Chapter 4

THE PHILIPS NEXPERIA DIGITAL VIDEO PLATFORM

J. Augusto de Oliveira and Hans van Antwerpen
Philips

Abstract: This chapter will outline the challenges in platform development for digital consumer home devices delivering multimedia content and novel services and applications. It will detail the requirements on this market and the approaches taken by Philips Semiconductors when developing the Nexperia-Digital Video Platform. The Nexperia-Digital Video Platform (Nexperia-DVP) comprises a family of Systems on a Chip (SoCs) and a software platform that allows Philips' customers to build cost effective, flexible Digital Video appliances.

Key words: Platform, Nexperia, SoC, video, digital video platform, consumer electronics

1. NEXPERIA DIGITAL VIDEO PLATFORM

1.1 The Digital Video Revolution

The transition from Analog to Digital Video is transforming the way we enjoy home entertainment: in addition to the higher quality of the video and audio programs, we are also experiencing novel ways to navigate, store, retrieve and share the digital programs as well as access to new interactive services and connectivity possibilities.

Our home entertainment systems will be implemented with a number of Digital Video appliances, such as Digital Televisions (DTVs), DVD Players, Digital Video Recorders and Set-top Boxes. These home entertainment systems will connect to each other and to the productivity cluster around the PC and to mobile devices like cellphones and automobiles via wired/wireless networks and/or removable optical and solid state memories.

In comparison with their PAL and NTSC analog predecessors, the new Digital Video appliances, in addition to being able to decode program
streams in a compressed digital format, will also include the computing power to navigate and process the digital stream. For example, a DVD Player, besides decoding the digital video, also provides the user with a simple yet sophisticated navigation system; and DTVs export Electronic Program Guides, greatly helping the consumer with the navigation of the live programming and the possible time-shifted recorded content.

Philips Semiconductors is developing a range of solutions for digital video appliances [1], based on a vision of this technology that is both imaginative and firmly rooted in real needs. The Digital Video vision must deliver valuable practical benefit to consumers. Digital Video appliances will provide consumers with access to and interaction with a powerful, coherent home network via an easy-to-use interface, which allows them to concentrate on what they want to do: to access their content-by-choice any time, any place with their device of choice. Users will focus on activities and needs, simply using the most convenient appliance, whether it's a DTV, a PC screen or any other part of the network.

1.2 The Philips Nexperia Platform Approach

Philips Semiconductors decided to serve the application domains of digital video and mobile with a platform approach that we call Nexperia [2]. The motivations for the Nexperia platform approach can be clustered around two main points: rising product complexity, both in silicon and software; and commonality of functions for the Digital Video scope, encompassing the following trends:

- Continued demand from end users for products that are simple to use but implement new applications and services, often leading to internal product complexity. Here we distinguish the internal complexity of the product, which makes new features possible, from the external complexity [3] that determines if the product is easy or difficult to use.
- Relentless increase in the number of manufacturable gates per wafer
- Relentless increase in the size of code and data storage space, encompassing solid-state, magnetic and optical storage
- Connectivity and convergence, allowing for interoperability and common functions among products from the previously separated domains of communications, computing and consumer worlds
- Expectations from customers that certain popular functions will probably spread across and become ubiquitous among product categories, such as MP3 audio and JPEG images.
- The possibilities created by Digital Video content demand an increasing flexibility of consumer appliances. The required set of applications and
their formats vary over time as standards evolve as well as per product and per country.

Consumer electronic manufacturers are expecting more from their silicon suppliers because of these trends: from flexible SoC platforms to complete solutions. This is inevitable in the age of SoCs, since so much of the system design and software/hardware partition is now done by the SoC semiconductor supplier. Therefore the industry is experiencing a re-aggregation of the design chain.

Nexperia embodies Philips Semiconductors’ vision that, in order to cope with these trends, these application domains are better served by families of flexible multimedia SoC solutions that we call Nexperia platforms.

Nexperia platforms serve as the foundation for our customers [4] to build their novel products and services, creating the ‘next experience’ multimedia products that the consumers want. The Nexperia platform approach embodies the following properties:

- Flexibility (through programmability and extensive software & hardware IP choice) for easy differentiation and product upgradeability
- Innovation – addressing new, exciting, consumer applications
- Future-proof via software upgrade ability and a roadmap of compatible platform instances
- The use of a architecture framework and IP blocks to flesh out designs

2. NEXPERIA-DVP PLATFORM CONCEPTS

2.1 Reference Architecture

Nexperia-DVP is a Reference Architecture, i.e. a set of documents that describes how the products of the Digital Video Product Family will be partitioned into subsystems and how functionality will be split over these subsystems. We normally describe Nexperia-DVP by documenting its three main parts:

a) Nexperia-DVP SoC Reference Architecture
b) Nexperia-DVP Software Reference Architecture
c) Nexperia-DVP System Reference Architecture

2.2 Standard Designs

The common subsystems, or building blocks, of the Reference Architecture are called Standard Designs. They can be a VLSI component
(also known as device IP), a Software component, or a number of VLSI/Software components implementing a certain function.

The development of a Standard Design that implements a particular function can and is normally organized as a project, performed by a multi-disciplinary project team. VLSI and Software developers work together with Product Management to specify, develop, test and document a new standard design.

2.3 Product Platforms

When all the subsystems have been realized as Standard Designs, they can be integrated to check that the subsystems work together and implement the functions required. This set of realized, pre-integrated subsystems is called the Product Platform. A Nexperia-DVP Product Platform complies with the Nexperia-DVP Reference Architecture. The difference between both is that the Reference Architecture is just a document, while the Product Platform is a set of realized and integrated components.

