Optically Reconfigurable Processors

Jose Mumbru[a], Gan Zhou[a,b], Suat Ay[c], Xin An[b], George Panotopoulos[a], Fai Mok[b], and Demetri Psaltis[a]

[a]Department of Electrical Engineering, California Institute of Technology Pasadena, CA 91125

[b]Holoplex Inc., 600 S. Lake Ave. Suite 102, Pasadena, CA 91106

[c]Photobit Corp., 135 N. Los Robles Ave. 7th Floor, Pasadena, CA 91101

ABSTRACT

Reconfigurable processors bring a new computational paradigm where the processor modifies its structure to suit a given application, rather than having to modify the application to fit the device. The Optically Programmable Gate Array, an enhanced version of a conventional FPGA, utilizes a holographic memory accessed by an array of VCSELs to program its logic. Combining spatial and shift multipexing to store the configuration pages in the memory, the OPGA module is very compact and has extremely short configuration time allowing for dynamic reconfiguration. The reconfiguration capability of the OPGA can be applied to solve more efficiently problems in pattern recognition and digit classification.

Keywords: Programmable logic, Neural networks, Optical memory, VCSELs.

1. RECONFIGURABLE COMPUTING

Reconfigurable processors make possible to use more efficiently their resources by adjusting themselves depending on the characteristics of the input or on non-satisfactory previous results to better implement the target task.

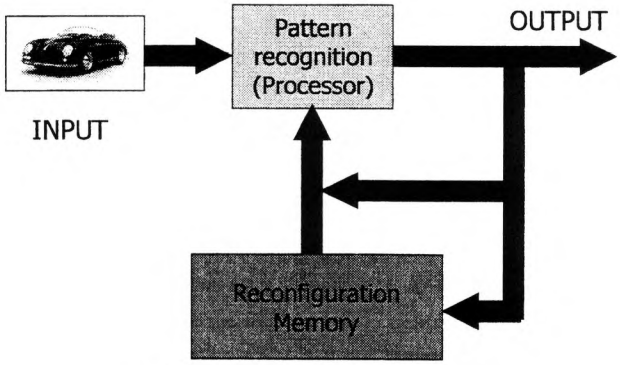


Figure 1. Reconfigurable processor.

1999 Euro-American Workshop Optoelectronic Information Processing: A Critical Review, edited by Philippe Refregier, Bahram Javidi, Proc. of SPIE Vol. 10296 (Vol. CR74), 102960D
 · CCC code: 0277-786X/17/$18 · doi: 10.1117/12.365910

Given an application, like pattern recognition, **figure 1**, the reconfigurable processor can be customized to deal with a specific class of objects. It can adapt itself in order to be robust to changes of orientation or illumination of the input object. By reprogramming, the same hardware can be time-multiplexed to carry out sequentially several tasks on the same input, or perform different task to different parts of the same input image. Reconfiguration also possibilities to implement learning by allowing the processor evolve in a controlled manner to learn the function that need to be computed.

2. FIELD PROGRAMMABLE GATE ARRAYS

A Field Programmable Gate Array (FPGA) is a device where this idea of reconfigurable hardware can be implemented. An FPGA consists of an array of Configurable Logic Blocks (CLBs) each one of them able to compute a basic logic function. Although different architectures for an FPGA exist, one of the more widely used is the symmetric array[1], **figure 2**. In this case, the CLBs are overlaid in a two-dimensional arrangement and interleaved with vertical and horizontal buses used to establish connectivity among them. Connections between segments in two different buses can also be performed by means of programmable interconnects in switching matrices. Finally, on the periphery of the chip, there are some input/output cells.

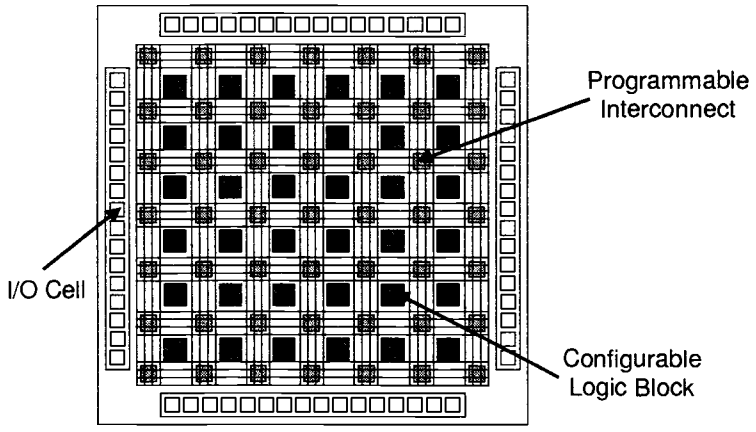


Figure 2. Architecture of a typical FPGA.

These devices have gained popularity due to the fact that they are between a software–oriented solution, like a microprocessor running a program stored in memory, and a hardware-oriented solution, like a specific circuit or ASIC. The FPGA based solution is faster than a microprocessor, speedups of several orders of magnitude have been achieved for some applications, and more flexible than an ASIC. FPGAs contain some hardware resources that can be programmed by the user to implement some given task and, by changing that configuration, the same hardware can be used to perform something totally different.

The configuration data of the FPGA is stored in an external memory and downloaded into the FPGA chip on demand. Although the size of these devices, in logic gates, can vary among different models and manufacturers, they can contain the order of 10^5 logic gates,

which means that the configuration data page can be as large as 1Mbit of information. In terms of Silicon area, the FPGA dies are relatively large, they can be 2cm by 2cm.

2. 1. CLB architecture

The basic building units of the FPGA are the CLBs. Despite the fact that there are CLBs based on other kind of logic blocks, like multiplexors or OR-AND arrays, the use of Look-Up Tables (LUTs) to synthesize logic functions is considered to enjoy of much higher flexibility.[1] A LUT can be seen as a small bank of memory where the inputs encode the address of a position in this memory, which stores the result of a pre-programmed logic function of the inputs. By changing the bits stored in the LUT, the function computed by this is altered.

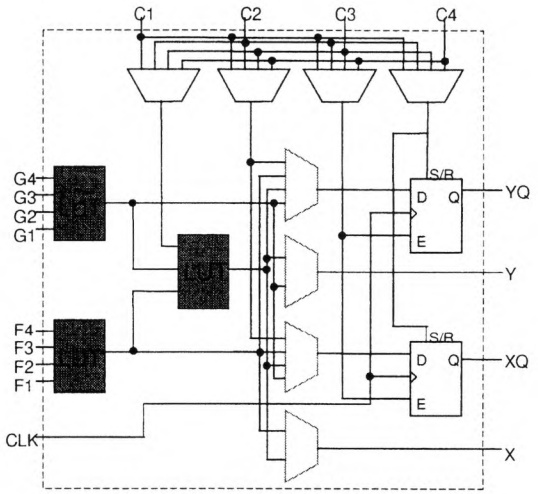


Figure 3. Schematic of a CLB.

