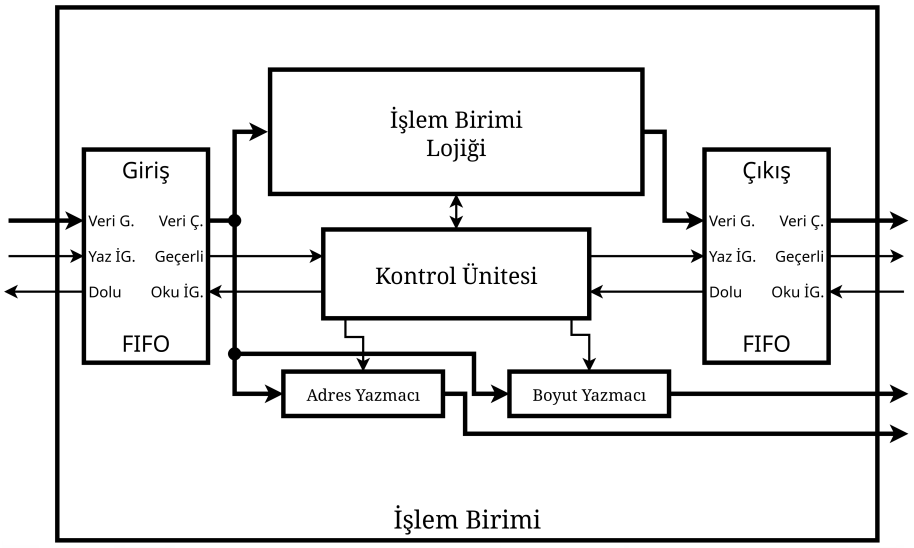
Kontrol ünitesi ÖDBEM'in kontrolünden sorumludur. Tüm kontrol işaretleri kontrol ünitesi tarafından kontrol edilir. ÖDBEM temel olarak işlemciden aldığı komutları gerçekleştirir. Buna ek olarak harici portlarda veri olduğunu işlemciye bildirmek ve işlem birimlerinde oluşan çıkış verisini belleğe yazma işlemlerini de yapar. Kontrol ünitesinin tasarımı Kontrol Ünitesi Tasarımı bölümünde detaylıca anlatılacaktır.

4.2 İşlem Birimleri

İşlem Birimleri, aldığı komut ve giriş verileri ile çalışan ve çıkış verisi üretebilen birimlerdir. ÖDBEM vasıtası ile bellekten okunan veri işlem birimlerine gönderilebilir, işlem birimlerinden okunan veri belleğe veya harici portlara gönderilebilir.

Bir işlem biriminin ÖDBEM kullanılarak haberleşebilmesi için ÖDBEM işlem birimleri arayüzüne sahip olması gerekir. Bunun için işlem birimi lojiğine bir üst modül eklenerek Ödbem işlem birimleri arayüzüne sahip bir işlem birimi haline getirilebilir. İşlem birimlerinin genel yapısı Şekil 4.5 de verilmiştir.

Bir işlem birimi; kontrol ünitesi, işlem birimleri lojiği, giriş FIFO'su, çıkış FIFO'su, adres yazmacı ve boyut yazmacından oluşur. Giriş FIFO'su ÖDBEM tarafından işlem birimlerine yazılacak veriyi tutar. Çıkış FIFO'su işlem sonucunda oluşacak çıkış verisini tutar. Adres yazmacı ve boyut yazmacı oluşacak çıkış verisinin yazılacağı bellek adresini ve boyutunu tutar. Bu adres ve boyut bilgisi işlemci tarafından belirlenerek ÖDBEM vasıtası ile işlem birimine gönderilir. Kontrol ünitesi, giriş FIFO'sundan okunan adres ve boyut bilgilerini adres ve boyut yazmaçlarına yazar.

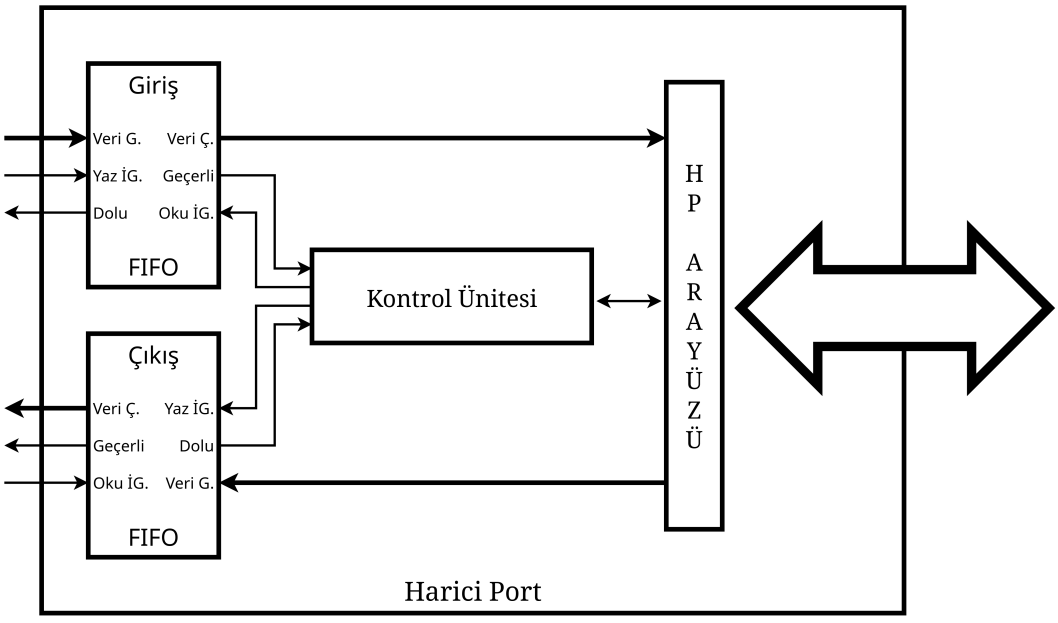


Şekil 4.5 : İşlem birimi genel yapısı.

Giriş FIFO'sundan okuduğu işlem birimi giriş verilerini işlem birimi lojiğine uygun şekilde yazar. İşlem birimi lojiğinde oluşan sonucu çıkış FIFO'suna yazar.

4.3 Harici Portlar

Harici portlar UART PCIE gibi fiziksel portların ÖDBEM ile haberleşmesini sağlar. Harici portların genel yapısı Şekil 4.6da verilmiştir.



Şekil 4.6 : Harici port genel yapısı.

Bir harici port; kontrol ünitesi, giriş FIFO'su, çıkış FIFO'su ve fiziksel port arayüzünden oluşur. Giriş FIFO'su ÖDBEM tarafından harici porta yazılacak veriyi tutar. Çıkış FIFO'su fiziksel porttan gelen veriyi tutar. Kontrol ünitesi, giriş FIFO'sundan okuduğu veriyi fiziksel portun yapısına uygun haberleşmeyi sağlayarak fiziksel port'a yazar. Fiziksel porttan gelen veriyi fiziksel portun yapısına uygun olarak çıkış FIFO'suna yazar. Fiziksel portlar ile yapılan haberleşmeden fiziksel portlardan gelen verinin ilk kelimesi gelecek mesajın boyutunu içermelidir.

4.4 ÖDBEM Komutları

ÖDBEM bir işlemci tarafında kontrol edilir. Bu kontrol bir takım komutlar ile sağlanır. İşlemci ile ÖDBEM arasında doğrudan bir arayüz bulunmadığı için komutların iletilmesi bellek üzerindeki belirli adresler üzerinden gerçekleşir. Kontrol adresleri ve açıklamaları Çizelge 4.2de verilmiştir.

Çizelge 4.2 : Bellek Üzerindeki Kontrol Adresleri

Adres	Veri Genişliği	Açıklama
Komut Adresi	4 x L1	Komut adresinden başlanarak tutulan 4 kelime sırasıyla İşlem Kodu, Kaynak, Hedef ve Boyut'dan oluşur.
İB Durum Adresi	L2	Her bitinde bir işlem birimi için durum işareti tutar. Bir işlem biriminde yapılan işlem bitip sonucu okunduğunda ÖDBEM tarafından İB DURUM ADRESİ'ndeki çalışılan işlem birimine ait bit '1' yapılır.
HP Durum Adresi	L2	Harici portlardan birinde veri olduğunu işlemciye bildirmek için kullanılır.
İB Sıfırlama Adresi	L2	Sıfırlama yazmacının değerini güncellemek için kullanılan adrestir.

İşlemci ile bellek arasındaki haberleşme belleğin birinci portu üzerinden işlemcinin veri yolu uzunluğuna eşit L1-bit uzunluğunda veri girişi-çıkışı, K-bit uzunluğunda adres ve 1-bitlik yazma ve/veya okuma izin girişleri kullanılarak gerçekleştirilir.

ÖDBEM ile bellek arasındaki haberleşme belleğin ikinci portu üzerinden tasarım sırasında değiştirilebilir K-bit uzunluğunda adres, L2-bit uzunluğunda veri girişi-çıkışı ve 1-bitlik yazma ve/veya okuma izin girişleri kullanılarak gerçekleştirilir.

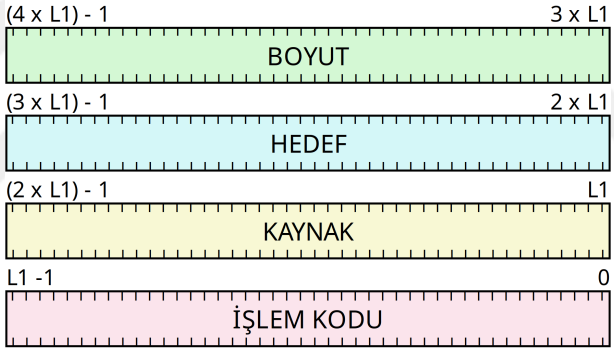
İşlemci tarafından gönderilecek olan komutlar L1-bit uzunluğunda 4 kelimeden oluşur. İşlemci kontrol komutlarını belleğin komut için ayrılan bölgesine komut adresinden başlayarak 4 adımda yazar. Komut adresi tasarım sırasında tasarımcı tarafından belirlenir.

ÖDBEM ikinci porttan komutları okur. L2 uzunluğu 4*L1'den büyük veya eşitse, ÖDBEM bellekten komut okumak için bir adet adres ve bellek yapısına bağlı olarak okuma izni işaretlerini belleğe gönderir. Okunan verinin düşük anlamlı 4*L1-biti komut olarak işlem görür.

L2 uzunluğu 4*L1'den küçükse, ÖDBEM bellekten komut okumak için 4*L1/L2 adet adres ve bellek yapısına bağlı olarak okuma izni işaretlerini belleğe gönderir. İlk okunan veri komutun en düşük anlamlı L2 biti, son okunan veri komutun en yüksek anlamlı L2 biti olacak şekilde, okunan veri parçaları birleştirerek komut elde edilir.

4.4.1 Komut formatı

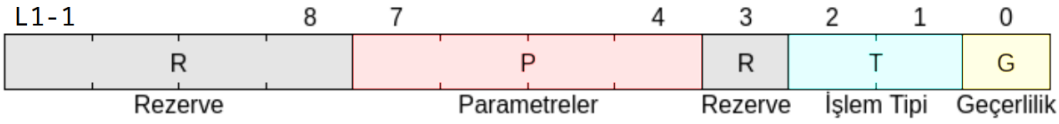
ÖDBEM komutunun 4 kelimesi İşlem Kodu, Kaynak, Hedef ve Boyut olarak adlandırılır. Şekil 4.7'de komut formatı verilmiştir.



Şekil 4.7 : Komut Formatı.

4.4.1.1 İşlem kodu

Komutun en anlamsız L1 biti işlem kodunu ifade eder. İşlem kodunun en anlamsız 8 biti komutun geçerliliğini, tipini ve bu komuta ilişkin parametreleri içerir. Şekil 4.8'de işlem kodu ve bileşenleri verilmiştir.



Şekil 4.8 : İşlem Kodu.