Nexperia-DVP has been developed using the Carrier Product concept. Carrier Products are developed in response to fast evolving customer demands or technology opportunities and positioned in markets that are as yet fairly unknown. They are planned as first-of-a-kind products and defined as a carrier for future Product Platforms. Engineering effort is necessary to upgrade/document the product architecture to a Reference Architecture and to rework subsystems that should become reusable assets in future projects, in order to bootstrap the Product Platform approach. Examples of Nexperia-DVP Carrier products are the PNX-8525 [5] and the PNX-8550 (a.k.a. Viper 1.0 and 2.0 respectively). This cycle is illustrated in Figure 4-1 below.

![Figure 4-1. Nexperia Product Development Cycle](image)
We expect that future developments of Nexperia-DVP will move from Carrier Products to the realization of Product Platforms. The new design content of a platform-based product is lowered due to the reuse of existing Standard Designs together with product-specific designs. Throughput time and risks are reduced. Platform based product realization requires that Product Platforms have been planned, created, tested and have reached a sufficient degree of maturity by the start of product realization.

2.4 Products

Nexperia-DVP Products serve on their turn as a platform to the consumer electronics manufacturers, or the foundation they use to create their own unique and novel products and services.

Nexperia-DVP products are developed rapidly and efficiently on the basis of a Product Platform, characterized by a Reference Architecture and a number of Standard Designs purchased or developed in-house.

Since Nexperia-DVP products are derived from a Nexperia-DVP Product Platform, they all share properties that are valuable to our customers like flexibility through programmability, cost effectiveness, and a guarantee of being future-proof via software upgradability and a roadmap of compatible products.

3. DESIGNING NEXPERIA-DIGITAL VIDEO SOCS

3.1 Nexperia-DVP SoC Reference Architecture

Figure 4-2 illustrates the Nexperia-DVP Reference Architecture from a SoC Connection Network point of view. The following main elements are identified:

a) Processor Cores:
We have selected MIPS CPUs [6] to be the main control processor for Nexperia-DVP. Philips has designed and licensed a complete range of MIPS CPUs compatible with MIPS32 or MIPS64 architectures. We have also selected TriMedia [7, 8] as the main Media DSP architecture, allowing the flexible implementation of many video and audio algorithms in software. The use of MIPS as a “brancher” and TriMedia + hardware as “streamers” capitalizes on the roles that best suit each architecture (RISC vs VLSIw DSP) for a total system aggregated computational performance that can be only matched by silicon many times more expensive. Nexperia-DVP also allows for MIPS-only and TriMedia-only
Products, depending on specific product family needs for media processing in software and advanced control processing.

b) Device IP Blocks:
The Device IP blocks implement in hardware (or micro coded HW) the necessary interfacing and processing functions for video & audio (input, output, scaling, encoding, decoding, transcoding), networking & connectivity (IEEE1394, USB, Ethernet), I/O (PCI/XIO, parallel, I2C), and architecture support (interrupt controller, semaphores, clock control).

c) Connection Network
The connection network binds all the traffic. The diagram illustrates two main networks: the Device Control & Status Network, used to read/write to the Device IP registers, and the Pipelined Memory Access Network, used by the CPUs and Device IPs to access the main memory.

d) Main Memory Interface (MMI):
Connects the Nexperia-DVP SoC to the main memory, typically a high-speed DRAM memory.

![Nexperia-DVP System on Silicon Diagram]

Figure 4-2. Nexperia-DVP Reference Architecture
3.2 Three Levels of Abstraction

We have defined the Nexperia-DVP SoC Reference Architecture using three levels of abstraction. These levels were used in order to avoid traditional architecture definitions that do not cope with evolution because the Device IP library is tied to a particular bus selection. The three Levels of Abstraction are:
- Level 1: Software-Hardware Platform Rules
- Level 2: Device Transaction Level
- Level 3: Connection Network

3.2.1 Level 1: DVP Software-Hardware Rules

This level deals with the software view of the hardware. The “DVP Software-Hardware Rules” defines:
- Unified Memory Architecture
- Rules for Data Movement
- Endianess
- Ordering & Coherency
- Interrupts
- Data formats, including Pixel Formats.
- Trimedia-MIPS Communication
- Protection
- Boot

The Unified Memory Architecture specifies that all addressable objects in the system should be identified by the same address. We have defined the two main types of data movement: Device Control & Status (DCS) transactions that access 32-bit only registers in the Device IP and must have no side effects on read, and Direct Memory Access (DMA) transactions that move data to/from memory to the Device IP.

3.2.2 Level 2: Device Transaction Level (DTL)

Level 2 deals with point-to-point transfers between Device IPs and the Connection Network, specifying a Device IP partition and architecture that is compatible with Level 1. Two main types of ports are typical of Device IP:
- Device Control & Status Ports: these are read/write to memory mapped control and status registers in the Device IP
- DMA ports: the Device IP uses these ports to communicate to memory. It also allows for direct Device-IP-to-IP communication.
With the introduction of DTL ports and protocol to our Device IP library, we managed to:

- Allow the reuse of the Device IP even when the Connection Network evolves.
- Remove details of the bus protocol from IP development, letting the IP move data in the most natural way for that IP.
- Move any clock domain boundary out of IP and into the bus interface.

The Nexperia-DVP concept of DTL ports and DTL protocols precedes and resembles VSIA’s VCI interface [9], but it has an important conceptual difference: While the VCI interface was meant to just abstract the specific type of bus utilized in the systems from the IP, with DTL we abstract the system aspects (memory bandwidth, optimal transaction lengths for the memory and the systems, amount of buffering needed, etc.) from the Device IP. Figure 4-3 shows an example of our Device IP architecture partition.