The schematic of a LUT-based CLB is shown in **figure 3**. In this case, two sets of inputs, on the left-hand-side, feed two independent 4-input LUTs. A third LUT has the ability of combining the results of the LUTs from the previous stage, increasing the functionality of the CLB to implement more complex logic functions. The results of the LUTs are routed through a series of multiplexors governed by some control signals that come from the top of the CLB. The two outputs of the CLB are on the right-hand-side and they can be buffered if necessary by means of flip-flops. These registers allow implementing sequential logic in the CLB.

3. OPTICALLY PROGRAMMABLE GATE ARRAY (OPGA)

Based on this FPGA architecture, the OPGA is a device where the computation is still performed by programmable logic blocks and interconnects as in the conventional FPGA, but where the reconfiguration is brought into the chip optically. This optical reconfiguration capability results from interfacing an optical or holographic memory with

a Silicon chip where, in addition to the logic resources, an array of photodetectors has been incorporated, as is illustrated by **figure 4**. The holographic memory can store a large number of configuration templates that can be transferred down to the logic in the FPGA chip in a page-oriented mode. By taking the reconfiguration circuitry out of the FPGA chip, the OPGA can achieve a larger logic density, i.e. more CLBs can be implemented, than in the conventional device.

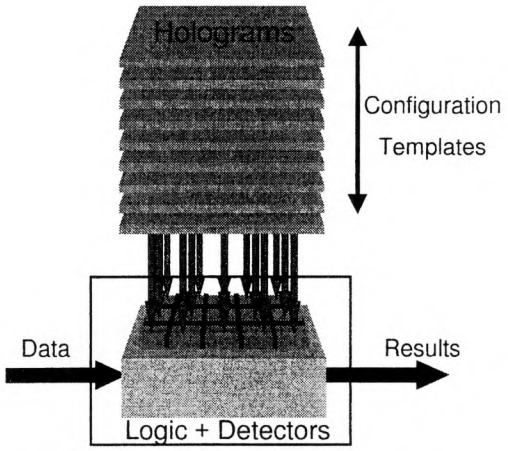


Figure 4. Interface between optical memory and FPGA.

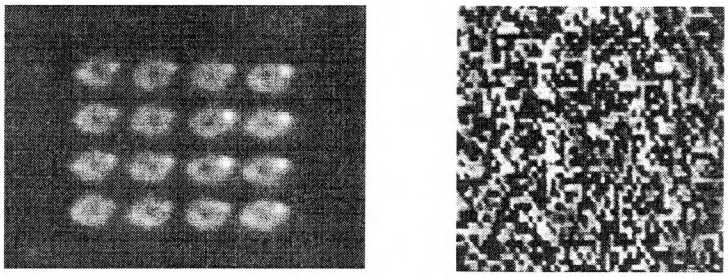
The OPGA is basically the integration of three main components or technologies: an array of VCSELs used to retrieve the templates stored in the memory; the optical memory that contains a large set of configuration contexts; and the VLSI chip that combines CMOS logic and photodetectors. Each one of these components presents a number of issues that need to be discussed in the following subsections.

In its initial implementation, the OPGA module is intended to operate as a Holographic Read-Only Memory (HROM), where a priori and for a given application, the user will decide the library of different configuration templates that needs to be stored in the memory. This eliminates all the Optics and Optoelectronics required to write in the memory, like spatial light modulator, in the OPGA module and makes it very compact.

3. 1. VCSEL array characterization

An array of VCSELs is used as light sources for the OPGA module to selectively retrieve one data page of the optical memory at a time. Experiments have been performed using a 4x4 symmetric array with 50μm pitch (**figure 5a**) provided by Honeywell. This array was tested and characterized to verify if this type of laser diodes is suitable for holographic recording. The elements in the array work in the red region of the spectrum around 680nm with a dispersion of values for the wavelength of just 0.2nm across the entire array. The VCSELs operate in single mode and the output power is on the range of hundreds of μWatts, from 50μW for the worst element up to almost 300μW for the best

one. These elements present very good coherence length, better than a meter, and a small divergence angle of 8.7°. When switching on the VCSELs, the risetime is less than 100ps, which allows switching speeds over 1GHz.



(a) (b)

Figure 5. (a) 4x4 VCSEL Array; **(b)** Hologram reconstructed using the VCSELs.

Their stability over time of the power and wavelength has also been studied. The power emitted by the VCSELs has been monitored over a period of 5 hours. In **figure 6**, it is shown the measurements for eight different elements in the array. As it can be observed, there is some initial drop in power due most probably to thermal heating of the P-N junction of the lasers. The long-term stability measured after 1-hour warm-up over the last 4 hours results to be on average 18.6%, although in the worst case can it be as large as 32.9%. A more realistic parameter is the short-term stability over a 15-minute interval after the warm-up, which turned out to be on average 4.8% and never larger than 9.2%.

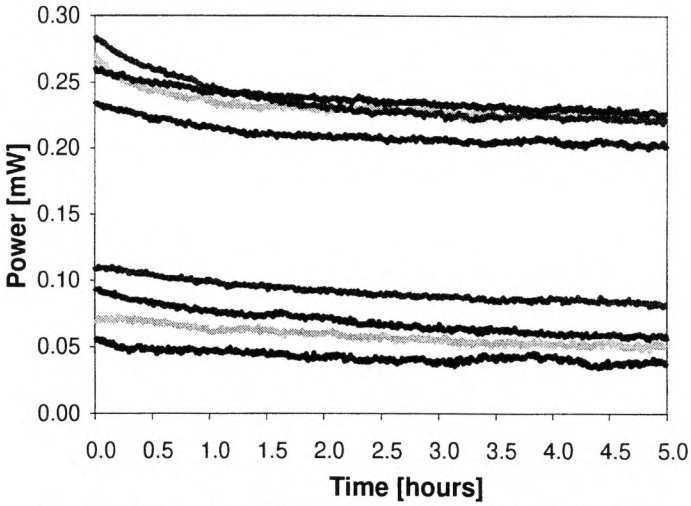


Figure 6. Power fluctuation in time for the VCSELs in the array.

The fluctuation in wavelength has also been investigated, and it has been found to be smaller than 0.016%. This very good stability, even without any thermal control of the VCSEL, makes the VCSELs adequate to record and readout holograms (**figure 5b**) and, therefore, a good choice for the OPGA system.