Geçerlilik

Geçerlilik işlem kodunun en düşük anlamlı 1-bitlik değeridir. İşlemci tarafından belleğe bir komut yazıldığında komut geçerlilik değeri "1" yapılır. Komut geçerlilik değerinin "1" olması ÖDBEM tarafından "işleme başla" kontrol işareti olarak kullanılır.

Çizelge 4.3 : Geçerlilik Değerleri

Geçerlilik Değeri	Anlamı
0	Komut Geçersiz
1	Komut Geçerli

ÖDBEM komutu okuyup işledikten sonra geçerlilik değerini "0" yapmak amacıyla komut adresine "0" değerini yazar. Bu aynı zamanda işlemci tarafından ÖDBEM'in verilen komutu işleyip bitirdiğini gösteren bir durum işareti olarak kullanılır. ÖDBEM içerisinde yer alan kontrol ünitesi komut okuma durumunda iken geçerlilik değeri "0" olduğu müddetçe, ÖDBEM belleğin komut adresindeki'ndeki değeri okur. Geçerlilik değeri "1" olduğunda komut okumayı tamamlar ve işlemeye başlar. Çizelge 4.4'de geçerlilik değeri geçiş çizelgesi gösterilmiştir.

Çizelge 4.4 : Geçerlilik Geçiş Çizelgesi.

Şimdiki Değer	Sonraki Değer	Geçişi Yapan	Anlamı
0	0	–	Yeni komut henüz yazılmadı.
0	1	İşlemci	Yeni komut yazıldı.
1	0	ÖDBEM	Komutun işlenmesi tamamlandı, yeni komut yazılabilir.
1	1	–	Komutun işlenmesi sürüyor, yeni komut yazılamaz.

İşlem Tipi ve Parametreler

İşlem Tipi, işlem kodunun 2. ve 1. bitleri arasındaki 2-bitlik değeridir. Parametler, işlem kodunun 7. ve 4. bitler arasındaki 4 bitlik değer olup işlem tipine göre kullanımları farklılık göstermektedir.

ÖDBEM kontrolü, Gönder, Kopyala, Al ve Sıfırla olmak üzere 4 temel komutla sağlanır.

- Gönder komutu bellekten okuyup harici portlara veya işlem birimlerine göndermeyi ifade eder.
- Kopyala komutu belleğin kendi içindeki bir adresten başka bir adrese kopyalama işlemini ifade eder.
- Al komutu harici portlardan veri okuyup belleğe yazmayı ifade eder.
- Sıfırla komutu işlem birimleri içinde yer alan yazmaçların sıfırlanma işlemini(reset) ifade eder.

Çizelge 4.5'de işlem tipleri verilmiştir.

Çizelge 4.5 : İşlem Tipleri

İşlem Tipi Değeri	Anlamı
00	Gönder
01	Kopyala
10	Al
11	Sıfırla

4.4.1.2 Kaynak

Kaynak, yapılacak veri transferinin kaynağını belirtir. Birim numarası veya adres olabilir.

4.4.1.3 Hedef

Hedef, yapılacak veri transferinin hedefini belirtir. Birim numarası veya adres olabilir.

4.4.1.4 Boyut

Boyut, yapılacak veri transferinin boyutunu belirtir. Birim olarak bayt kullanılır.

4.4.2 Komutlar

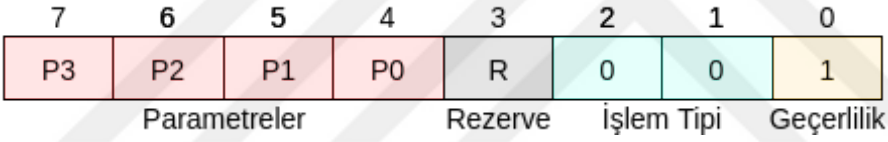
ÖDBEM; gönder, kopyala, al ve sıfırla olmak üzere 4 temel komutla kontrol edilir.

4.4.2.1 Gönder

Gönder komutu bellekten okunan verinin harici portlara veya işlem birimlerine gönderilmesini ifade eder.

- Şekil 4.7'de gösterilen komut formatında Kaynak kelimesi bu komutta gönderilecek verinin okunacağı bellek adresini ifade eder.
- Şekil 4.7'de gösterilen komut formatında Boyut kelimesi gönderilecek verinin bayt uzunluğunu ifade eder.
- Şekil 4.7'de gösterilen komut formatında Hedef kelimesi gönderilecek verinin gönderileceği işlem biriminin veya harici portun numarasını ifade eder

Şekil 4.9 da Gönder komutuna ilişkin işlem kodu verilmiştir.



Şekil 4.9 : Gönder komutu işlem kodu.

Gönder komutunun parametreleri Çizelge 4.6'de verilmiştir.

Çizelge 4.6 : Gönder komutu parametreleri.

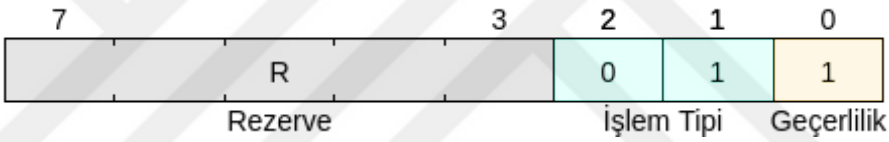
Parametre	'0' Anlamı	'1' Anlamı
P0	İşlem birimine gönder.	Harici porta gönder.
P1	İşlem biriminde başlatılan işlem sonuçlandığında, sonucunu belleğe gönder.	İşlem biriminde başlatılan işlem sonuçlandığında, sonucunu harici porta gönder.
P2	Kopyalama yaparken bayt sıralamasını değiştirme.	Kopyalama yaparken bayt sıralamasını ters çevir.
P3	Kaynak adresinden başla, adres değerini arttırarak Boyut kadar veri gönder.	Kaynak adresinden başla, adres değerini azaltarak Boyut kadar veri gönder.

4.4.2.2 Kopyala

Kopyala komutu belleğin bir adresinden diğer adresine kopyalama yapar.

- Şekil 4.7'de gösterilen komut formatında Kaynak kelimesi kopyalanacak verinin bellek adresini ifade eder.
- Şekil 4.7'de gösterilen komut formatında Boyut kelimesi kopyalanacak verinin bayt uzunluğunu ifade eder.
- Şekil 4.7'de gösterilen komut formatında Hedef kelimesi kopyalanan verinin yazılacağı bellek adresini ifade eder.

Şekil 4.10'de kopyala komutuna ilişkin işlem kodu verilmiştir.



Şekil 4.10 : Kopyala komutu işlem kodu.

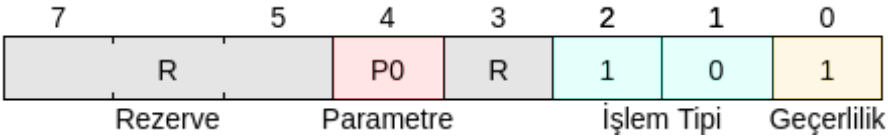
Kopyala komutunda parametre kullanılmaz.

4.4.2.3 Al

Al komutu harici portlardan veri okuyup belleğe yazmayı ifade eder.

- Şekil 4.7'de gösterilen komut formatında Kaynak kelimesi P0 = 1 olduğunda verinin okunacağı harici portu belirtir. P0 = 0 olduğu durumda kaynak kelimesi kullanılmaz.
- Şekil 4.7'de gösterilen komut formatında Boyut kelimesi al komutunda kullanılmaz. Kopyalanacak verinin boyutu harici porttan okunacak verinin ilk kelimesinden okunur. Harici porttan gelen verinin ilk kelimesi gelen paketin boyutunu içermelidir.
- Şekil 4.7'de gösterilen komut formatında Hedef kelimesi alınacak verinin yazılacağı bellek adresini ifade eder.

Şekil 4.11'de al komutuna ilişkin işlem kodu verilmiştir.



Şekil 4.11 : Al komutu işlem kodu.

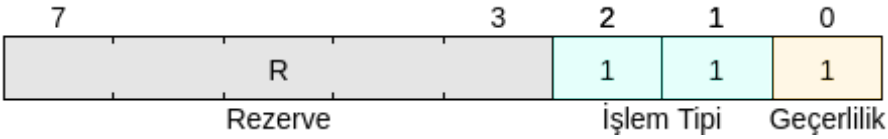
Al komutunun parametresi Çizelge 4.7'de verilmiştir.

Çizelge 4.7 : Al komutu parametreleri.

Parametre	'0' Anlamı	'1' Anlamı
P0	Hangi harici porttan veri alınacağı Öncelikli kodlayıcı ile belirlenir.	Hangi harici porttan veri alınacağı "Kaynak" kelimesi ile Belirlenir..

4.4.2.4 Sıfırla

Sıfırla komutu ÖDBEM içerisinde bulunan sıfırlama yazmacının değerini güncellemek için kullanılır. Sıfırla işlemi için önceden belirlenen İB SIFIRLAMA ADRESİ'ne, sıfırlanmak istenen işlem birimine veya birimlerine ait bitlerin değeri '1' olacak şekilde ilgili değer yazılır. Sıfırlama komutu gönderildiğinde İB SIFIRLAMA ADRESİ'ndeki değer, sıfırlama yazmacına yazılacaktır. Sıfırlama işlemini bitirmek için de ilgili bitlerin değeri '0' yapılarak tekrar Sıfırla komutu gönderilmesi gerekir. Şekil 4.12'de sıfırla komutuna ilişkin işlem kodu verilmiştir.



Şekil 4.12 : Sıfırla komutu işlem kodu.

Sıfırla komutunda parametre kullanılmaz. Şekil 4.7'de gösterilen Kaynak, Hedef ve Boyut kelimeleri sıfırla komutunda kullanılmaz.