![Diagram of Device IP Architecture Partition](image)

**Figure 4-3. Device IP Architecture Partition**

### 3.2.3 Level 3: Connection Network

The Nexperia-DVP Level 3 Connection Network deals with the more traditional Bus Hierarchy & Bus Level: it specifies a bus hierarchy and for each bus their wires, clock cycle protocols, AC characteristics, hardware architecture & hardware blocks.

The second generation Nexperia-DVP is composed of:

- The Device Control & Status (DCS) Network
- The Memory Connection Network
3.2.3.1 Device Control & Status (DCS) Network

From a system/SoC architecture point of view, the DCS Network is primarily a low latency communication path for the CPUs and other initiators to access the control & status register in the Device IPs. From a VLSI physical architecture point of view, the DCS Network allows for the implementation of a Physical Design Strategy of “islands of synchronicity” that we will describe later in this section. A DCS Network has the following properties:

- 8, 16 and 32 bit transactions
- Time-out generation and multiple bus system design
- Sampling of bus signals in case of error or time-out generation
- Both posted and precise writes
- Low power design
- Protection with selective blocking of initiators access to targets
- Signals and protocol very similar to DTL, allowing for simple and efficient adapter design
- Compatible with “chiplet” (or island) physical design approach
  - Synchronous and asynchronous interconnect options
  - Efficient layout with few top level wires and minimum netlist partitioning requirements
  - Fast timing closure

Figure 4-4 shows the logical view of a DCS network. Typically each Device IP has a DTL-based DCS port that is hooked to the DCS Network via a DTL-DCS adapter. Device IPs are designed such that DTL DCS ports can operate at the IP Clock (from 70 to 150 MHz) or at the DCS Network clock (currently at 200 MHz).

A DCS Network Controller is configured to each initiator/target characteristics, such as the clocking mechanism (synchronous or asynchronous), and the type of register write allowed (posted or non-posted writes). The DCSN Controller implements round-robin arbitration between initiators, the splitting of command to targets (address decode), the collection and multiplexing of response from targets, the timeout generation, error handling (captures error transaction), and protection.
3.2.3.2 Memory Connection Network

Nexperia-DVP SoCs Memory Connection Networks are expected to evolve over time, and we will automate the generation of such structures. The current generation is implemented with two key elements: Memory Transaction Level (MTL) ports and protocol and the Pipelined Memory Access Network, that are both described below.

MTL is a communication protocol optimized for communication to a (DRAM) memory controller. Figure 4-5 shows the application of MTL ports in a typical Nexperia-DVP SoC. CPUs as well as DMA agents communicate to memory using MTL. MTL is a point-to-point interface protocol, which means that CPUs typically have their own private MTL connection to the memory controller. In the DMA memory infrastructure MTL is used to connect DMA adapters to the connection network and to interface the connection network to the memory controller. In mid-range/low-cost applications there might be one DMA adapter that interfaces directly with MTL to the controller. DTL-MTL Adapters are required to translate Device IP style transaction to system/memory optimized transactions.

We have implemented the current generation of our Memory Connection Network via PMAN (Pipelined Memory Access Network). From a system/SoC architect point of view, the PMAN is primarily a high bandwidth hierarchical communication path for the Device IPs to communicate to memory. From a VLSI physical architect point of view, the PMAN allows for the implementation of a Physical Design Strategy of
"islands of synchronicity" and hierarchical split that we will describe later in this session.

![Diagram](image)

*Figure 4-5. Application of MTL Ports in a typical Nesperia SoC*

PMAN allows the implementation of a mixture of deferred/local arbitration; non-symmetrical hierarchies for write and control data muxing; MTL to MTL routing; memory protection via the partition of memory into "sand boxes". Each MTL initiator is allowed or denied access to each sand box. Current PMAN implementation is clocked at 250 MHz and allows a maximum data rate of 1 Gbytes/second.

### 3.3 SoC Implementation

As we have seen, the Nesperia-DVP SoC Architecture strives for the orthogonalization of the Communication Architecture versus Device IP Functional considerations. This approach and the characteristics of the protocols selected are with respect to the SoC implementation:
- Physical Hierarchy Friendly
- Timing Closure Friendly
- Handoff Friendly
- Change Friendly

This allowed for a SoC Physical Implementation that is characterized by:
a) Physical hierarchical partition in "chiplets"
b) Islands of synchronicity

Partitioning divides the logic hierarchy into manageable sized blocks, called chiplets. Modules (processors, IP devices and the Communication networks) are divided over the chiplets to create Islands of Synchronicity within the chiplets. Inter-chiplet communication is either asynchronous (DCS), or flop-to-flop (PMAN).

![Diagram of logical to chiplet mapping](image_url)

Figure 4-6. Logical to Chiplet Mapping

Closely related to top-level floorplanning, partitioning follows these guidelines:
- The logical hierarchy, characterized by the DCS and PMAN network dataflow in the design
- Size and complexity of modules
- Clock domains within modules; clock domains (except PMAN clock) are within one chiplet, making it locally synchronous.
- Chip IO Timing requirements of the modules

Signals connect between the chiplets via abutment. There is no top-level routing, except for the clocks that had to be matched. The inter-chiplet signals between non-neighboring chiplets are routed by inserting feed-through buffers in the chiplets that the signal had to route through. Figures 4-6 and 4-7 show the logical to chiplet mapping and the resulting floorplan for the PNX-8550. A more detailed description of the implementation of our first generation of a Nexperia-DVP product is described in [10] and [11].
Figure 4-7. Floorplan for the PNX-8550

4. NEXPERIA-DVP SOFTWARE ARCHITECTURE

In this section we will cover a brief overview of the Nexperia-DVP software architecture. A thorough discussion of the entire software architecture for Nexperia-DVP products is beyond the scope of this chapter, so we will introduce only those concepts and parts that are most relevant to hardware/software co-design challenges that will be covered in the next section.
4.1 Nexperia-DVP Platform Software

At the top-level, a Nexperia-DVP product typically consists of three large subsystems: the operating system, the Nexperia-DVP platform software and the customer’s middleware and applications.