3. 2. VLSI Silicon chip design

The OPGA chip contains in addition to the logic circuit, as in a conventional FPGA, the array of photodetectors. The detectors must have very small pitch to result in a low area overhead and enough sensitivity to guarantee short integration time. There are basically two topologies to incorporate the photodetectors to the existing logic of the FPGA: sparsely overlay the detectors across the whole chip interleaving them with the logic, or conversely, pack all the detectors in a single large array on a specific region of the chip.

From the electronics point of view, it is more convenient the first one, so each pixel is detected exactly where it is needed to program the logic element. This makes unnecessary to distribute the detected signals all across the chip. However from the optics side, to have detectors spread over the entire chip means that the quality of the reconstructed hologram must be much more uniform over a large area. Therefore, the second topology makes the optics simpler because the hologram needs to be uniform in a much smaller region. However, this comes at the price of having to implement a more complex mesh of buses to deliver the detected signals to the logic blocks. Therefore, the first topology has been adopted for the OPGA chip. The detectors are implemented in small arrays in the CLBs and interconnect boxes.

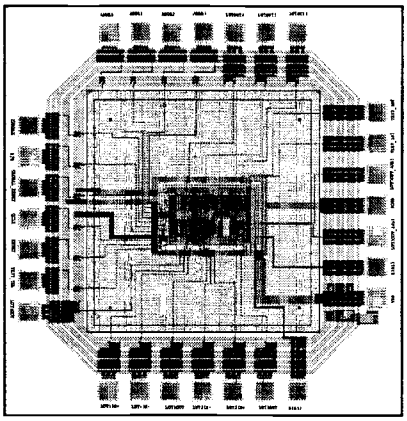


Figure 7. Chip floorplan.

A first prototype chip containing optically addressable logic blocks has been designed and fabricated. The CMOS 0.35μm process chip contains a 2x3 array of CLBs, **figure7**. The areas of the OPGA chip that are not light sensitive have been shielded with metal to avoid stray radiation interfering with the logic circuit. Each block is simply a 4-input 1-output LUT that can be optically programmed by an array of 16x2 photodetectors implemented inside the logic block. The light detection is done in differential mode to enhance the SNR of detected signal. The block diagram of one of these logic elements, **figure 8**, consists of a 4-bit decoder and an array of 16 latches, as in a conventional LUT, and a 16x2-pixel array and a two-stage amplifier. The size of each CLB is about 125μm x 85μm, and the overhead due to optoelectronics is 24.8%.

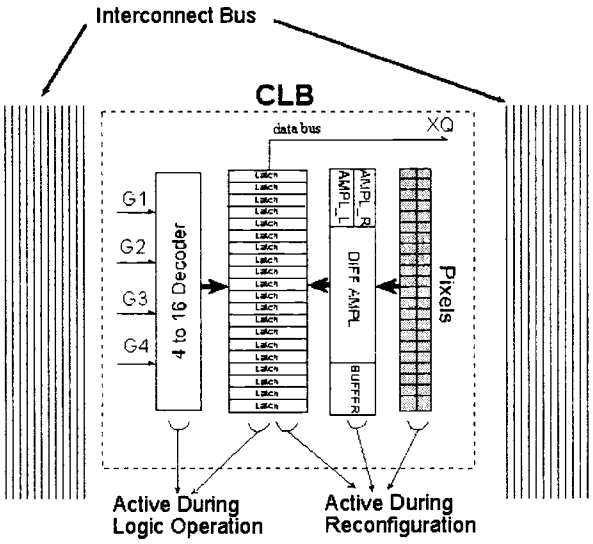


Figure 8. Block diagram of an optically addressable LUT.

Our present generation silicon chip is designed with passive pixels, mainly because of its smaller size and lower overhead on silicon area. We are in the process of designing our second-generation chip using active CMOS pixels to further optimize the speed and SNR of the optical reconfiguration circuitry.

3. 3. OPGA architecture

The technique used to store and multiplex the holograms in the optical memory determines the architecture of the entire module. For this reason, it is not possible to discuss on the holographic memory without giving a more general view to the system that encompasses both the VCSEL array and the array of photodetectors in the chip.

The initial design for the OPGA module is as depicted in **figure 9**. Each one of the VCSELs in the array is used to record a different hologram in the memory, which in this case is a thin layer of red-sensitive Du Pont photopolymer HRF700 100μm thick. For a matter of robustness, the layer of polymer has been sandwiched between two prisms. The technique used to multiplex the holograms is shift multiplexing with spherical reference.[2] Therefore, the shift selectivity of the material needs to be matched to the spacing between adjacent VCSELs in order not to have crosstalk between pages. Phase-conjugate reference is used to readout the data pages, so the reconstructions self-focus on the array of photodetectors on the FPGA chip. Since no additional optics is required for the readout, the module is very compact.

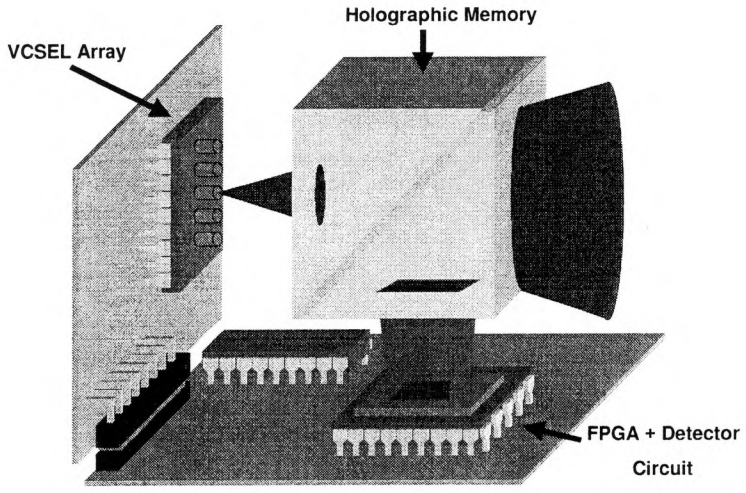


Figure 9. Optically Programmable Gate Array module.

Using this architecture, some experiments have been performed to record and multiplex the holograms in the memory. In the recording setup, as the schematic in **figure 10a** shows, the beam emitted by the VCSELs is collimated by the first lens. Then a beam splitter (BS) creates the reference and signal beams. An SLM is used to transfer the information that needs to be stored in the memory into the signal beam. The SLM image is projected to the plane P, where the photodetectors will be upon readout, by a lens. The lens on the reference arm focuses the reference beam creating a converging spherical beam used to record the hologram on the photopolymer. During readout the array of VCSELs should be placed on the plane where the converging reference beam focuses, so that the diverging beam emitted by the VCSEL creates the phase-conjugate reference that reads out the hologram.