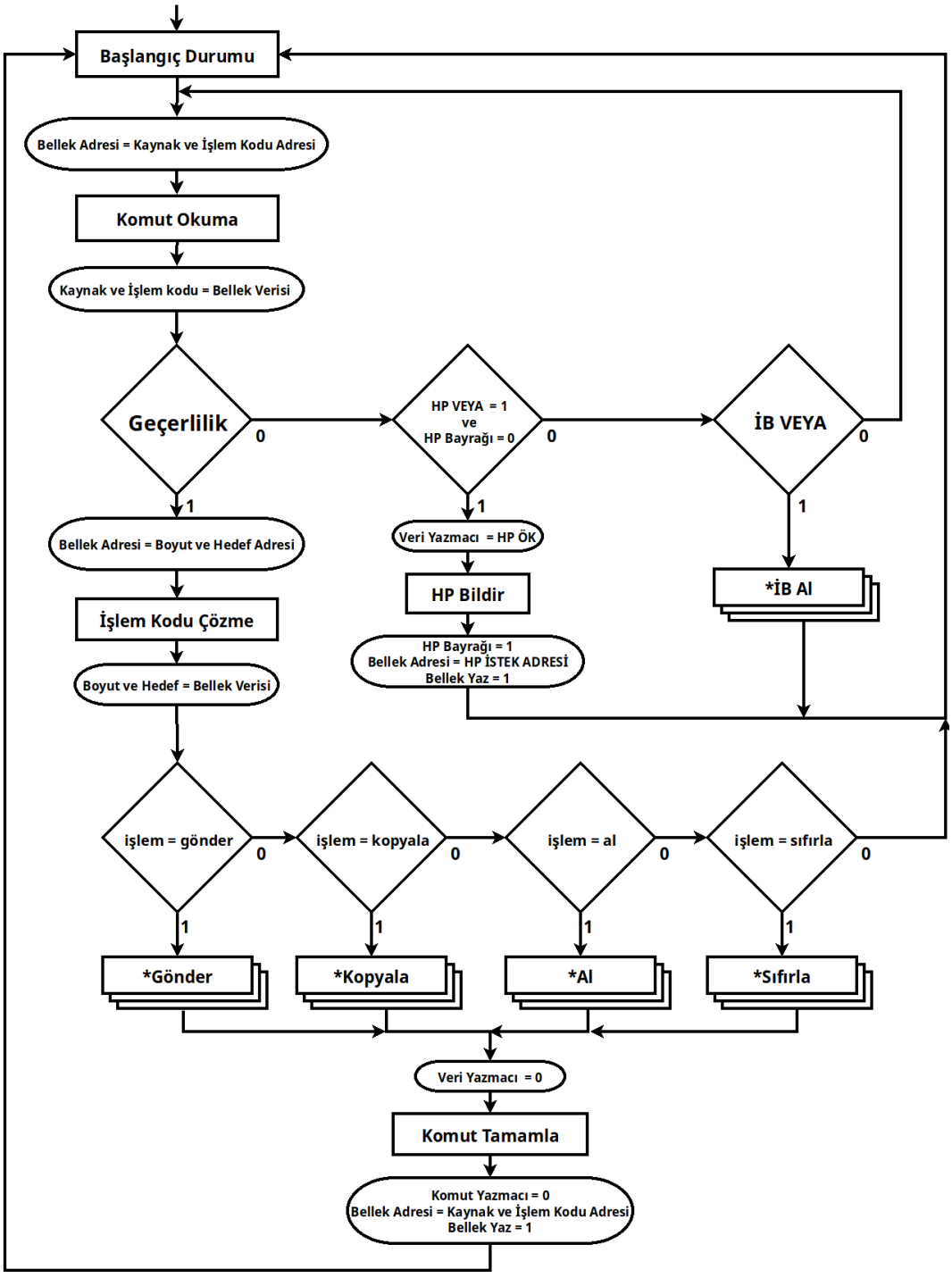
4.5 Kontrol Ünitesi Tasarımı

Tasarım parametrelerini verildiği Çizelge 4.1'de verilen L1 ve L2 parametreleri için farklı tasarım seçenekleri uygulanabilir olsa da kontrol ünitesi tasarımı L2 = 2 x L1 için açıklanmıştır. Küçük değişiklikler ile farklı L1 ve L2 bit genişlikleri için de kolaylıkla uygulanabilir. ÖDBEM'e ait algoritmik durum makinası Şekil 4.13'de verilmiştir.

ÖDBEM komutları işlemci tarafından belleğin KOMUT ADRESİ olarak belirlenen adresine yazılır. ÖDBEM Kontrol Ünitesi başlangıç durumunda bellek adresi çıkışına KAYNAK VE İŞLEM KODU ADRESİ'ni gönderir. Komut okuma durumunda komutun Şekil 4.7'de görülen işlem kodu kelimesinin Şekil 4.8'de görülen geçerlilik kısmı '1' olduğunda işlenmeye hazır bir komut olduğu anlamına gelir.

Komut okuma durumunda Şekil 4.8'de'de görülen işlem tipi kısmı çözülerek ilgili komut tipine göre Gönder, Kopyala, Al, Sıfırla işlemlerinden biri ilgili algoritmik durum makinesine geçilerek yapılır. Şekil 4.13 'de görülen "*" işareti içeren kutucuklar sadece 1 durumu değil birden fazla durumu içeren bir algoritmik durum makinesini gösterir. Komut gerçekleştirilir ve ÖDBEM başlangıç durumuna geri döner.

Eğer işlemciden gelen geçerli bir komut yoksa, yani okunan işlem kodu'nun geçerlilik değeri '0' ise, harici portlarda veri olup olmadığı Şekil 4.4 de görülen harici portlar giriş bloğunun VEYA çıkışına bakılarak kontrol edilir. VEYA çıkışının '1' olması harici portlardan herhangi birinde veri olduğu anlamına gelir. Eğer VEYA çıkışı '1' ise Kontrol ünitesi içerisinde yer alan HP Bayrağı kontrol edilir. HP Bayrağı'nın '1' olması , harici portlardan herhangi birinde veri olduğunun işlemciye bildirildiği anlamını taşır. HP Bayrağı '0' ve VEYA Çıkışı '1' olduğunda, bellek adres çıkışına tasarım sırasında belirlenmiş olan HP Durum Adresi yazılır. Veri çıkışına Şekil 4.4 'de görülen Öncelikli Kodlayıcı'nın çıkışı verilir ve belleğe gerekli yazma işaretleri gönderilir. Aynı zamanda HP Bayrağı '1' yapılır. Al komutu gönderilip herhangi bir harici porttan veri okunana dek HP Bayrağı'nın değeri değişmez.



Şekil 4.13 : ÖDBEM'e ait algoritmik durum makinası.

Eğer harici portlarda veri yok veya veri olduğu işlemciye bildirilmişse Şekil 4.3'de görülen işlem birimleri giriş bloğunun VEYA çıkışına bakılarak işlem birimleride veri olup olmadığı kontrol edilir. VEYA çıkışı '1' ise Şekil 4.13'de görülen İB Al işlemine gidilir. İşlem birimi verisi okunarak belleğe ya da bir harici porta yazılır.

4.5.1 Gönder

Şekil 4.13'de görülen *Gönder işlemine ililşkin algoritmik durum makinesi Şekil 4.14'de verilmiştir.

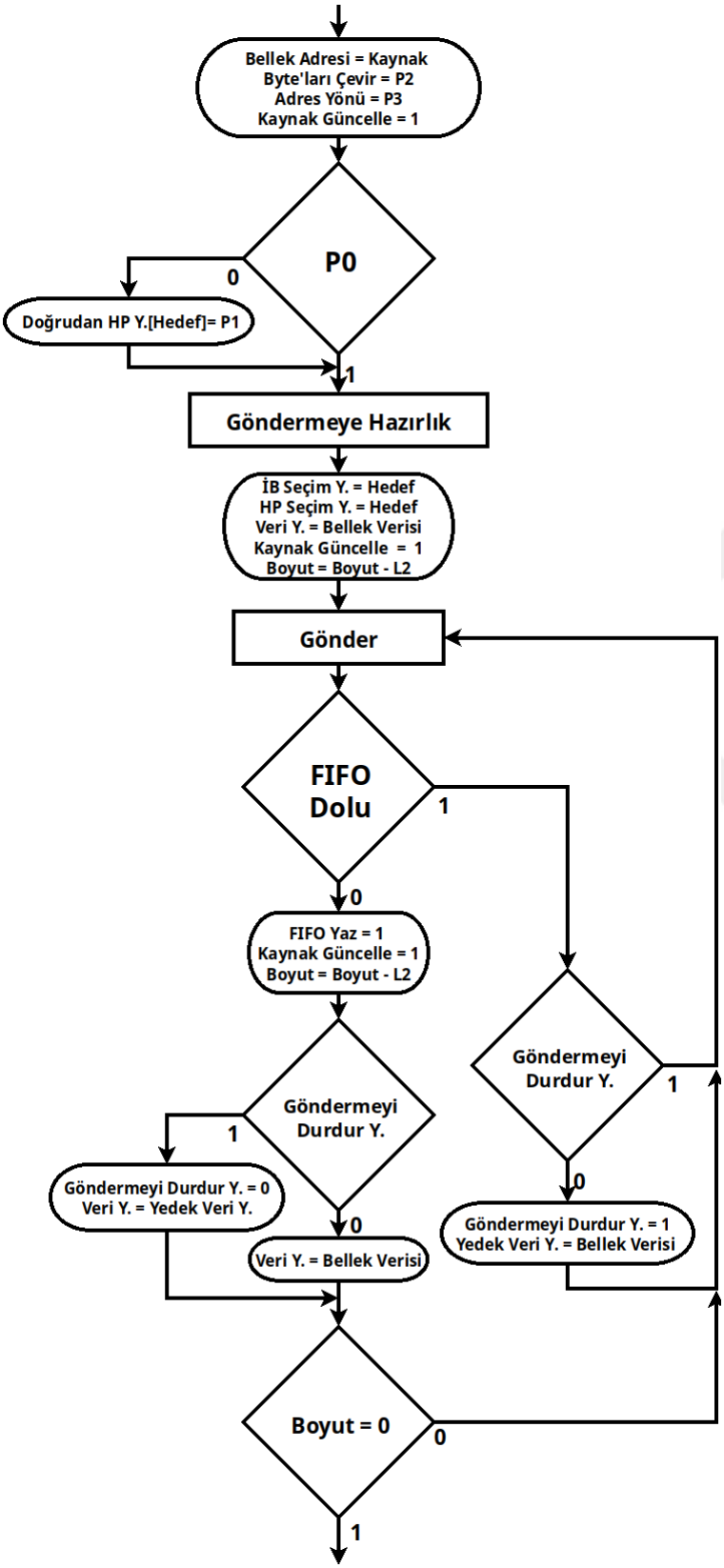
Şekil 4.13'de görülen algoritmik durum makinesinin İşlem Kodu Çözme durumunda, işlem kodu yazmacının Şekil 4.8'de görülen İşlem Tipi değeri '0' olduğunda *Gönder komutunu işlemek için Şekil 4.14'de görülen Göndermeye Hazırlık durumuna geçilir. Bu geçiş esnasında Bellek adresine Kaynak yazmacının değeri verilir. Baytları Çevir, Adres Yönü yazmaçları işlem kodunun parametreler kısmından alınır. Kaynak Güncelle işareti '1' yapılır. P0 parametresinin değerine göre gönderme işleminin işlem birimine veya harici portalara yapılacağı belirlenir. Eğer işlem birimlerine veri gönderilecekse Doğrudan HP yazmacının veri gönderilecek olan işlem birimine ait bitine P1 parametresinin değeri yazılır.

Baytları Çevir yazmacı ÖDBEM Kontrol Ünitesi içerisinde bulunan, bellek verisinin doğrudan mı yoksa Şekil 4.2'de görülen Baytları Ters Çevir(BTÇ) bloğundan geçirilmiş olarak mı gönderileceği bilgisini tutar. Eğer Baytları Çevir yazmacının değeri '1' ise veri yazmacına veri yazılırken bellek verisi olarak baytları ters çevir bloğunun çıkışı kullanılır.

Adres Yönü yazmacı birden fazla adımda yapılacak veri gönderme işinde bellekten veri okunurken ardışık iki adım arasında adresin artmasını veya azalmasını kontrol eder. Adres Yönü '1' ise bir sonraki adımda Kaynak değeri artar, '0' ise azalır. Kaynak Güncelle işareti, Adres Yönü yazmacının değerine göre Kaynak yazmacının değerinin güncellenmesini sağlar.

Doğrudan HP yazmacı ÖDBEM Kontrol Ünitesi içerisinde bulunan, işlem birimi sayısınca bitten oluşan bir yazmaçtır. İşlem birimlerine gönderilecek veri ile başlatılan işlemin sonucunun belleğe mi yoksa harici portlardan birine mi yazılacağı bilgisini tutar. Doğrudan HP yazmacının Hedef ile ilgili biti '1' olduğunda işlem biriminde başlatılan işlemin sonuç verisi harici portlardan birine yazılır.

Göndermeye Hazırlık durumunda İB Seçim ve HP Seçim yazmaçlarına Hedef yazmacının değeri yazılır. Veri yazmacına bellek verisi yazılır. Kaynak yazmacı



Şekil 4.14 : Gönder İşlemi Algoritmik Durum Makinesi

güncellenir ve Boyut yazmacının değeri L2 bayt kadar azaltılarak Gönder durumuna geçilir.

Gönder durumunda, FIFO Dolu işareti kontrol edilir. FIFO Dolu ve FIFO Yaz işaretleri Şekil 4.6 ‘de görülen HP Giriş FIFO veya Şekil 4.5’de görülen İB Giriş FIFO’nun işaretleri arasından P0 parametresinin değerine göre seçilir. P0 parametresi ‘0’ ise işlem biriminin, ‘1’ ise harici portların işaretleri kullanılır.

FIFO Dolu işareti ’1’ ise Göndermeyi Durdur yazmacına bakılır. Göndermeyi Durdur yazmacı ÖDBEM Kontrol Ünitesi içerisinde bulunan, FIFO’ya yazma işlemleri esnasında veri kaybını engellemek amacıyla kullanılan yazmaçtır.