- **Operating System**: In principle the Nexperia-DVP architecture can function with any operating system on its main control processor. The operating systems supported by Philips Semiconductors, however, are WindRiver’s VxWorks and Linux. It is the explicit strategy of Nexperia-DVP to make the best and widest possible use of the services and libraries offered by the operating system’s vendor. This not only includes real-time-kernel services (i.e. task scheduling) but also graphics, user interface management systems, protocol stacks, file systems etc.

- **Nexperia-DVP Platform Software**: The Nexperia-DVP platform software, developed and maintained by Philips Semiconductors basically deals with all handling of audio and video streams. It spans multiple processors and covers codecs and other DSP routines as well as sophisticated filter graph management, synchronization and buffer management. The Nexperia-DVP platform software offers an extendable, standardized programming interface (API) for most functions, and is programmed using standard industry practices [12]. The Nexperia Development Kit (NDK) allows customers and partners to develop software for the platform and is provided with the platform software.

- **Customer’s Middleware and Applications**: As the name implies, Philips Semiconductors generally does not offer more than platform software and operating system as described above (although counterexamples do exist for some markets). The middleware running on Nexperia-DVP products spans a huge variety of programming languages, vendors, sizes and application domains, so further generic discussion is not possible.

4.2 Multiple Processor Software Consequences

The Nexperia-DVP platform software often spans multiple processors in a single system. These may be general purpose CPUs (either on-die or off-die), or specialized VLIW audio/video DSP CPUs (TriMedia’s [5]). All these CPUs utilized are plain-C programmable, and the Nexperia-DVP software architecture has a high degree of transparency and uniformity across processors.

Most inter-process communication is handled via remote-procedure-calling (RPC), with automatic proxy/stub generation. Because most components in the Nexperia-DVP stack are available for execution on any
processor, this provides almost complete functional transparency with respect to which component runs on which CPU.

Of course, the execution and performance impact of partitioning tasks over CPUs is still very large, and the systems contain CPUs specialized towards particular tasks. So any Nexperea-DVP product typically contains a fixed partitioning of tasks over CPUs that is carefully designed up front. Dynamic balancing of loads and transferring of tasks is implemented only in some products, and then in a hard-coded, pre-designed fashion.

4.3 Streaming Component Architecture

Nexperea-DVP deploys a two-level component architecture for its streaming software: standard components and functional subsystems.

Standard Components: All audio and video streaming components in Nexperea-DVP follow a set number of coding and functional standards, collectively known as the Trimedia Software Streaming Architecture (TSSA). Of course the TSSA standard provides for different classes and types of components such as implementations of codecs, hardware accelerator drivers, buffer managers, and filter graphs. The TSSA architecture is internally layered; this is not covered further in this text. Examples for TSSA components are an MPEG Transport Stream (TS) demultiplexer, an MPEG2 video decoder and an AC3 audio decoder.

Functional Subsystems: From this library of standard components a number of larger functional subsystems are built that can be combined into a product. These subsystems are generally aligned with the top-level product functions and features as requested by customers. Examples are a DVD playback subsystem; an ATSC (Advanced Television Standards Committee) broadcast decoding subsystem, etc. It is important to note that different functional subsystems can contain (different instances of) the same component, e.g. both the ATSC player and the DVD player mentioned above use the same MPEG2 video decoder.

4.3.1 Reference Architecture

A reference architecture is deployed to describe the different possible combinations of functional subsystems as described above. This reference architecture is illustrated in Figure 4-8. The three main classes of functional subsystems in this diagram are players that decode compressed audio/visual (A/V) streams, recorders that encode uncompressed A/V streams and transformers that translate compressed A/V streams into different compressed A/V streams.
The Nexperia-DVP software reference architecture has an uncompressed A/V part (top-most part of Figure 4-8), which includes analog A/V input subsystems and a presentation engine containing all audio and video processing like sound equalizing, improvement and mixing, and video improvement, scaling and pixel/frame rate conversions. Since performance requirements are most stringent in especially the uncompressed video domain, Nexperia-DVP utilizes several hardware acceleration options for these operations.

The uncompressed A/V parts of the reference architecture (bottom 2/3rds of Figure 4-8), consists of a flexible network of five classes of functional subsystems:

- Uncompressed (Digital) Inputs: The subsystems contain the driver for a digital interface, either broadcast (push-mode) or networked or storage (pull-mode), and potentially any associated protocol stack s/w. Digital inputs deliver compressed A/V streams over a unified interface to any player (depending on format of course).
- A/V Players: These subsystems contain relevant components for stream demultiplexing, audio and video decoding, synchronization and any system information processing as required. Players can take their input from any uncompressed input, or from any recorder/transcoder through a loopback.
- A/V Recorders: These subsystems take uncompressed audio and/or video and encode this into a compressed A/V stream. They contain relevant components for encoding and stream multiplexing.
- Transcoders: These subsystems take an uncompressed A/V stream in, and generate another uncompressed stream out, which is in a different format, has a selected service, a different bit-rate etc. Transcoders can be implemented as full decoder/encoder combinations, or as dedicated transcoding implementations.
- Uncompressed (Digital) Outputs: These subsystems are the counterparts of the uncompressed inputs, and contain driver and protocol stack for the interface in question. For read/write capable interfaces (e.g. HDD or Ethernet), the uncompressed input and output subsystem may actually be implemented as a single subsystem.
5. NEXPERIA-DVP SYSTEM INTEGRATION

In the previous section we discussed techniques deployed in Nexperia-DVP to manage the large diversity of product types and instances, product family members and product use-cases that occur in the Nexperia-DVP family. In this section we will discuss how to make this variety of systems and operating modes actually work and achieve optimal performance characteristics.