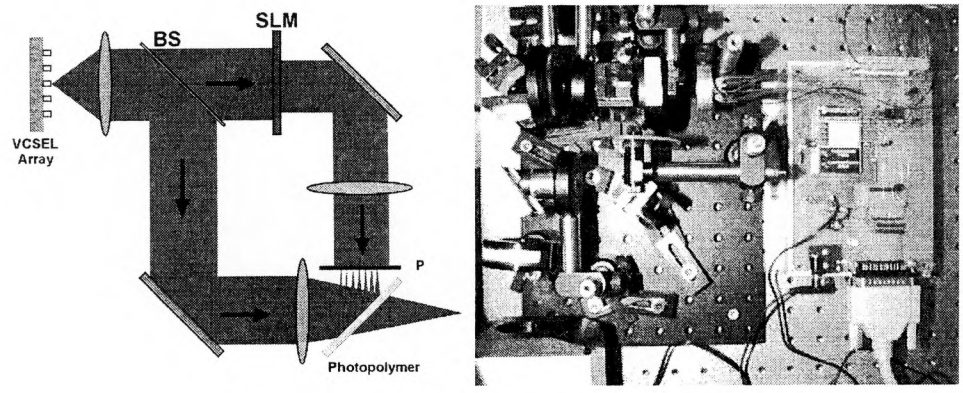


Figure 10. (a) Recording setup, **(b)** Experimental setup for the OPGA.

In the experimental setup, **figure 10b**, a single chessboard mask was used instead of the SLM and it was rotated to store different holograms. The board on the right corresponds

to the circuit to drive the array of VCSELs. **Figure 11** presents the reconstruction of two different pages. Since the VCSELs can be switched on and off in less than a nanosecond, different pages can be retrieved in very short time. This means that the OPGA module could switch rapidly among configuration contexts.

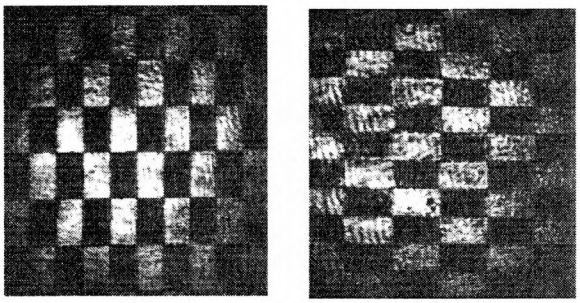


Figure 11. Detail of the reconstruction of two multiplexed holograms.

A number of problems have been encountered in this architecture, not intrinsically related to the architecture itself but to the VCSELs, which were not specially designed for this application, and the medium used in the optical memory. The main problem, noticeable in **figure 11**, is that there is some crosstalk between holograms. The reason is that for the thickness of the medium, 100μm, the shift selectivity of the memory is larger than the spacing between VCSELs and, consequently, Bragg mismatch between neighboring holograms is not achieved. Placing the VCSELs much closer to the optical medium would decrease the shift selectivity of the memory. However, the small divergence angle of the VCSEL requires placing the array far enough as to overlap the entire signal beam in order to record the hologram. Therefore, these two requirements result to be incompatible. Nevertheless, this limitation could be overcome by having a custom array of VCSELs with larger separation between elements and larger divergence angles. Another possibility would be to place a lenslet array on top of the VCSELs in order to increase their divergence angle.

Another problem is the quality of the reconstructed holograms. In this architecture, the signal beam spot on the material is relatively large, which makes the recording very sensitive to irregularities in the optical medium. Nevertheless, a more serious issue is the power required per VCSEL to have short reconfiguration times in the OPGA. If we target the device to fully reconfigure in just 50μs, a straightforward calculation assuming M/5 and 100 pages of 1Mbit each reveals that each laser should output a few mWatts. However, we know after the characterization of the VCSEL array that these elements output a few hundreds of μWatts at most.

Mainly because of the limitation in power, we decided to evolve towards a new design for the OPGA where we could still have short reconfiguration times, in the range of tens of μs, but with a not so demanding requirement on the power per VCSEL. The technique used to store the holograms combines both spatial and shift multiplexing. The main difference with respect the previous architecture is that upon recording, **figure 12a**, a lens focuses the beam that impinges the SLM down to a small spot on the recording medium. By changing the angle of incidence of the beam on the lens, the signal spot focuses on a different location in the material, which is partially overlapping with the previous ones.

The pages of data are recorded in these partially overlapping circles that span a stripe on the optical material. To achieve Bragg mismatch among holograms, a converging reference beam needs to be shifted accordingly to illuminate the corresponding signal spot.

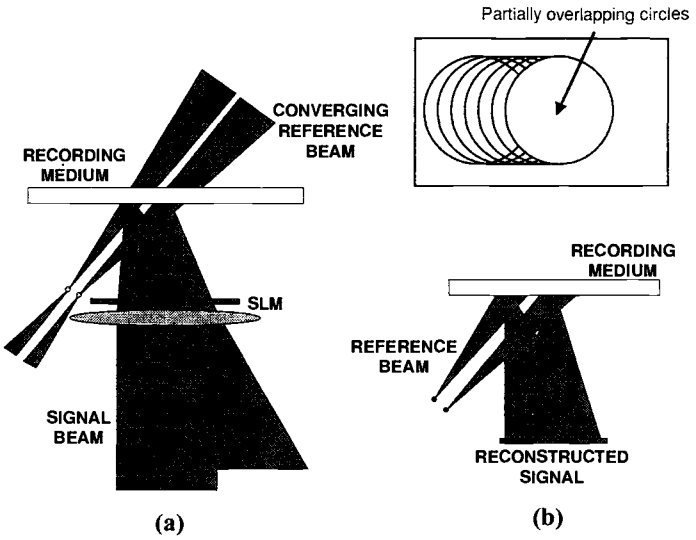


Figure 12. Multiplexing technique for the OPGA: **(a)** Recording, **(b)** Read-out.

In the recording setup (**figure 13**), a laser diode with enough coherence length can be used instead of the VCSEL array. The beam emitted by the diode is collimated and splitted into the signal and reference arm. The signal beam passes through a rotation stage and a 4-F system that changes its angle before it illuminates the SLM. The reference beam is focused by a lens mounted on a mechanical scanner used to translate the beam beyond the shift-selectivity of the optical medium.