- Göndermeyi Durdur yazmacının değeri ‘0’ ise ‘1’ yapılır ve bellek verisi Yedek Veri yazmacına kaydedilir.
- Göndermeyi Durdur yazmacının değeri ‘1’ ise bir işlem yapılmadan Gönder durumunda kalınır.

FIFO Dolu işareti ‘0’ ise FIFO Yaz kontrol işareti ‘1’ yapılır. Kaynak güncellenir. Boyut’un değeri L2-Bayt kadar azaltılır. Göndermeyi Durdur yazmacının değeri kontrol edilir.

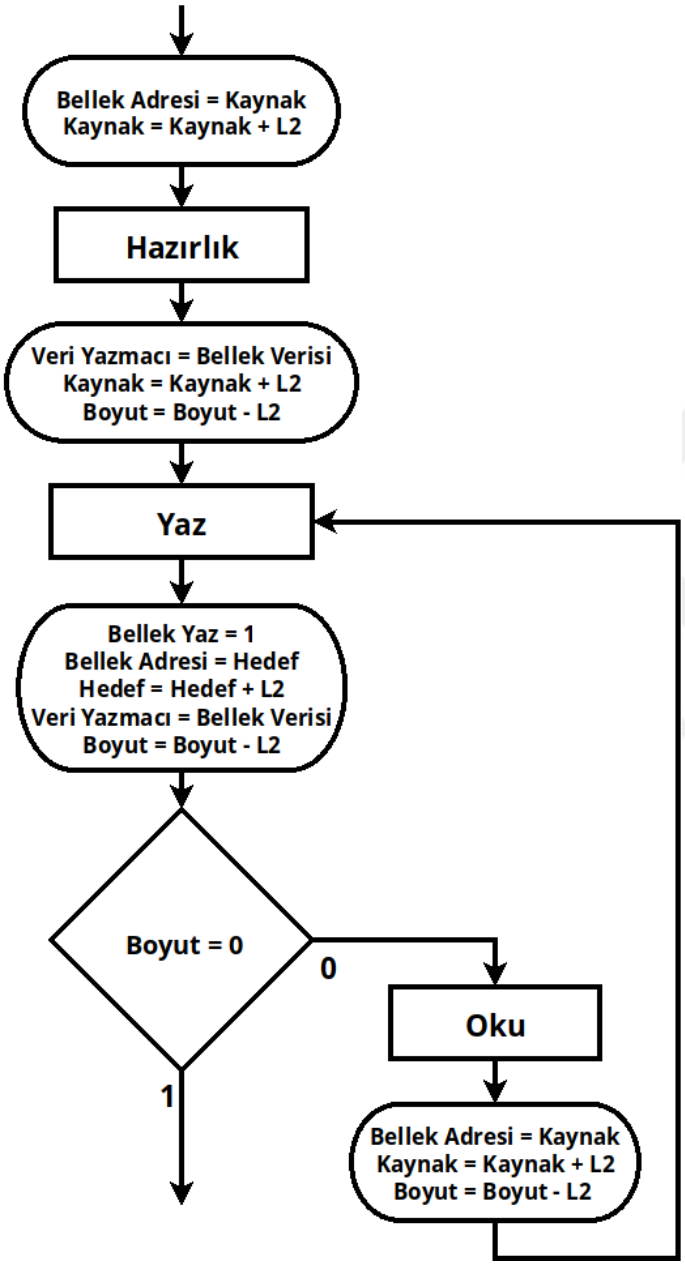
- Göndermeyi Durdur yazmacının değeri ‘1’ ise ‘0’ yapılır. Veri yazmacına Yedek Veri yazmacında tutulan veri yazılır.
- Göndermeyi Durdur yazmacının değeri ‘0’ ise Veri yazmacına bellek verisi yazılır.

Boyut yazmacının değeri ‘0’ olana kadar Gönder durumunda kalınır. Boyut yazmacının değeri ‘0’ olduğunda Şekil 4.13’de görülen *Gönder işlemi tamamlanmış olur.

4.5.2 Kopyala

Şekil 4.13’de görülen *Kopyala işlemine ilişkin algoritmik durum makinesi Şekil 4.15’de verilmiştir. Şekil 4.13’de görülen algoritmik durum makinesinin İşlem Kodu Çözme durumunda, işlem kodu yazmacının Şekil 4.8’de görülen İşlem Tipi değeri ‘1’

olduğunda Kopyala komutunu işlemek için Şekil 4.15'de görülen Hazırlık durumuna geçilir. Bu geçiş sırasında bellek adresine Kaynak yazmacının değeri verilir. Kaynak yazmacının değeri L2 kadar artırılır.



Şekil 4.15 : Kopyala İşlemi Algoritmik Durum Makinesi

Hazırlık durumunda Veri yazmacına bellek verisi yazılır. Kaynak, L2 kadar arttırılır. Boyut, L2 kadar azaltılır. Yaz durumuna geçilir.

Yaz durumunda, bellek adresine Hedef yazmacının değeri verilir ve Bellek Yaz emir çıkışı ‘1’ yapılır. Hedef yazmacının değeri L2 arttırılır. Boyut yazmacının değeri L2 azaltılır. Veri yazmacına bellek verisi kaydedilir. Boyut değerine bakılır.

- Eğer Boyut ’0’ ise *Kopyala işlemi tamamlanmış demektir.
- Değilse Oku durumuna gidilir.

Oku durumunda bellek adresine Kaynak yazmacının değeri verilir. Kaynak L2 kadar arttırılır. Boyut L2 kadar azaltılır. Yaz durumuna geçilir.

4.5.3 Al

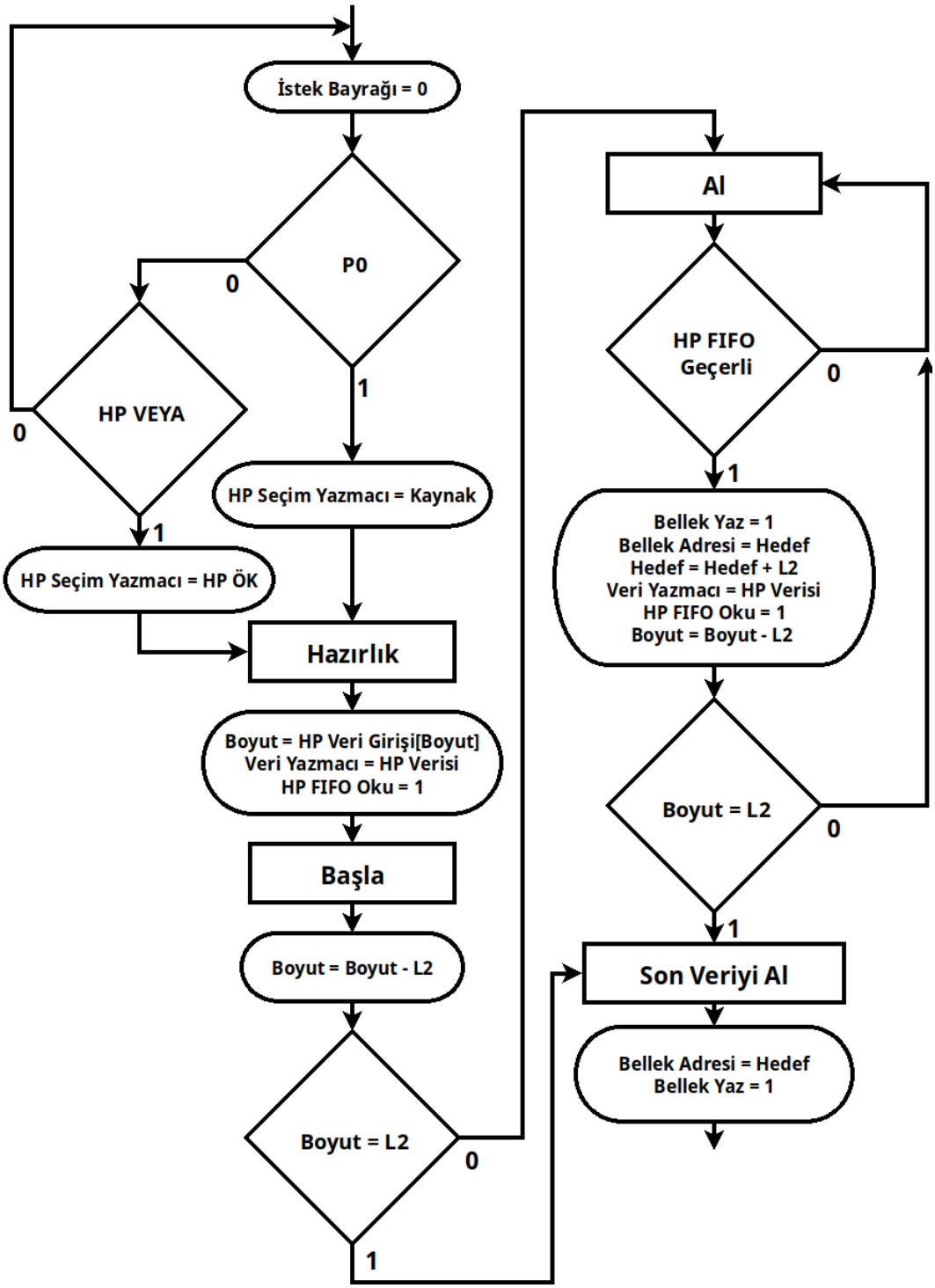
Şekil 4.13’de görülen *Al işlemine ililşkin algoritmik durum makinesi Şekil ’da verilmiştir.

Şekil 4.13’de görülen algoritmik durum makinesinin İşlem Kodu Çözme durumunda, işlem kodu yazmacının Şekil 4.8’de görülen İşlem Tipi değeri ‘10’ olduğunda Kopyala komutunu işlemek için Şekil 4.16’da görülen Hazırlık durumuna geçilir. Bu geçiş sırasında İstek Bayrağı’nın değeri ‘0’ yapılır. Şekil 4.11’de görülen P0 parametresinin değerine bakılır.

- ’1’ ise HP Seçim yazmacına Kaynak yazmacının değeri yazılır.
- ’0’ ise Şekil 4.4 ’de görülen VEYA çıkışına bakılır.
  - ’0’ ise bir işlem yapılmadan İşlem Kodu Çözme durumunda kalınır.
  - ‘1’ ise HP Seçim yazmacına Şekil 4.4’de görülen Öncelikli Kodlayıcı çıkışı yazılır.

Hazırlık durumunda, Boyut yazmacına HP veri girişinin boyut kısmı kaydedilir. Veri yazmacına HP verisi yazılır. HP Oku kontrol işareti ’1’ yapılır ve Başla durumuna geçilir.

Başla durumunda Boyut yazmacının değeri L2 kadar azaltılır ve Boyut kontrol edilir.



Şekil 4.16 : Al İşlemi Algoritmik Durum Makinesi

- L2'ye eşitse, harici porttan alınacak verinin tek kelimelik olduğu anlamına gelir ve Son Veriyi Al durumuna geçilir.
- Değilse Al durumuna geçilir.

Al durumunda HP FIFO Geçerli işareti kontrol edilir.

- '0' ise herhangi bir işlem yapılmadan Al durumunda kalınır.
- '1' ise bellek adresine Hedef yazmacının değeri verilir ve Bellek Yaz çıkışı '1' yapılır. Hedef yazmacının değeri L2 kadar arttırılır. Veri yazmacına HP verisi yazılır. HP FIFO Oku çıkışı '1' yapılır. Boyut, L2 kadar azaltılır ve Boyut kontrol edilir.
  - L2'ye eşitse, son kelime belleğe yazılmak üzere Son Veriyi Al durumuna geçilir.
  - Değilse Al durumunda kalınır.

Son Veriyi Al durumunda bellek adresine hedef adresi verilir ve Bellek Yaz çıkışı '1' yapılır. *Al işlemi tamamlanmış olur.