As is the case with all embedded digital software architectures, getting it right with respect to the execution architecture of the entire Nexperia-DVP platform has proven to be one of the most challenging achievements in the Nexperia-DVP software architecture. Nexperia-DVP deploys a collection of sophisticated mechanisms that together allow a near optimal utilization of system resources, retaining excellent product stability and robustness.
When reading this chapter, one must realize a basic ‘rule of thumb’ that holds for Nexperia-DVP systems, as well as many other systems: resource overkill can make the execution architecture challenge an almost trivial one.

Roughly speaking, the basic architecture mechanisms provided in TSSA for Nexperia-DVP products require little/no additional ‘tuning’ if usage of critical resources (memory, time etc) is not required to exceed an average of 60-70%. Only when one or more available resources must be utilized to their fullest potential (even if only in some use cases) will it become significantly more complex to make each product and each use-case work.

The consumer electronics markets, where many Nexperia-DVP products are used, often require highly maximized usage of all available resources, which makes execution architecture a challenging and important topic for Nexperia-DVP. This is a direct consequence of balancing the cost limitations imposed by a mass consumer market and at the same time pushing the performance envelope with advance applications.

The following sections will elaborate on the basic principles of Nexperia-DVP execution architecture, and provide some quantitative guidance on the basic parameters.

5.1 Performance Characteristics

The following are the four ‘core’ performance requirements in most Nexperia-DVP systems:

- **The system must work under normal conditions:** The obvious one, although often the hardest to achieve.

- **The system must have a given (short) maximum input/output delay:** A long input-to-output delay not only translates into a lot of memory usage, it is also unacceptable for many audio/video consumer products for reasons of multiple product synchronization issues, user feedback and reaction time issues etc. Many Nexperia-DVP video products require (and achieve) input to output delays of one field or less, where a field refers to one-half of the TV frame that is composed of either all odd or even lines. In CCIR systems each field is composed of 625/2 = 312.5 lines (20ms), in EIA systems 525/2 = 262.5 lines (16.66ms). There are 50 fields/second in CCIR/PAL (European standard), and 60 in the EIA/NTSC (North American) TV system.

- **The system must behave gracefully under exceptional conditions:** Under conditions of erroneous inputs, inputs exceeding bandwidth specifications, misbehaving other parts of the system, etc., the Nexperia-DVP platform must show predictable and acceptable behavior. Not only should the system not crash, it should not display unacceptable artifacts on audio or video outputs.
The system must be as cost effective as possible: This requirement is very difficult to quantify, and can only be measured against competitor offerings in the market. However, the high price sensitivity and high product volumes of the markets in which Philips operates with Nexperia-DVP make spending effort to optimize and maximize system performance a reality. Of course this requirement remains a trade-off between development effort and unit cost (of Nexperia-DVP chip and additional external components).

5.2 Critical Resources

In Nexperia-DVP products the following three resources are regarded as the most ‘scarce’ or ‘expensive’ and are managed accordingly.

- Memory Bandwidth: This is the most critical quantity for many Nexperia-DVP products. Due to its unified memory architecture, all general-purpose CPUs and hardware accelerators share access to a single memory bus connected to external DDR RAMs. Ability to maximize memory utilization translates directly into cheaper memories. Current Nexperia-DVP systems typically can use 16-bit 133MHz DDR RAMs providing 533MB/s raw bandwidth, up to 32-bit 225MHz DDR RAMs providing 1.8GB/s raw bandwidth or more.

- CPU Cycles: The second most critical quantities for most systems are the CPU cycle budgets of the various CPUs in the system. This poses a ‘classical’ process scheduling challenge, as well as a complex budgeting challenge due to the interaction between memory bandwidth and CPU performance (further explained below).

- (RAM) Memory Size: The third and final of this list, memory size obviously translates directly to product cost (given the use of external memory). Due to the streaming nature of most Nexperia-DVP products, memory consumption for video buffers often accounts for half or more of the total system memory usage. In particular, the number and size of audio/video buffers required can be improved by optimizing the system. Typical memory sizes currently are 32 and 64 Megabytes.

In addition to the three quantities mentioned above, Nexperia-DVP deploys techniques to budget, schedule and manage, amongst other factors, network channel bandwidth, flash memory size and power consumption. These topics are beyond the scope of this text.

5.3 Bandwidth and CPU performance

One of the complicating factors in designing optimized schedules for unified memory products like Nexperia-DVP are the interaction between
memory bandwidth consumption and CPU performance. Primarily this is
due to the effects of hardware accelerator and ‘other’ CPU accesses to
memory on average memory transaction latency. Let’s take a closer look at
the requirements CPUs and hardware accelerators have on memory
bandwidth and latency.

- **Hardware Accelerators:** Most accelerators used in Nexperia-DVP are
asynchronous in nature and have an internal clock frequency significantly
higher than that required for the systems in which they are used. All
accelerators, even the synchronous ones, deploy FIFO buffers in their
DMA channels, resulting in a rather low sensitivity to memory
transaction latency.

As long as sufficient bandwidth can be made available during the desired
periods of execution the blocks will meet their performance
requirements. Obviously, as the desired period of execution approximates
that minimum execution time (determined by the accelerators internal
clock frequency), the sensitivity of the block to memory starvation
increases.