During readout, the system becomes very compact (**figure 12b**) because of two reasons. First, we use reflection geometry for recording, so upon readout the reading beam from the VCSEL and the array of photodetectors are both located on the same side of the material. Secondly, phase-conjugate readout makes unnecessary the use of any extra component. Placing the VCSEL array at the plane where the converging reference beams used for recording focus, each VCSEL illuminates one of the spots in the memory and all the reconstructed images back-propagate to the plane of the SLM where the photodetector array is upon readout.

As benefit from this new architecture, we obtain an important increase in the diffraction efficiency per hologram, which now scales not as the total number of stored holograms but the number of overlapping ones at any location. A simple system design calculation helps to illustrate the reduction of the power required per VCSEL. Assume that we design the memory to store 100 configuration pages, each page is 1000x1000 pixels with a pitch of 5μm, and we use an optical material 200μm thick with M/5. If the number of overlapping holograms is set to 20, then the diffraction efficiency per hologram is as high as 6.25%. Therefore if we consider the photon-budget at the photodetectors and impose

1000 photons to be detected in order to have an acceptable Signal-to-Noise Ratio (SNR), we can parameterize the power per VCSEL as a function of the integration time of the detectors. If the integration time is set to be just 1μs, each VCSEL must output 6.4mW. If a longer integration time is allowed the power required per VCSEL falls into the range of values of the present VCSEL array (320μW provides in 20μs enough photoelectrons for a good SNR).

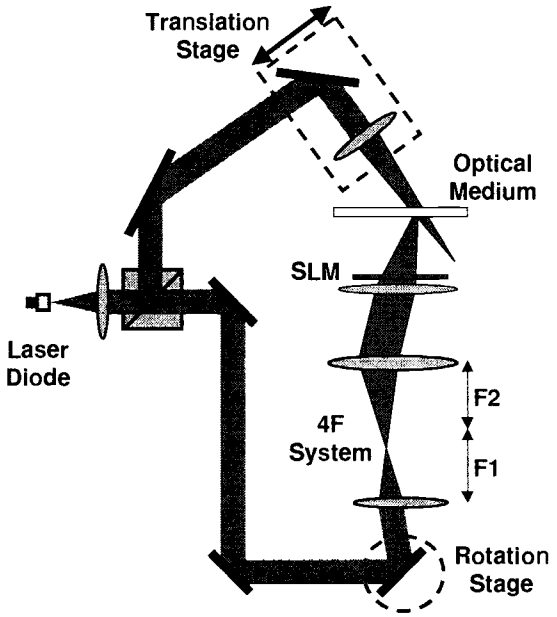


Figure 13. Recording setup combining spatial & shift multiplexing.

Another advantage of this architecture is that the total area used on the recording medium is smaller than in the original OPGA design. If the lens that focuses the signal beam has a focal length of 10mm, the signal spot size on the material is just 2.7mm in diameter and 100 holograms can be stored on a stripe 2.7mm wide by 16mm long. Given the small dimensions of the area where the pages are recorded, the holograms are much less sensitive to any non-uniformity on the medium and, consequently, the quality of the reconstructed images is better.

3. 4. Optical materials

Once decided the mechanism to store the configuration templates in the optical memory, we need to consider which optical media are appropriate for the OPGA system. On the materials side, there are mainly four options: Du Pont photopolymer, PQ-doped PMMA, Polaroid film and $LiNbO_3$.

In the experiments that have been carried out, red-sensitive Du Pont photopolymer, the HRF-700 series, has been used. This material presents very good dynamic range, M/# can be as large as 5, in proportion to its thickness, between 10μm and 100μm and it also has high sensitivity. For the sake of comparison among the different materials, and due to the

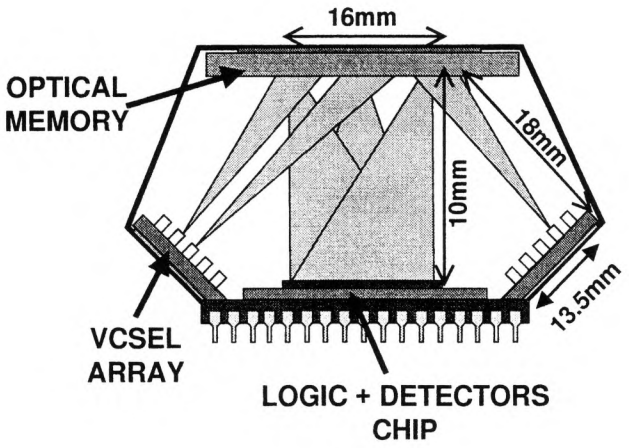
non-linearity of the sensitivity with thickness for polymers, a better way to express the recording speed is the exposure energy required to achieve 1% diffraction efficiency. For the Du Pont such energy is about 5mJ/cm^2. However, the main drawback of this polymer is its poor optical quality due to non-uniformity in the material, which distorts the reconstructed images. This problem becomes more important as the pixel size is reduced, even if phase-conjugate readout is used.

It has also been explored the possibility of using phenanthrenequinone (PQ)-doped PMMA.[3] This material shows good optical quality and M/#s between 2 and 5. The recording speed is 200mJ/cm^2 for a 1mm thick sample at green wavelength. However, the material has extremely poor absorption in the red, at the wavelengths of the VCSELs. This means that the material is not useful for the OPGA unless we use green light sources, which does not seem a plausible solution.

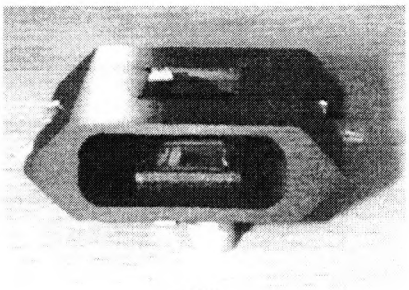
Another alternative is the Polaroid film, which presents properties superior to Du Pont photopolymer, M/10 and 150mJ/cm^2 exposure energy for a 200μm thick sample and at the same time higher optical quality. The red sensitive Polaroid material is still under development and availability is the main issue. However, the most solid choice seems to be iron doped $LiNbO_3$, which is red-sensitive and presents acceptable M/# (M/2.6 for an 8mm thick 0.02%wt Fe sample). Although there is a large drop in the sensitivity in $LiNbO_3$ when compared with the other materials, this is relatively unimportant for this application since the system utilizes the holographic memory as a ROM.