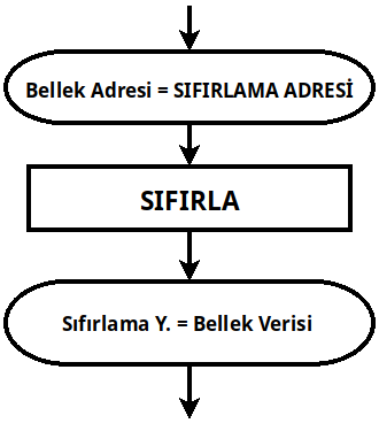
4.5.4 Sıfırla

Şekil 4.13'de görülen *Sıfırla işlemine ilişkin algoritmik durum makinesi Şekil 4.17'de verilmiştir.

Şekil 4.13'de görülen algoritmik durum makinesinin İşlem Kodu Çözme durumunda, işlem kodu yazmacının Şekil 4.8'de görülen İşlem Tipi değeri '11' olduğunda *Sıfırla komutunu işlemek için Şekil 4.17'de görülen Sıfırla durumuna geçilir. Bu geçiş esnasında bellek adresi çıkışına SIFIRLAMA ADRESİ değeri yazılır.

Sıfırla durumunda bellek verisi, Şekil 4.2'de görülen sıfırlama yazmacına yazılır.

Şekil 4.13'de görülen *Sıfırla işlemi tamamlanmış olur.



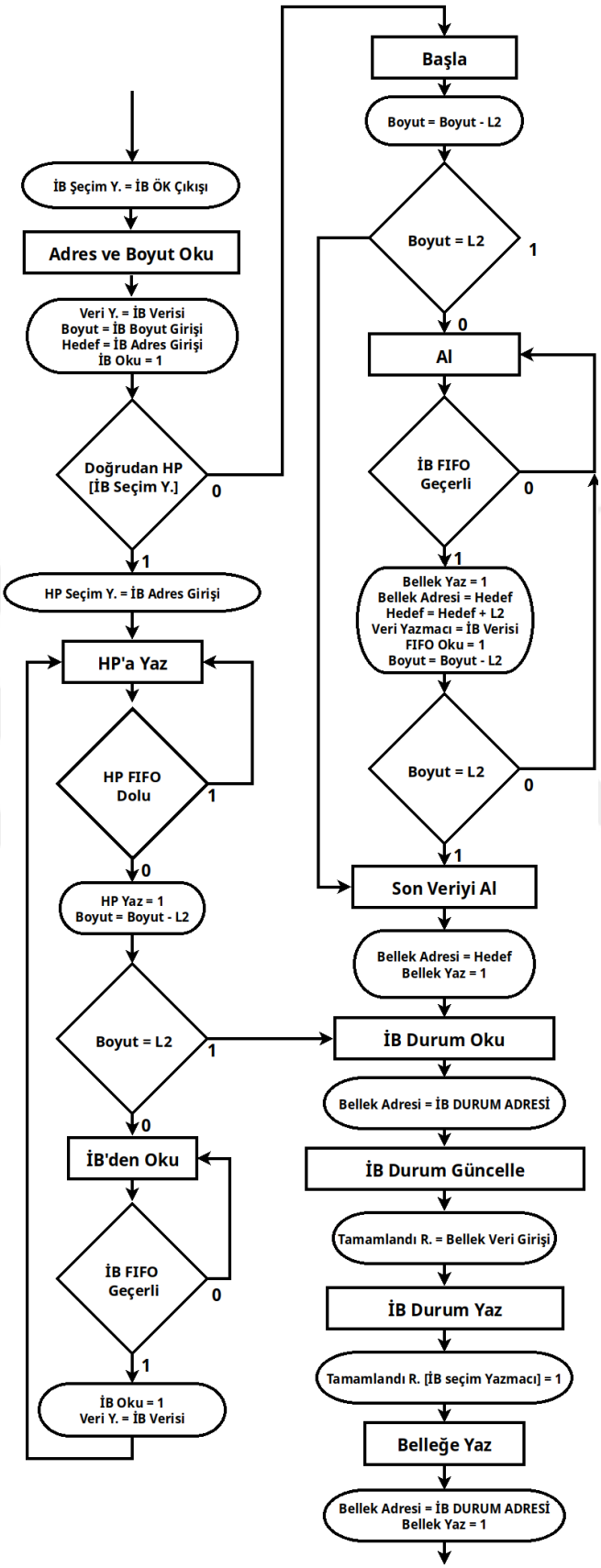
Şekil 4.17 : Sıfırla İşlemi Algoritmik Durum Makinesi

4.5.5 İB al

Şekil 4.13'de görülen *İB Al işlemine ilişkin algoritmik durum makinesi Şekil 4.18'de verilmiştir.

Şekil 4.13'de verilen algoritmik durum makinesinde komut okuma durumundan İB Al işleminin şartları sağlandığında Şekil 4.18 de görülen durum makinesinin Adres ve Boyut Oku durumuna geçilir. Bu geçiş esnasında ilgili işlem birimi Şekil 4.3'de görülen İşlem Birimleri Giriş Bloğu Öncelikli Kodlayıcı çıkışı kullanılarak belirlenir ve ÖDBEM Kontrol Ünitesi'nde bulunan İB Seçim Yazmacı'na yazılır. Şekil 4.3'da görülen İB Giriş Bloğu'nun seçim girişi, Şekil 4.2'de görülen Veri Seçici 2 (VS2) ve Veri Dağıtıcı 2 (VD2)'nin seçim girişleri İB Seçim Yazmacı'na bağlıdır.

Adres ve Boyut Oku durumunda Veri Yazmacı'na ilgili işlem biriminin giriş verisini seçen Veri Seçici 2 çıkışı yazılır. Hedef yazmacına işlem birimi adres girişi, boyut yazmacına işlem birimi boyut girişi yazılır. İşlem Birimi Oku kontrol işareti 1 yapılır. Doğrudan HP Yazmacı işlem birimi sayısınca bitten oluşan bir yazmaç olup Komut Formatı-İşlem Tipleri bölümünde anlatılan Gönder işleminin özel bir durumunu ifade eder. Adres ve Boyut Oku durumunda Doğrudan HP Yazmacı'nın ilgili bitine bakılarak işlem biriminden okunacak verinin Belleğe veya bir Harici Porta yazılacağına karar verilir.



Şekil 4.18 : İB Al İşlemi Algoritmik Durum Makinesi

Seçilen işlem birimi ile ilgili Doğrudan HP bitinin değeri 0 ise işlem birimi verisinin belleğe yazılması gerekiyor demektir. Başla durumuna gidilir. Başla durumunda Boyut, L2 bayt azaltılır. Aynı zamanda Boyut yazmacının değeri kontrol edilir.

- L2 bayttan oluşan bir veri belleğe gönderilecekse yapılan işlem tek adımlık olacağından Son Veriyi Al durumuna geçilir,
- Gönderilecek veri L2-Bayttan fazla ise Al durumuna geçilir ve ilgili İB Çıkış FIFO'sunun geçerli işareti kontrol edilir.
  - '0' ise herhangi bir işlem yapılmadan aynı durumda kalınır.
  - '1' ise belleğin Hedef Yazmacı'nda tutulan adresine Veri Yazmacı'nın değeri yazılır. Hedef Yazmacı'nın değeri L2-bayt kadar arttırılır. Veri Yazmacı'na İB verisi yazılır. İB FIFO Oku işareti '1' yapılır. Boyut değeri L2-bayt kadar azaltılır. Boyut değeri kontrol edilir ve Boyut L2 olana dek bu durumda kalınarak aynı işlemler tekrarlanır. Boyut L2'ye eşit olduğunda son veri alınmak üzere Son Veriyi Al durumuna geçilir.

Son Veriyi Al durumunda İB FIFO'sundan okunacak veri bitmiş olduğundan sadece belleğe yazma işlemi yapılır ve İB Durum Oku durumuna geçilir.

İB DURUM ADRESİ düşük anlamlı bitten başlanarak her bir işlem birimi için bir bitlik durum işaretinin tutulduğu adrestir. ÖDBEM tarafından işlem biriminden sonuç verisi okunduktan sonra İB DURUM ADRESİ'ndeki ilgili bitin değeri '1' yapılır.

- İB Durum Oku durumunda belleğin adres çıkışı İB DURUM ADRESİ yapılır.
- İB Durum Güncelle durumunda ÖDBEM Kontrol Ünitesi içerisinde bulunan Tamamlandı Yazmacı'na bellek veri girişinin değeri yazılır.
- İB Tamamlandı Yaz durumunda Tamamlandı Yazmacı'nın ilgili biti '1' yapılır.
- Belleğe Yaz durumunda Tamamlandı Yazmacı'nın değeri belleğin İB DURUM ADRESİ'ne yazılır. Şekil 13'de görülen *İB Al işlemi tamamlanmış olur.

Adres ve Boyut Oku durumunda ilgili Doğrudan HP bitinin değeri ‘1’ ise işlem biriminden okunacak veri harici portlardan birine yazılacak demektir. Hangi harici porta yazılacağını İB Adres Girişi’nin değeri belirler. HP Seçim Yazmacı’na, İB Adres Girişi’nin değeri yazılarak HP’a Yaz durumuna geçilir.

HP’a Yaz durumunda ilgili harici portun Şekil 3’de görülen Giriş FIFO’sunun dolu durum işareti kontrol edilir.

- Giriş FIFO’su dolu ise bir işlem yapmadan HP’a Yaz durumunda kalınır.
- Giriş FIFO’su dolu değilse HP Yaz kontrol işaretinin değeri 1 yapılır. Boyut yazmacının değeri L2-bayt kadar azaltılır. Boyut yazmacının değeri kontrol edilir.
  - Boyut L2-Bayt’a eşitse kopyalama işi tamamlanmış olacağından İB Durum Oku durumuna geçilir,
  - Boyut L2-Bayt’a eşit değilse işlem biriminden yeni veriyi okumak için İB’den Oku durumuna geçilir.

İB’den Oku durumunda İşlem biriminin geçerli işareti kontrol edilir.

- ‘1’ ise İB Oku kontrol işaretinin değeri ‘1’ yapılır. Veri yazmacına işlem birimi verisi yazılır ve HP’a Yaz durumuna dönülür.
- ’0’ ise İB’den Oku durumunda kalınır.

5. DONANIM TASARIMI VE UYGULAMA

5.1 Donanım Tasarımı

ÖDBEM tasarımı için Tasarım bölümünde anlatılan tasarım detaylarına uygun olacak şekilde VHDL donanım tanımlama dili kullanılarak kod yazılmıştır. Yazılan kodda Çizelge 4.1'de verilen parametrelerin kolayca değiştirilebilir olmasına önem verilmiştir. Çizelge 5.1'de sentez ve gerçekleme için seçilen tasarım parametreleri verilmiştir.

Çizelge 5.1 : Sentez ve Gerçekleme için Seçilen ÖDBEM Tasarım Parametreleri Değerleri

Parametre	Değer	Açıklama
L1	32	Belleğin işlemciye bağlı portunun veri genişliği
L2	64	Belleğin ÖDBEM'e bağlı portunun veri genişiliği
K	16	Belleğin adres işaretinin veri genişliği
n	2	İşlem birimleri sayısı
N	1	İşlem birimleri sayısının ifade edilebileceği en küçük bit sayısı
m	1	Harici port sayısı
M	1	Harici port sayısının ifade edilebileceği en küçük bit sayısı
A	16	İşlem birimleri adres işaretinin veri genişliği
B	16	İşlem birimleri boyut işaretinin veri genişliği
T	16	Harici portlar paket boyutunun veri genişliği

ÖDBEM tasarımı Çizelge 5.1'de verilen tasarım parametreleri ile Xilinx Vivado aracı kullanılarak sentez ve gerçekleme aşamalarından geçirilmiştir. Gerçekleme sonucuna göre en yüksek saat frekansı 330 MHz. olarak belirlenmiştir. Şekil 5.1'de kaynak tüketimi ile ilgili sonuçlar verilmiştir.

Tasarım açık kaynak kodlu Openlane aracı kullanılarak da sentez ve gerçeklemesi yapılmış, kaynak tüketimi Şekil 5.2 ve alan bilgisi Şekil 5.3'de verilmiştir.