- **CPUs:** The CPUs deployed in Nexperia-DVP products (MIPS and
TriMedia) all include both instruction and data caches. Nevertheless any
CPU remains highly sensitive not only to available bandwidth, but
especially to the memory transaction latencies experienced when
consuming its bandwidth allocation. In current Nexperia-DVP products
with CPUs in the 300MHz range and 32-bit DDR memory in the
250MHz range, transaction latencies for a 128-byte cache-line refill
measure from 30 to 100+ CPU clock cycles. The increasing divergence
between CPU and memory speeds will only compound this in the future.

5.4 Effect of bandwidth on transaction latency

The memory transaction latency experienced by a CPU is influenced by
four different factors:

- **Minimum transaction cost:** The minimum length of a CPU memory
transaction is composed of three factors: cycles required in the bus and
arbitration system, actual memory access and DDR access penalty cycles.
These factors together give a minimum average transaction length that is
achieved only if no other blocks access the memory. For most Nexperia-
DVP products this is in the 20-40 CPU cycles range.

- **Transactions of other blocks that take precedence:** Some hardware
accelerators, or other CPUs will have the ability to issue transactions that
take precedence over the CPU’s pending requests. This can be due to
slotting, priority or other effects (see DDR arbitration below). These
transactions are the dominating factor in determining the additional memory access penalty.

- **Pending transactions of blocks that do not take precedence:**
  Transactions that would not take precedence over the CPU when arriving are completed when pending at the time of CPU transaction request arrival. In Nexperia-DVP systems it is possible to interrupt such transactions, but this option is seldom used due to the bandwidth loss it causes (generated by higher DDR inefficiency).

- **Other transactions of the CPU itself:** Higher bandwidth consumption of the CPU itself does not by itself increase its memory transaction latency. However, interaction between the buffering DMA patterns of hardware accelerators and high CPU transaction volumes does move hardware accelerator transactions out of the ‘quiet zone’: the period right after a CPU memory access when it is using its newly acquired cache line(s).

The intricate nature of the interaction between these factors makes simple stochastic models highly inaccurate, especially in the corners of the performance envelope. Simulations of bus behavior under realistic conditions show that transaction latency increases dramatically when (precedence taking) DMA load exceeds 60-70% of the available bandwidth. This is shown in Figure 4-9.

![Figure 4-9. Memory Latency vs. DMA Bandwidth](image-url)
5.5 Effect of transaction latency on CPU performance

The effect of long memory transaction latencies on CPU performance is governed primarily by a CPUs cache miss rate. The type of CPU, type of software and size of caches used influences this miss rate significantly.

In Nexperia-DVP code is classified as either control or DSP-type. This classification can be done at various levels of granularity (task, file, function, statement) depending on the accuracy required.

- **DSP-type code**: This type of code generally manipulates actual audio and/or video samples. Examples are MPEG codecs, filters, scaling operations etc. Roughly speaking this code is characterized by long complex expressions and frequently repeating tight loops. Also the average working set of a DSP task is small, often much smaller than the CPUs I-cache size.

- **Control-type code**: This type of code generally deals with controlling the hardware and software (DSP) operations on audio/video or with other parts of the system. Roughly speaking this code is characterized by lots of decision-making (switch/if-statements) and function calls. The average working set of a control task (or control portion of a task) is often very large, often much larger than the CPUs I-cache size.

**Table 4-1. Characteristics of DSP and Control Code**

<table>
<thead>
<tr>
<th>Characteristics</th>
<th><strong>DSP code</strong></th>
<th><strong>Control code</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
<td>Long expressions</td>
<td>Lots of switch/if statements</td>
</tr>
<tr>
<td></td>
<td>Highly repetitive loops</td>
<td>Many function calls</td>
</tr>
<tr>
<td>Working Set Size</td>
<td>Typically smaller than CPUs I-cache size</td>
<td>Typically much larger than CPUs I-cache size</td>
</tr>
<tr>
<td>Typical instruction repetition rate before</td>
<td>40 or more</td>
<td>Between 1 and 3</td>
</tr>
<tr>
<td>cache line invalidate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Typical CPU stall cycles due to cache misses on</td>
<td>20% and below on TM</td>
<td>80% and above on TM</td>
</tr>
<tr>
<td>moderately loaded system</td>
<td></td>
<td>60% and above on MIPS</td>
</tr>
<tr>
<td>Predominant Cache Misses</td>
<td>D-Cache</td>
<td>I-Cache</td>
</tr>
<tr>
<td>Typical bandwidth consumed for each effective</td>
<td>400KB/s on TM</td>
<td>6.4MB/s on TM</td>
</tr>
<tr>
<td>CPU Mcycle (i.e. excluding cache-stall cycles)</td>
<td></td>
<td>1.5MB/s on MIPS</td>
</tr>
</tbody>
</table>
Typical behavior of control and DSP tasks differs dramatically on both general purpose and VLIW CPUs. The preceding table (Table 4-1) illustrates the typical average behavior of these types of code on a Nexperia-DVP system. Of course, individual tasks and systems vary widely in their expressed behavior and are generally measured on a task-by-task basis.

From the numbers in the table above one can see that increasing memory transaction latency has a dramatic effect on the performance of the CPUs, especially in the case of control code execution. The DMA load at a specific point in time may influence the time required to complete a control task by a factor of two or more!

5.6 Effect of CPU performance on system scheduling

As all blocks in a Nexperia-DVP system except input and output blocks are controlled and triggered by software, the performance of the controlling CPU on these control tasks influences the timing and schedule of execution of these blocks. This brings us full circle: the scheduling of on-chip, hardware-based function accelerators influences memory bandwidth; memory bandwidth influences memory transaction latency; memory transaction latency influences CPU performance; and finally, CPU performance influences scheduling.