3. 5. Module packaging

The last topic to be discussed is the one regarding the integration of the three major components, VCSELs, optical memory and CMOS chip, on a package. The main goal is that the OPGA module needs to be small enough to be mounted on a board in a computer. The main constrain is in the height of the module, and this depends only on the focal length of the lens used before the SLM. As already discussed, this distance can be made as little as 1cm. The new architecture, described in subsection 3.3, is very compact due to the lensless readout and to the small size of the area of recording medium used to store the holograms. The package shown in **figure 14** houses the optical memory on the top rectangular window. The VCSEL arrays, integrated on both sides, retrieve the holograms detected on the chip located on the bottom of the package. The package also needs to be robust to ensure the alignment between all the components. It is important to preserve the one-to-one correspondence between the pixels in the hologram and the photodetectors on the chip and also to avoid any change on the areas illuminated by the VCSELs on the optical material.



(a)



(b)

Figure 14. OPGA module package: **(a)** schematic; **(b)** mechanical design.

4. PERFORMANCE COMPARISON

FPGAs have been traditionally used as accelerators. In a typical arrangement, as shown in **figure 15**, the FPGA is setup as a coprocessor that is controlled by the microprocessor. For a given application, if there is some task that is computationally very expensive, the microprocessor can program the FPGA to perform that task much faster than if it was executed by the main processor. The microprocessor just feeds the data into the FPGA and waits for the results, and all the cumbersome computation has been hardwired inside the FPGA. In most cases, the FPGA is configured only once and this configuration is downloaded into the FPGA off-line, before the execution is started. Although the device is able of being reprogrammed multiple times, the user typically does not take advantage of this feature. The main reason for not reconfiguring dynamically the device, i.e. to change its internal configuration once the execution has started and some data is already flowing into the device, has been the small communication bandwidth between the external configuration memory and the FPGA chip itself. The configuration data for a medium size FPGA can be in the range of 1Mbit. Upon configuration, this data is downloaded serially by shifting a long bitstream into the FPGA. The transfer data rate between memory-and FPGA can be on the range of 100 Mbit/s, which results on configuration times of tens or even hundreds of milliseconds. These long reconfiguration

times, if compared to clock cycles of just tens of nanoseconds, makes the reconfiguration time to be an important overhead.

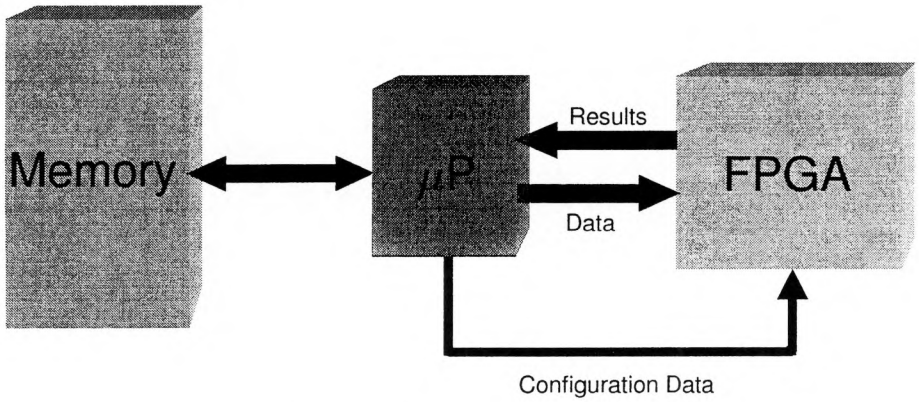


Figure 15. FPGA as accelerator.

The OPGA can overcome the limitations of the conventional FPGA with a much higher communication bandwidth between memory and FPGA chip because of the parallel interface due to the holographic memory. The reconfiguration time is only limited by the integration time of the photodetectors on the chip, given that the time to switch the VCSEL on is negligible. In the previous section we have seen that for some reasonable numbers for the optical material and VCSELs this time can be made to be just tens of microseconds, which is three orders of magnitude better than for the conventional FPGA. This reduction in the reconfiguration time possibilities run-time reconfigurable computing and opens a new computational paradigm.

4. 1. Run-time reconfiguration

Run-time, or dynamic, reconfiguration refers to the situation where we would like to be able to switch among different contexts once the execution has started, which means that it is necessary to do it in real time. One application that could directly benefit from the idea of dynamic reconfiguration is real-time video processing, **figure 16**. In this case the video camera supplies a new video frame every 33ms, and this image needs to be processed by a bank of filters that typically contain the order of 100 kernels. We would like to implement each one of these convolution kernels on the same FPGA by time-multiplexing its hardware, and do all the convolutions before the camera provides the next video frame. The available time per convolution result in just 330μs, which is clearly insufficient to fully configure the device loading the configuration data from the memory and process the input image.

Consider now the OPGA applied to the same real-time video-processing problem. As discussed in section 3, the current VCSEL array could allow reconfiguration times of 20μs, which means that 100 reconfigurations can be flashed into the OPGA chip in just 2ms, leaving 31ms to compute 100 convolutions. Assuming 512x512-pixel images

encoded with 8bits/pixel, and that the OPGA has 64-line data bus and can be clocked at 120MHz, then the processing time per image is 273μs, shorter than the 310μs that were available. Therefore a single OPGA can run the application in real-time, while for the conventional FPGA, the only way to work in real-time is to go to a multi-FPGA system, where 100 FPGAs working in parallel would be required. However, this solution supposes high power dissipation and an expensive cost in hardware. This simple example reveals the potential of an optically reconfigurable processor.

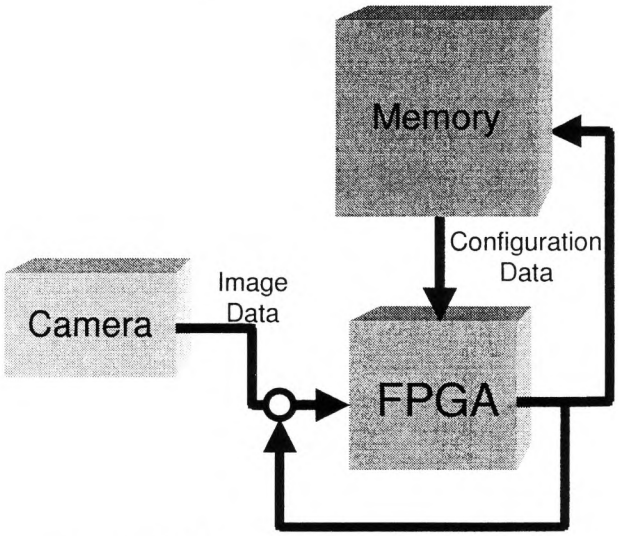


Figure 16. Real-time video processing application.