Resource	Utilization
LUT	1195
LUTRAM	35
FF	277

Şekil 5.1 : ÖDBEM tasarımı için kaynak tüketim değerleri.

```
=== mem_cpy_top ===

   Number of wires:                 4831
   Number of wire bits:             6816
   Number of public wires:           174
   Number of public wire bits:      2159
   Number of memories:                 0
   Number of memory bits:              0
   Number of processes:                0
   Number of cells:                 5480
     $_ANDNOT_                       948
     $_AND_                          156
     $_MUX_                         1983
     $_NAND_                          81
     $_NOR_                          198
     $_NOT_                          387
     $_ORNOT_                        132
     $_OR_                           924
     $_XNOR_                          27
     $_XOR_                          170
     sky130_fd_sc_hd__dfxtp_2        474
```

Şekil 5.2 : ÖDBEM tasarımı için Openlane kaynak tüketim değerleri.

```
=======================================
 report_design_area
=======================================
Design area 72812 u^2 8% utilization.
```

Şekil 5.3 : ÖDBEM tasarımı için Openlane alan değeri.

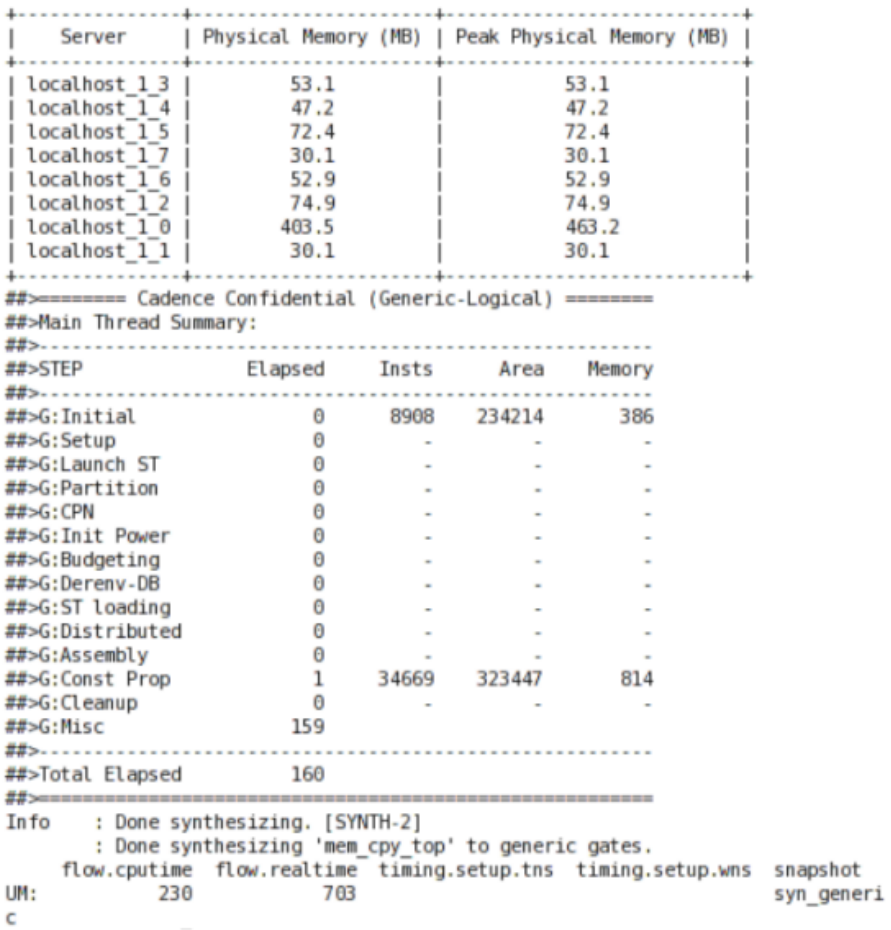
Tasarım Cadence Genus aracı ve TSMC 90nm kütüphanesi kullanılarak gerçeklenmiş ve alan tüketim değerleri Şekil 5.4'de verilmiştir.

5.2 Uygulama

Bu bölümde ÖDBEM tasarımını test etmek üzere yapılan bir donanım tasarımı ve donanım tasarımının çalışması için yapılan yazılım tasarımı anlatılacaktır.

5.2.1 Kırmık üstü sistem tasarımı

ODBEM tasarımını test etmek üzere işlemci, bellek, bir harici port ve bir işlem biriminden oluşan bir kırmık üstü sistem tasarımı yapılmıştır. Tasarım yapılırken Xilinx Vivado uygulaması kullanılmıştır. İşlemci olarak RISC-V mimarisinde bir tasarım olan Hornet çekirdeği kullanılmıştır. Bellek olarak Xilinx kütüphanesinde



```
+--------------+---------------------+--------------------------+
|    Server    | Physical Memory (MB) | Peak Physical Memory (MB) |
+--------------+---------------------+--------------------------+
| localhost_1_3 |         53.1        |           53.1           |
| localhost_1_4 |         47.2        |           47.2           |
| localhost_1_5 |         72.4        |           72.4           |
| localhost_1_7 |         30.1        |           30.1           |
| localhost_1_6 |         52.9        |           52.9           |
| localhost_1_2 |         74.9        |           74.9           |
| localhost_1_0 |        403.5        |           463.2          |
| localhost_1_1 |         30.1        |           30.1           |
+--------------+---------------------+--------------------------+
##>======== Cadence Confidential (Generic-Logical) ========
##>Main Thread Summary:
##>-----------------------------------------------------
##>STEP              Elapsed     Insts      Area    Memory
##>-----------------------------------------------------
##>G:Initial               0      8908    234214       386
##>G:Setup                 0         -         -         -
##>G:Launch ST             0         -         -         -
##>G:Partition             0         -         -         -
##>G:CPN                   0         -         -         -
##>G:Init Power            0         -         -         -
##>G:Budgeting             0         -         -         -
##>G:Derenv-DB             0         -         -         -
##>G:ST loading            0         -         -         -
##>G:Distributed           0         -         -         -
##>G:Assembly              0         -         -         -
##>G:Const Prop            1     34669    323447       814
##>G:Cleanup               0         -         -         -
##>G:Misc                159
##>-----------------------------------------------------
##>Total Elapsed         160
##>=====================================================
Info    : Done synthesizing. [SYNTH-2]
        : Done synthesizing 'mem_cpy_top' to generic gates.
     flow.cputime  flow.realtime  timing.setup.tns  timing.setup.wns  snapshot
UM:          230           703                                       syn_generi
c
```

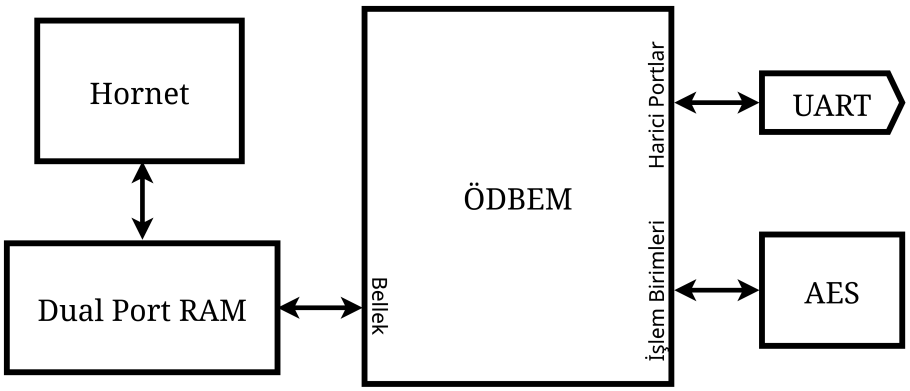
Şekil 5.4 : ÖDBEM tasarımı için Cadence Genus TSMC 90nm alan tüketim değerleri.

bulunan çift portlu bellek (Dual Port RAM) kullanılmıştır. Harici port için fiziksel port olarak UART seçilmiş ve UART için harici port yapısına uygun bir üst modül tasarımı yapılmıştır. İşlem birimi için AES algoritması seçilmiş ve işlem birimi yapısına uygun olacak şekilde bir üst modül tasarımı yapılmıştır. ÖDBEM tasarımı test etmek üzere yapılan blok tasarım Şekil 5.5 de verilmiştir.

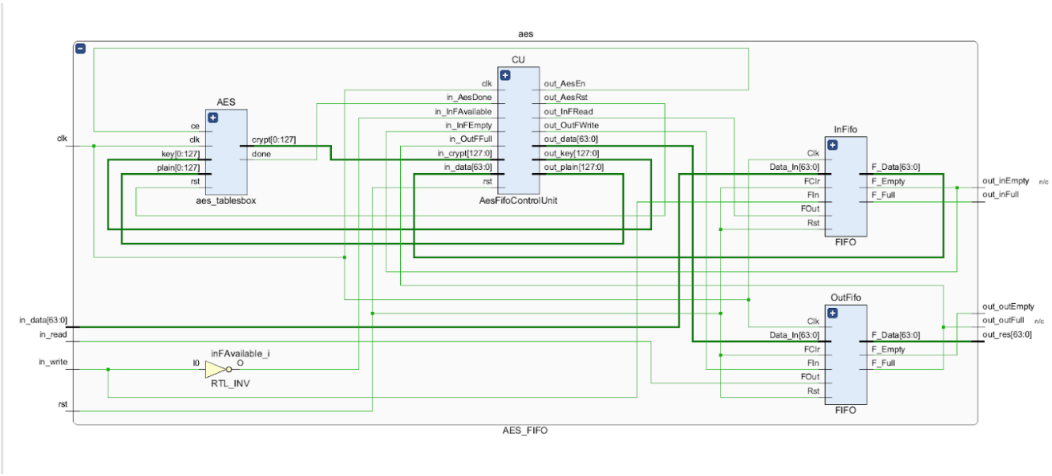
Şekil 5.5 de verilen blok tasarımın gerçekleştirilebilmesi için UART ve AES bloklarına ÖDBEM ile bağlantı yapılabilecek şekilde bir arayüz eklenmesi gerekir. AES için bölüm 4.2de verildiği şekilde bir tasarım yapılmıştır. Tasarıma ait elde edilen RTL şeması Şekil 5.6de verilmiştir.

Benzer şekilde UART için Bölüm 4.3 de verilen yapıya uygun bir üst modül tasarımı yapılmış ve rtl şeması Şekil 5.7de verilmiştir.

Xilinx Vivado kullanılarak Şekil 5.5de verilen tasarım için donanım tanımlama dilleri ile kod yazılmış ve RTL şeması oluşturulmuştur. RTL şeması 5.8de verilmiştir.



Şekil 5.5 : ÖDBEM tasarımını test etmek için oluşturulan blok tasarım.

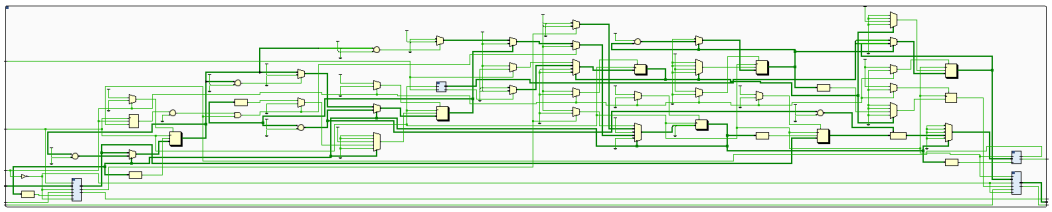


Şekil 5.6 : AES işlem birimine ait rtl şeması.