As said before, only when achieving bandwidth and CPU utilizations exceeding 60-70% do these interactions have to be completely modeled and measured. This is the case for many Nexperia-DVP products and the markets they serve.

The following sections will discuss the basic memory arbitration and scheduling techniques deployed, and then show the process used in Nexperia-DVP to obtain a working schedule for a use-case.

5.7 Memory Arbitration

Many use-cases in Nexperia-DVP products have, either probabilistically or deterministically, periods of execution in which the total amount of bandwidth requested by all CPUs and hardware accelerators exceeds the available bandwidth. To regulate the distribution of available bandwidth over the requesting blocks Nexperia-DVP utilizes an advanced DDR arbitration scheme that aims to achieve two goals:

- Fair distribution of bandwidth over requesting blocks
- Shortest possible transaction latency for CPUs

From a software point-of-view, the techniques deployed to achieve short CPU transaction latency are, although very important, of little interest in the execution architecture design process, since they require little or no tuning.
Most noteworthy is the decision whether or not to allow interruption of pending memory transactions to shorten CPU transaction latency. As said before, in most systems today this option is not used due to the additional bandwidth loss it causes.

The Nexperia-DVP memory arbiter deploys a sophisticated algorithm for distribution of bandwidth over requesting blocks. The essence of what is offered for software control, however, is simple, and constitutes two principles:

- **Guaranteed bandwidth:** In Nexperia-DVP it is possible to guarantee a specified amount of bandwidth to each individual block (CPU or hardware accelerator). This amount can be less than, equal to, or more than the amount needed/requested by the block at any point in time. Bandwidth guaranteed but not used by any block is added to the remainder pool, which is distributed on priority basis (next paragraph). If a block requests more bandwidth than has been guaranteed, the block will either be stalled (i.e. run slower), or consume additional bandwidth from the remainder pool.

Bandwidth guarantees are necessary primarily to guarantee proper operation of synchronous hardware accelerators (mostly input and output blocks), and accelerators that must run close to their maximum performance level (i.e. cannot handle a lot of stalls). Because of this Nexperia-DVP implements bandwidth guarantees within a fairly short period (several 100 cycles). This, in turn, strongly influences the CPU transaction latency. As a general rule of thumb, bandwidth guarantees are kept as low as possible to ensure proper operation of the system.

- **Priority based distribution of remainder:** In Nexperia-DVP the total sum of all bandwidth guaranteed to blocks is always significantly less (20\%-70\%) of the total available. The remaining bandwidth that has not been guaranteed plus the guaranteed bandwidth that has not been consumed is distributed over CPUs and accelerators in a priority order. Blocks may appear both in the guarantees list and the priority list, and frequently do.

### 5.8 Scheduling Techniques

In Nexperia-DVP several different scheduling techniques [13] are used to generate working system. We will first discuss the basic task scheduling mechanisms used, then we will study the scheduling of hardware accelerators, and finally we will discuss the actual approach used in current Nexperia-DVP products.
5.8.1 Scheduling of Tasks

Although Nexperia-DVP deploys standard embedded operating systems (pSoS, VxWorks, etc), it uses multiple additional mechanisms to schedule the various tasks on the CPUs.

- **Priority based scheduling**: This is the standard mechanism offered by the operating systems used. The various tasks in the system are all assigned a priority, either statically or dynamically, and execution is in priority order. Standard techniques like Rate Monotonic Analysis are used to determine working schedules.

- **Reservation based scheduling**: This technique assigns a CPU cycle ‘budget’ to a task or a set of tasks. Tasks are scheduled so that each task gets its assigned budget. This technique is combined with priority based scheduling by first assigning a budget to a group of tasks and then using priority based scheduling within this group.
  
  Usage of this technique is under development in Nexperia-DVP. It is expected to be most useful in achieving a higher level of robustness in use-cases dealing with multiple media streams (where each stream would be individually budgeted).

- **Hard coded scheduling**: If a group of tasks is required to achieve extremely tight schedules (with respect to memory size, bandwidth or CPU performance), it is sometimes necessary to completely plan out a schedule and execute it in a hard-coded fashion, i.e. in a fixed task-to-task order.
  
  In Nexperia-DVP this technique is used in the uncompressed video domain, where typically the number of operations is very small (less than 10), the consumption rates are very high (several 10%’s), and the requirements tight.
  
  This technique is significantly more robust and reliable than the other two, but also less flexible. The Nexperia-DVP methodology contains methods and tools to ease the complexity of designing and implementing hard-coded schedules.

5.8.2 Scheduling of Hardware Accelerators

Although scheduling of tasks as discussed above is important *per se*, it is important to note that a typical audio/video flow graph in a Nexperia-DVP system contains a mix of software (DSP) operations and hardware-accelerated operations. Almost all hardware accelerators in Nexperia-DVP can process up to a complete field/frame without software intervention. This makes determining a working schedule in essence a many-processor scheduling problem.
Some special techniques are used to make use of this parallelism potential (see also Figure 4-10):

- **Sequencing of operations**: This is the simplest of techniques. Essentially when one operation is executing on frame/field number N, the previous operation starts executing on frame/field number N-1. In this way two processors never operate on the same frame/field (in the temporal sense) at the same time.

- **Slicing**: Many hardware accelerators and DSP routines in Nexperia-DVP support operating on parts of a frame/field. This operating mode is called slice based processing. When two processors are concatenated and are both running slice-based, scheduling can be based on slices too. Basically, if processor A has completed its first slice of frame N, it is handed over to processor B. Now processor A starts executing on slice 2 at the same time that processor B starts executing on slice 1 of the same frame/field.