4. 2. Computational density

The fast interface with memory makes possible to bring the reconfiguration time down to the range of microseconds. The OPGA uses this fact to reprogram the chip in real-time. However, the OPGA is not the only approach to achieve dynamic configuration. Some FPGA manufacturers, like Xilinx, have tried a different solution. They have developed FPGAs where a cache memory is built in the chip to store locally, inside the FPGA, a few configurations, **figure 17**. The cached configurations can be accessed very fast, in nanoseconds, and transferred in parallel to the logic. The cache memory can store those configuration templates that are more frequently used, reducing the number of accesses to the slow external memory to fetch the new configuration. The asymmetry between many accesses to a fast cache memory and only a few accesses to the slow external memory helps to amortize the cost of the reconfiguration during run-time.

The on-chip cache memory FPGA decreases the reconfiguration time at the expense of decreasing the amount of logic implemented on the chip, since the same die area needs to be shared to implement banks of SRAM. Switching fast among contexts stored in this local memory implies higher power dissipation of the chip.

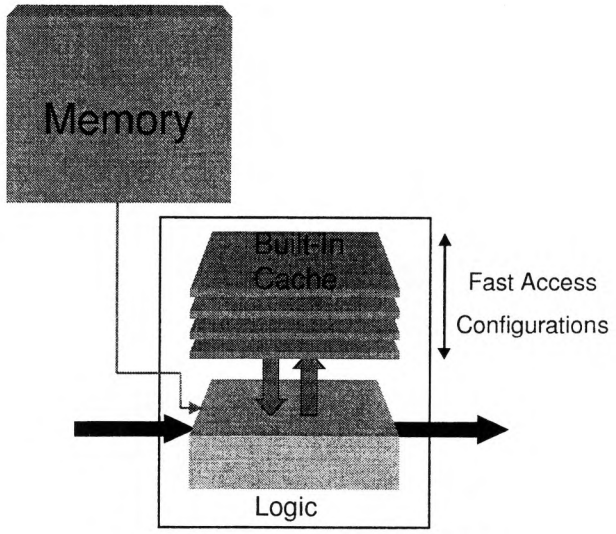


Figure 17. On-chip cache memory FPGA.

In this section, we present a comparison between the two architectures (**figure 18**), OPGA versus cache-memory-based FPGA, in terms of logic density. In both cases, it is necessary to trade off some area of the chip to implement either an array of photodetectors, in the OPGA, or banks of RAM memory, in the cache-based FPGA. We want to investigate which one of these two approaches allows for a higher computational capacity defined in terms of CLBs.

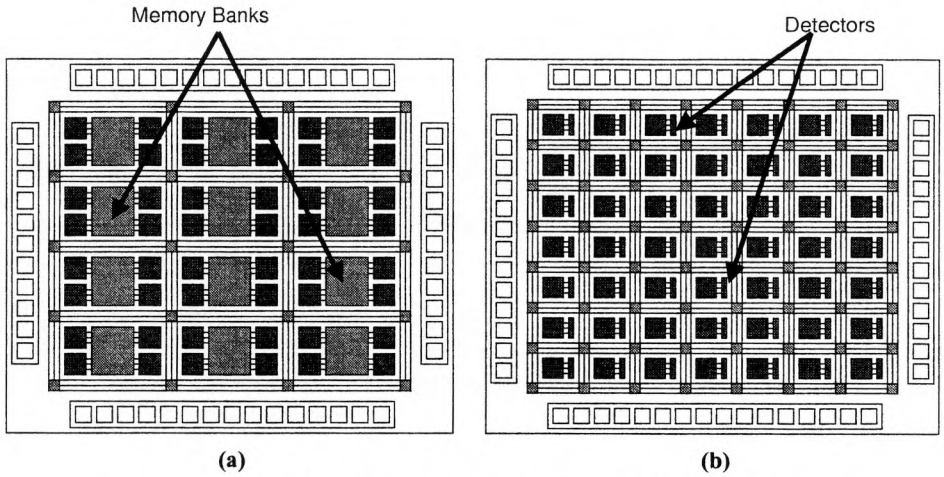


Figure 18. Area trade-off: **(a)** Built-in Cache memory FPGA; **(b)** OPGA.

The model analyzes the number of CLBs that can be implemented in a fixed die area as a function of the number of configuration templates that the system is dealing with. This model does not make a distinction between logic blocks and programmable interconnects and buses. It only counts how many CLBs can be implemented in the available area after having subtracted the area dedicated to photodetectors or cache memory. This number

can be understood as CLB-equivalents, so that a switching matrix or bus segments can have a cost in terms of CLB-equivalents. Thus, it is left as a chip design decision how to balance that number of CLB-equivalents among real CLBs, interconnect matrices and buses to reach some optimum. The number of CLBs, N_{CLB}, can be expressed as the ratio of the total die area and the area per logic block, A_{CLB}, and its area overhead due to either photodetectors or cache memory. If b_{CLB} stands for the number of bits required to configure a CLB, then the optical overhead is b_{CLB} times the area of a pixel detector, $A_{Detector}$. The cache-memory overhead is given by the size in pages of the RAM, N_{Config}, and the area of a 1-bit memory cell, A_{RAM}. The number of CLBs for the cache memory based FPGA and for the OPGA is given by equations 1 and 2 respectively.

$$N_{CLB} = \frac{Die\ Area}{A_{CLB} + N_{Configurations} \times b_{CLB} \times A_{RAM}}, \tag{1}$$

$$N_{CLB} = \frac{Die\ Area}{A_{CLB} + b_{CLB} \times A_{Detector}}, \tag{2}$$

If some numerical values are taken into consideration, the number of CLBs for both situations can be plotted as a function of the number of templates. For the simulation in **figure 19**, the CLB considered is similar to the one in the XC3000 series from Xilinx. The logic block requires 64 bits of configuration and its area is 291x156 μm^2. For the banks of memory, both DRAM and SRAM has been considered, although due to the non-volatility of the optical memory it is more fair a comparison with SRAM rather than DRAM. The size for 1-bit memory cell is $8\mu m^2$ for SRAM and $1.5\mu m^2$ for DRAM. The analysis has studied two sizes of detectors: 5x5 μm^2 and 2x2 μm^2 pixel size.