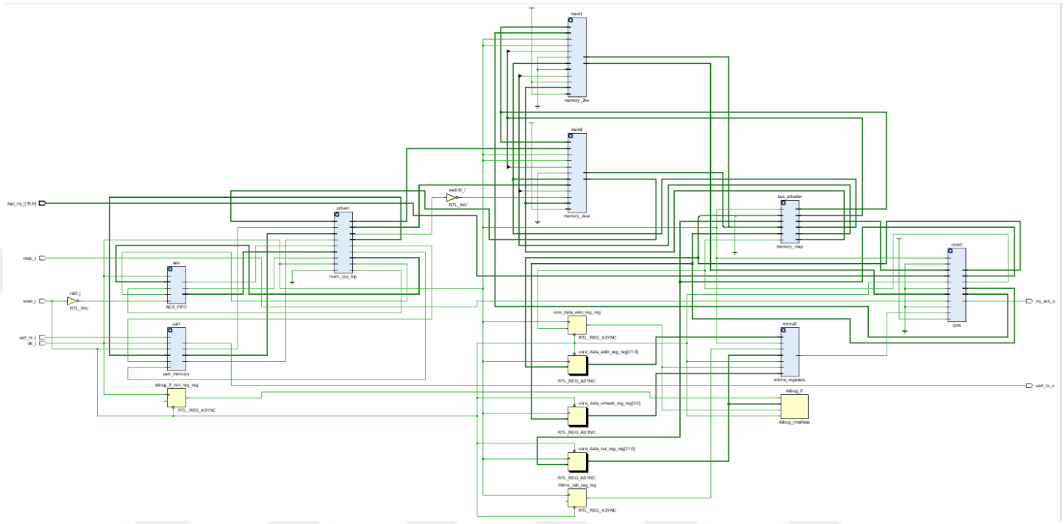
5.2.2 Yazılım tasarımı

Bu bölümde ÖDBEM tasarımını işlemci tarafından yönetebilmek için yazılan sürücü kodları ve ÖDBEM'i test etmek için oluşturulan sistemde çalışacak bir uyglumanın kodları verilmiştir. ÖDBEM tasarımının çalışabilmesi için gerekli sürücü kodlar Şekil 5.9 ve Şekil 5.10 verilmiştir.

ÖDBEM sürücü kodları kullanılarak oluşturulan bir test uygulaması Şekil 5.11'de verilmiştir. Bu uygulamada kırmık üstü sistemin dışından AES kullanılarak şifrelenmesi için bir veri ve şifrelemede kullanılacak bir anahtar gönderilmesi ve şifreli metnin kırmık üstü sistemden okunması planlanmıştır. Verinin gönderilmesi ve okunması için UART kullanılacaktır.



Şekil 5.7 : UART harici portuna ait rtl şeması.



Şekil 5.8 : ÖDBEM tasarımını test etmek için oluşturulan blok tasarımın rtl şeması.

5.2.3 Davranışsal benzetim

Benzetim için C yazılım dili ve UART kullanarak SoC dışına veri gönderilmesi test edildi. Yazılan test kodunda ilk önce şifrelenecek veri ve anahtar UART'a yazılarak kırmık üstü sisteme iletildi. Yazılan uygulama doğrultusunda şifrelenen veri UART'dan başarılı şekilde okundu. Davranışsal benzetim sonucu Şekil 5.12'de verilmiştir. Sistemin doğru çalıştığı gözlemlenmiştir.

5.2.4 Sentez ve gerçekleme

Davranışsal benzetim sonrası, davranışın beklendiği gibi olması sonucunda sentez ve gerçekleme yapıldı. Gerçekleme için öncelikle FPGA olarak Nexys A7-100T seçildi. Sentezlenme sonucunda kullanım raporu Şekil 5.13'de verilmiştir.

Sentez sonrasında gerçekleme yapılmış ve gerçekleme sonucunda kaynak kullanımı aşağıda Şekil 5.14'de verilmiştir. Görüldüğü üzere gerçekleme sonrasında LUT kullanımı azaldı.

```
// ÖDBEM Sürücü Kodları
// HP: Harici Portlar
// IB: İşlem Birimleri
#define ISLEM_KODU      (*(volatile uint32_t *)0x00000200)
#define KAYNAK          (*(volatile uint32_t *)0x00000204)
#define HEDEF           (*(volatile uint32_t *)0x00000208)
#define BOYUT           (*(volatile uint32_t *)0x0000020C)
#define IB_DURUM        (*(volatile uint64_t *)0x00000210)
#define IB_SIFIRLAMA    (*(volatile uint64_t *)0x00000210)
#define HP_DURUM        (*(volatile uint32_t *)0x00000220)

// İşlem Kodları
// GONDER_{IB,HP}_{MEMory,HP}_{Düz,Ters}_{ARtarak,AZalarak}
#define GONDER_IB_MEM_D_AR  0x00000001
#define GONDER_IB_MEM_D_AZ  0x00000081
#define GONDER_IB_MEM_T_AR  0x00000041
#define GONDER_IB_MEM_T_AZ  0x000000C1
#define GONDER_IB_HP_D_AR   0x00000021
#define GONDER_IB_HP_D_AZ   0x000000A1
#define GONDER_IB_HP_T_AR   0x00000061
#define GONDER_IB_HP_T_AZ   0x000000E1
#define GONDER_HP           0x00000011
//KOPYALA
#define KOPYALA 0x00000003
//AL_{Oncelikli Kodlayıcı,KAYNAK}
#define AL_OK       0x00000015
#define AL_KAYNAK   0x00000015
//SIFIRLA
#define SIFIRLA     0x00000007
```

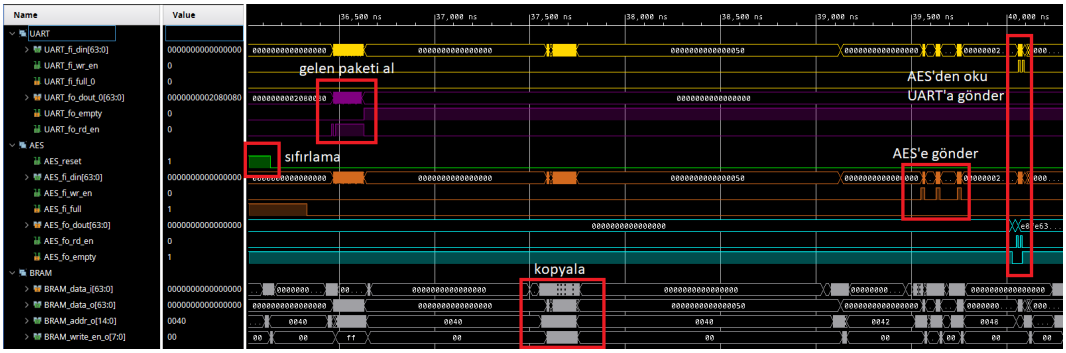
Şekil 5.9 : ÖDBEM sürücü kodlarında gerekli tanımlamalar.

```
// send data to cores: bram -> cores
void odbem_gonder(uint32_t hedef, uint64_t *kaynak, uint32_t boyut,
                                  uint32_t islem_kodu)
{
    HEDEF = hedef;
    KAYNAK = (uint32_t)kaynak;
    BOYUT = boyut;
    ISLEM_KODU = islem_kodu;
    while (ISLEM_KODU != 0);
}

void odbem_kopyala(uint64_t *hedef, const uint64_t *kaynak, uint32_t boyut)
{
    HEDEF = (uint32_t)hedef;
    KAYNAK = (uint32_t)kaynak;
    BOYUT = boyut;
    ISLEM_KODU = KOPYALA;
    while (ISLEM_KODU != 0);
}

void odbem_al(uint64_t *hedef, uint32_t kaynak, uint32_t islem_kodu)
{
    HP_DURUM = 0;
    HEDEF = (uint32_t)hedef;
    KAYNAK = kaynak;
    ISLEM_KODU = islem_kodu;
    while (ISLEM_KODU != 0);
}


void odbem_sıfırla(uint64_t sıfırla_deger)
{
    MEMCPY_REG_PE_RESET = sıfırla_deger;
    ISLEM_KODU = SIFIRLA;
    while (ISLEM_KODU != 0);
}
```

Şekil 5.10 : ÖDBEM sürücü kodlarında gerekli fonksiyonlar.

```c
int main(void){

    //AES çekirdeği sıfırlama girişini 1 yap.
    odbem_sıfırla(0x0000000000000001);
    // UART'dan veri gelene dek bekle
    while(HP_DURUM = 0);
    // Gelen paketi kaydetmek için oluşturulan adres.
    uint64_t gelen_paket;
    // Gelen paketi gelen_paket adresinden başlayarak yaz.
    odbem_al(gelen_paket, 0, AL_OK);
    // AES Sıfırlama girişini 0 yap.
    odbem_sıfırla(0x0000000000000000);
    // AES'e gönderilecek paketi oluştur.
    uint64_t aes_paketi[5];
    // AES paketinin ilk kelimesine sonuç adresini ve boyutunu yaz.
    // Sonucu doğrudan UART'a göndereceğimiz için MSB 32 bitlik kelime
    // UART'ın bağlantı numarası yani '0'
    // Çıkış Boyutumuz ise 128 bit olacak yani 16 bayt.
    aes_paketi[0] = 0x0000000000000010;
    //Gelen paketteki anahtar ve şifrelenecek veriyi aes paketine kopyalıyoruz
    odbem_kopyala(aes_paketi[1], gelen_paket[1], 4*8);
    //ODBEM gönder dediğimizde AES'e oluşturulan paket ODBEM tarafından
    //gönderilecek ve şifrelenmiş metin AES'den okunup doğrudan UART'a gönderilecek.
    odbem_gonder(0,aes_paketi,5*8,GONDER_IB_HP_D_AR);
    //işlem bitene kadar bekle.
    while(IB_DURUM = 0);
}
```

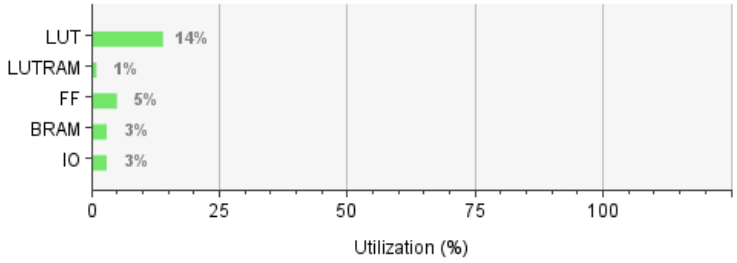
Şekil 5.11 : ÖDBEM tasarımını test etmek için yazılan uygulama kodu.



Şekil 5.12 : Davranışsal benzetim sonucu.

Summary

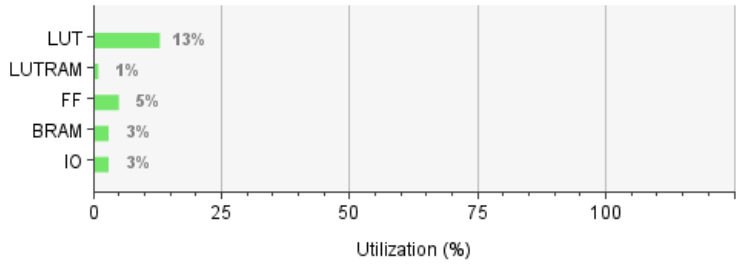
Resource	Utilization	Available	Utilization %
LUT	8640	63400	13.63
LUTRAM	176	19000	0.93
FF	5825	126800	4.59
BRAM	4	135	2.96
IO	7	210	3.33



Şekil 5.13 : Sentez sonucu kaynak kullanımı.