- **Staggering**: When two processors A and B are concatenated and are both sequential and localized, i.e. they read their input and write their output front-to-back, it is possible to start processor B before processor A is finished with the same frame. If the delay in starting processor B plus the minimum execution time of processor B is at least the maximum execution time of processor A, it is guaranteed that processor B will never overtake processor A on the same frame. Obviously, successful use of this technique depends on the ability to accurately predict minimum and maximum execution times of all processors in the system.

Obviously the techniques above require very different software control. Slicing is significantly more expensive than sequencing and staggering due to the higher task-switch or interrupt frequency it causes (software intervention is required to trigger every slice of both processor A and B). If the slices are small enough this becomes very significant. Staggering is not much different from sequencing in amount of software control effort required, but requires accurate time/threshold delays as a concept in the architecture.

It is also worth noting that the different techniques above have a different influence on the amount of memory required for audio/video buffers. When using sequencing, two full frame/field buffers are required between each pair of operations. When slicing still two buffers are required between each pair but they may be one slice in size. When staggering, most pairs can do with only one full frame/field buffer. Of course, actual buffers required must be studied on a case-by-case basis, and depends on the algorithms implemented by the processors and the history they require.
The following table summarizes the properties of these techniques:

<table>
<thead>
<tr>
<th></th>
<th>Sequencing</th>
<th>Slicing</th>
<th>Staggering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software effort</td>
<td>Lowest</td>
<td>Higher, depends on slice size</td>
<td>Low</td>
</tr>
<tr>
<td>Buffer memory</td>
<td>Largest</td>
<td>Smallest</td>
<td>Middle</td>
</tr>
<tr>
<td>Input/output delay</td>
<td>Longest</td>
<td>Shorter</td>
<td>Shorter</td>
</tr>
<tr>
<td>Requires of processors</td>
<td>Nothing</td>
<td>Must support slicing</td>
<td>Must be sequential and localized</td>
</tr>
</tbody>
</table>

5.9 Putting it all together

The approach used in Nexperia-DVP can be summarized as follows:
- The uncompressed video domain, i.e. all operations on uncompressed video buffers are singled out as "big ticket items". In most Nexperia-DVP systems the uncompressed video operations account for >60% of
memory size, bandwidth and CPU budgets.
For these operations the following techniques are used:
   a) A hard-coded schedule is designed that correctly sequences all
      operations with an acceptable bandwidth profile.
   b) Operations are sliced to reduce memory consumption, usually half
      or quarter field/frame.
   c) Slice operations are then staggered to further shorten input/output
      delay and memory consumption.
   - The remainder of tasks runs in the remainder of CPU cycles and memory
     bandwidth, using the following techniques:
   d) Basic priority based scheduling. Reservation based scheduling is
      being developed as an additional technique.
   e) Simple sequencing of operations.

Now we have discussed all techniques used in Nexperia-DVP to develop
working schedules for specific use-cases, we can summarize the process of
developing such schedules. We will describe the process for a single use-
case. Additional implementation techniques present in Nexperia-DVP to
manage the diversity of, parameterization of and switching between use-
cases are beyond the scope of this text. The process consists of two iterative
steps.

5.9.1 Step 1: to determine the uncompressed video schedule and
bandwidth profile.

First the flow graph of all uncompressed video operations is determined,
and raw data for each operation measured. This includes clock cycles
required, bandwidth required and, for CPUs, cache miss profiles and counts.

Then a schedule is designed, by hand, that sequences in time the various
uncompressed video operations using sequencing, slicing and staggering as
described above.

Tools are available to determine the resulting timing and bandwidth
profiles of the schedule design. Usually the system either does not meet its
deadline or exceeds bandwidth requirements, or both, and the process
becomes an iterative manual process of tuning the schedule until it fits.

The final schedule is then translated into executable form (by tools), and
bandwidth and CPU profile diagrams (also by tools).

5.9.2 Step 2: to determine the priority-based schedule of other tasks
in the system.

First, an average bandwidth, and CPU cycle budget, are computed from
the profiles in Step 1.
Secondly a complete list of tasks, deadlines, periods is generated and RMA analysis (rate monotonic analysis) done.

If the system can meet all its deadlines and CPU cycles remaining are acceptable to execute remaining non-hard-real-time tasks, we are done. If not, some initial assumptions or parameters must be relaxed and the process reiterated beginning at either Step 1 or Step 2.

6. CONCLUSION

Platforms are proving to be an effective strategy to cope with ever increasing product complexity, and to facilitate both Hardware and Software reuse. The on chip SoC communication infrastructure decisions are critical to effectiveness on hardware reuse. Proper infrastructure decisions can make the design timing closure friendly and derivative design friendly, as well as decoupling the Device IP library assets from the evolution of the SoC communication infrastructure. Similarly the Software architecture is critical to effectiveness in software reuse. Proper software architecture can provide the flexibility that is key to cope with the diversity of simultaneous sources and sinks of video/audio data with the need to support multiple simultaneous players, recorders, transcoders and rendering options.

The Philips Nexperia Digital Video Platform embodies state of the art approaches to SoC hardware and software infrastructure and is architected for maximum flexibility of creation of cost effective products. The Nexperia-DVP approach recognizes that current and future SoC functionality can only be exercised via software, and therefore Philips Semiconductors takes a system-level approach to SoCs, delivering a software platform and total system solutions. This dramatic shift from silicon to software and systems is well under way, and our Nexperia platform approach is proving to be one of most effective ways to meet today's SoC business challenges.

REFERENCES