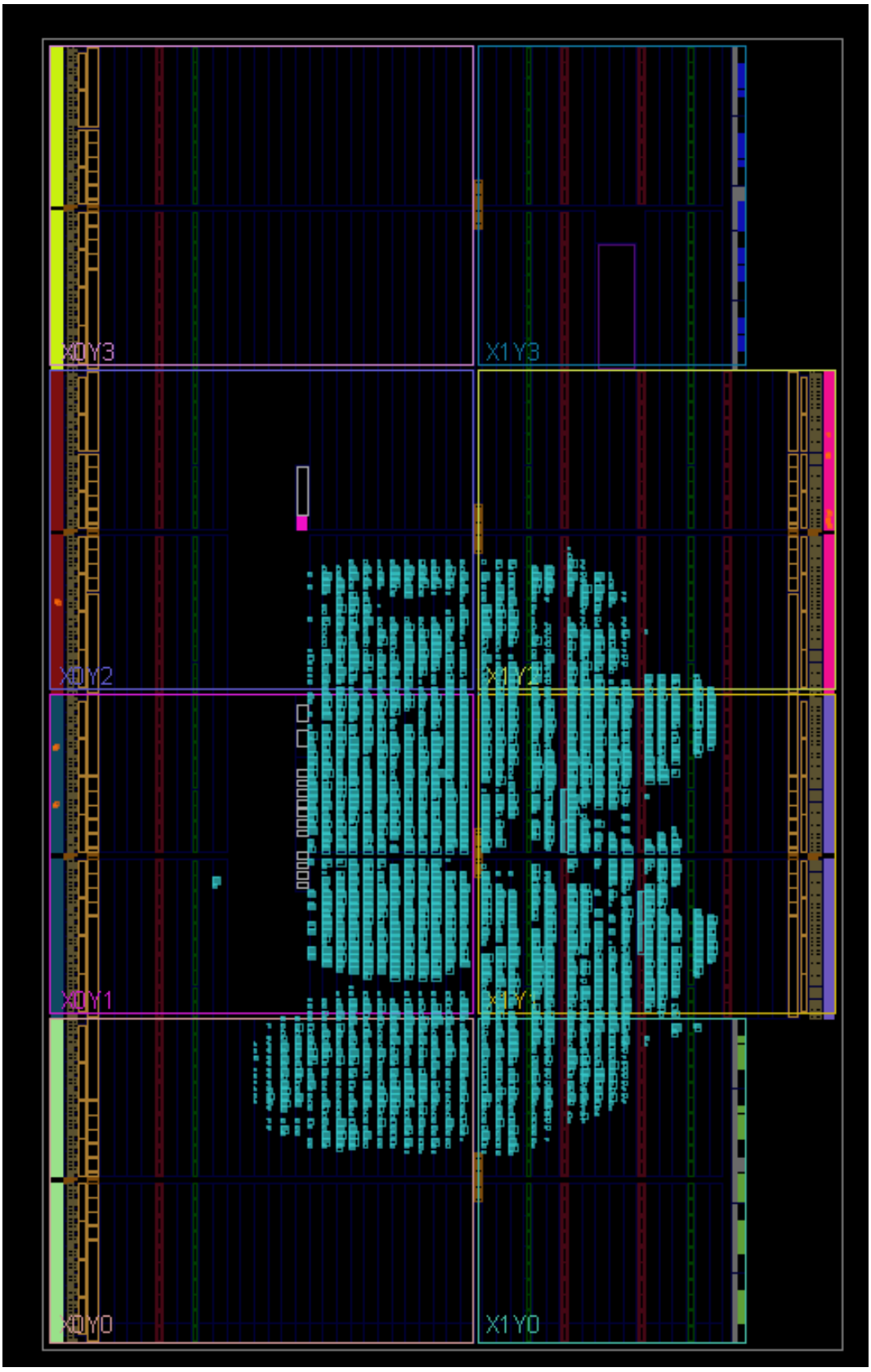
Summary

Resource	Utilization	Available	Utilization %
LUT	8479	63400	13.37
LUTRAM	176	19000	0.93
FF	5825	126800	4.59
BRAM	4	135	2.96
IO	7	210	3.33



Şekil 5.14 : Gerçekleme sonucu kaynak kullanımı.

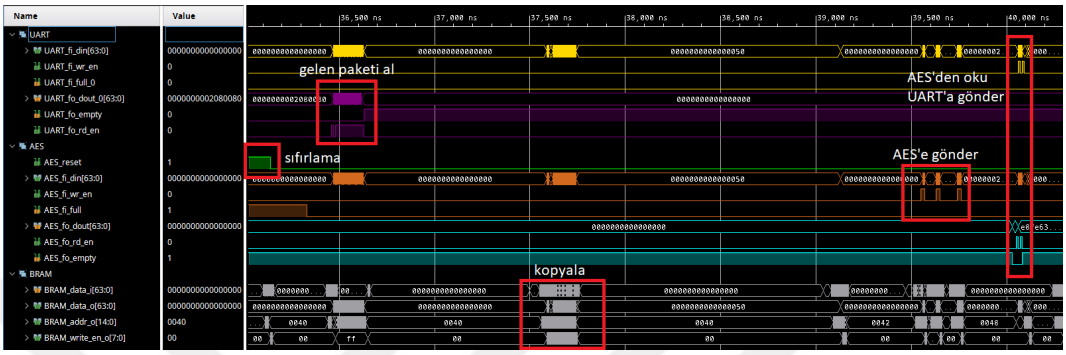
Gerçeklemenin cihaz üzerindeki yerleşimi sınırlanmadığı için yerleşim gerçekleyiciye bırakıldı. Gerçekleme sonucunda tasarımın cihaz üzerindeki görünümü Şekil 5.15'de verilmiştir.



Şekil 5.15 : Tasarımın cihaz üzerinde görünümü.

5.2.5 Sentez sonrası benzetim

Sentez sonrasında aynı test dosyası kullanılarak test yapıldı. Testin doğru sonuçlandığı görüldü. Sentez sonrası test sonucu Şekil 5.16'de verilmiştir.



Şekil 5.16 : Sentez sonrası benzetim sonucu.

6. SONUÇLAR

Bu çalışma kapsamında literatürde bulunan DMA ile ilgili makale ve patentler incelenmiş, işlemci ile doğrudan bağlantı gerektirmeyen, işlemci tercihinde bir kısıt oluşturmayan, işlemci üzerinde herhangi bir yapısal değişiklik gerektirmeyen, DMA komutlarının bellek üzerinden iletildiği bir tasarıma rastlanmamıştır. Bu sebeple bu çalışma özgün bir tasarım içermektedir.

Çalışma kapsamında bir özel doğrudan bellek erişim modülü tasarımı yapılmış, tasarım detayları paylaşılmıştır. Tasarlanan ÖDBEM modülü işlemciden aldığı komutlarla çalışan bir doğrudan bellek erişim modülüdür. ÖDBEM komutlarının iletilmesi için işlemci ve ÖDBEM arasında doğrudan bir bağlantı olmasına gerek yoktur. Komutlar bellek üzerinde kırmık üstü sistemin tasarımı sırasında belirlenen bellek adresleri üzerinden iletilir.

ÖDBEM, işlemci, bellek, işlem birimleri ve harici portlardan oluşan bir kırmık üstü sistemde ÖDBEM kullanılarak, harici portlardan okunan veri belleğe yazılabilir, işlem birimlerinde bir işlem başlatılabilir ve sonucu belleğe veya doğrudan bir harici porta yazılabilir, işlem birimlerinin sıfırlama (reset) girişleri kontrol edilebilir, belleğin bir adresinden başka bir adresine kopyalama yapılabilir, bellekteki veri harici portlara yazılabilir.

Yapılan ÖDBEM tasarımı ölçeklenebilir ve geliştirilebilir bir yapıdadır. İşlem birimlerinin sayısı, harici portların sayısı ve bellek arayüzünün veri genişliği değiştirilebilir. Komut yapısı oluşturulurken ve algoritmik durum makineleri tasarlanırken geliştirmeye uygun ve sürdürülebilir bir tasarım hedeflenmiştir.

İşlem birimleri ve harici portların tanımlamaları yapılmış ve ÖDBEM ile haberleşmeleri için bir üst modül tasarımı verilmiştir. İşlem birimleri ve harici portların veri transferlerinde FIFO yapısı temel alınmış ve arayüz olarak FIFO'dan veri okuma ve FIFO'ya veri yazmak için gerekli en basit sinyaller seçilmiştir. Dolayısıyla

Custom IP'ler ve fiziksel portlar için eklenecek ÖDBEM arayüzünün uygulaması kolaylaştırılmıştır.

Yapılan ÖDBEM tasarımı için donanım tanımlama dilleri ile kod yazılmış, kod yazılırken olabildiğince ölçeklenebilir bir yapı kurulmuştur. ÖDBEM'in işlemci tarafından yönetilebilmesi için sürücü kodları yazılmıştır. ÖDBEM, işlemci, bellek, bir işlem birimi ve bir harici porttan oluşan bir kırmık üstü sistem tasarımı yapılmış ve bu tasarımın test edilmesi için işlemci üzerinde çalışacak bir test uygulaması yazılmıştır. Test uygulaması davranışsal benzetimde kullanılmış ÖDBEM'in ve kırmık üstü sistemin doğru çalıştığı gözlemlenmiştir. Davranışsal olarak doğru çalışan sistemin sentez ve gerçeklemesi yapılmış, kaynak kullanımına ilişkin bilgiler verilmiştir. Davranışsal benzetim için yazılan test uygulaması sentez sonrası benzetimde kullanılmış, ÖDBEM'in ve kırmık üstü sistemin doğru çalıştığı gözlemlenmiştir.

KAYNAKLAR

[1] **Patterson, D. ve Hennessy, J.** (2017). Computer Organization and Design RISC-V Edition: The Hardware Software Interface, bölüm 6, Elsevier Science, s.1054.

[2] **Wolf, W.** (2002). Modern VLSI Design: System-on-Chip Design, bölüm 1, Pearson Education, s. 3.

[3] **Stallings, W.** (2003). Computer Organization and Architecture: Designing for Performance, bölüm 7, Pearson Education, s.236–242.

[4] **RISC-V Organizaton**. *https://riscv.org/*.

[5] **Mano, M.** (2002). Digital Design, bölüm 7, Prentice Hall, s.299–315.

[6] **Taraate, V.** (2022). Digital Logic Design Using Verilog: Coding and RTL Synthesis, bölüm 22, Springer Singapore, s.511–534.

[7] **Morales, H.**, **Duran, C. ve Roa, E.** (2019). A Low-Area Direct Memory Access Controller Architecture for a RISC-V Based Low-Power Microcontroller, *2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS)*, s.97–100.

[8] **Shirur, Y.J.M.**, **Sharma, K.M. ve A, A.** (2018). Design and implementation of Efficient Direct Memory Access (DMA) Controller in Multiprocessor SoC, *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, s.1–6.

[9] **Chi, Y. ve Zheng, Z.** (2018). A Design of Direct Memory Access Controller for Wireless Communication SoC in Power Grid, *2018 IEEE 2nd International Conference on Circuits, System and Simulation (ICCSS)*, s.76–79.

[10] **Henson, M. ve Baker, D.C.** (2008). *Systems and Methods for Direct Memory Access*, United States Patent, No: 7376762 dated 20.5.2008.

[11] **Rader, S.M.**, **Garani, P.**, **Steininger, F. ve Lucas, B.G.** (2007). *Memory Access System Including Support for Multiple Bus Widths*, United States Patent, No: 7281066 dated 9.10.2007.

[12] **Pham, C.H. ve Konda, D.R.** (2016). *System and Methods for Managing Direct Memory Access Operations*, United States Patent, No: 9229893 dated 5.1.2016.

[13] **Curtis, K.E.** (2021). *Programmable Arbitrary Sequence Direct Memory Access Controller for Configuring Multiple Core Independent Peripherals*, United States Patent, No: 10983936 dated 20.4.2021.

[14] **Dorst, J.R. ve Liu, X.** (2014). *Direct Memory Access Controller with Hybrid Scatter-Gather Functionality*, United States Patent, No: 0317333 dated 23.10.2014.

[15] **Rajbharti, N.** (2008). *Direct Memory Access Controller*, United States Patent, No: 0126662 dated 29.5.2008.

[16] **Triece, J.W.**, **Pesavento, R.J.**, **Lathi, G.D. ve Dawson, S.** (2018). *Direct Memory Access Controller*, United States Patent, No: 9921985 dated 20.3.2018.

ÖZGEÇMİŞ

Adı SOYADI: Mustafa Mert ESEN

ÖĞRENİM DURUMU:

- **Lisans:** 2019, İstanbul Teknik Üniversitesi, Elektrik Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisligi Bölümü

MESLEKİ DENEYİMLER VE ÖDÜLLER:

- 2019 yılından beri PROCENNE şirketinde sayısal tasarım mühendisi olarak çalışıyor.