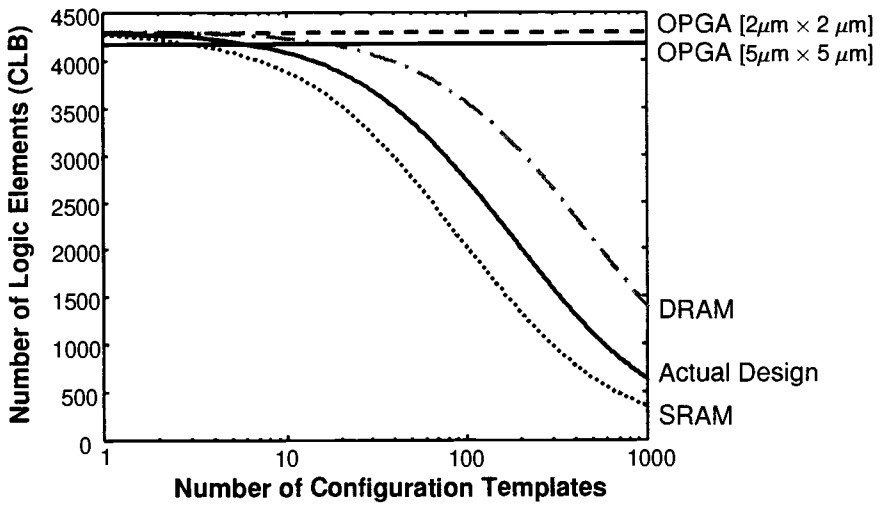


Figure 19. Performance comparison between OPGA and cache memory FPGA.

As it can be observed in **figure 19** and described by equation 2, the overhead due to the photodetectors remains constant independently of the number of templates in the application. However, the performance of the cache-based FPGA decreases as more and

more templates need to be stored in the chip using up the area dedicated to logic. Note that in **figure 19** the horizontal axis is in logarithmic scale. For a small number of templates, the conventional FPGA outperforms the OPGA since almost no reconfiguration is involved in the application. Nevertheless, the breakeven point occurs around 3 or 4 configuration pages. As expected from the difference in area, the decrease in performance for a SRAM cache starts sooner than for the DRAM cache. It is important to mention that this model does not consider the overheads in the memory due to additional circuitry like sense amplifiers or row/column decoders. The reason is that it is difficult to quantify such overheads since they can significantly vary from one chip to another. However, it has been represented the curve that corresponds to an actual design of DRAM-cache-based FPGA.[4] The effect of these overheads is to shift the curve downwards. The OPGA outperforms clearly the cache-based FPGA for applications involving the order of hundreds of templates. This defines a domain of applications that can be carried out much more efficiently using an OPGA.

5. APPLICATION: CLASSIFICATION OF DIGITS

In this section we present an application of OPGAs in pattern classification using neural networks. When implementing a neural network based classifier the number of units in each hidden layer required to achieve a certain correct classification rate specification depends on the number of output units (i.e. the number of classes). Therefore in a context of limited hardware resources there are problems which cannot be solved using such an approach. In what follows we will use a toy problem to demonstrate the limitations of the aforementioned approach and we will present a number of strategies which overcome these limitations using the flexibility provided by OPGAs.

In this problem, we are presented with an 8 by 8 image of a handwritten digit. Our system will classify the digit in one of 10 categories. The database of handwritten digits used in the simulation is composed of a training set of 3823 digits and a test set of 1797 digits. The digits are almost uniformly spread across the 10 categories.

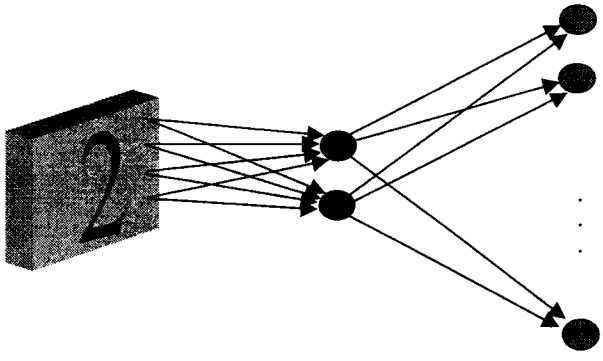


Figure 20. Neural network architecture.

All neural networks used have 64 input units and 2 hidden units, as represented in **figure 20**. The number of output units varies depending on the exact use of the net according to

the classification strategy. The networks are trained using back propagation with momentum, based on a random initialization of the weights. We use 500 training iterations.

5. 1. Classic neural classifier

To show the limitations in a classic classifier, we first try to classify digits from an increasing number of classes using just one neural network. The correct classification performance obtained is summarized in **figure 21**.

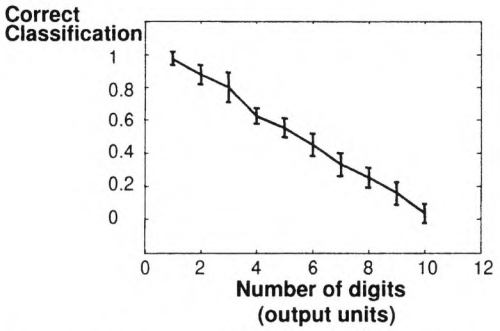


Figure 21. Performance of a single neural network classifier.

We see that the performance decreases rapidly as the number of classes increases. To overcome this problem we should either use a neural network with more hidden units (if the hardware resources allow such a choice) or an alternative strategy. Three different strategies to the problem of digit classification are investigated: a fully parallel search, a tree search and a sequential search in both exhaustive and non-exhaustive modes. From these strategies, the first one does not solve the hardware resources limitation problem but it is presented here to illustrate the tradeoffs we can make using OPGAs. A brief discussion of each one follows.

5. 2. Fully parallel search

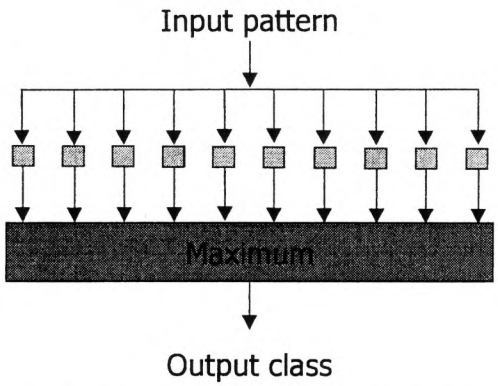


Figure 22. Fully parallel search architecture.

This strategy consists of the simultaneous implementation of 10 neural networks, each of which is trained to recognize a particular digit. When a digit is input to the system it is presented to all 10 neural networks. The network with the higher output determines the class that is output by the system. We could think of this architecture as a 4-layer neural network with the output unit of each specific net being a hidden unit, driving a winner-take-all circuit at the output layer.

This architecture consumes the greatest amount of hardware resources, but is the fastest one and yields optimal results regarding correct classification. The reconfiguration data required is 10 pages, with each page containing the reconfiguration data for the weights of one of the digit-specific neural networks.

5. 3. Sequential search

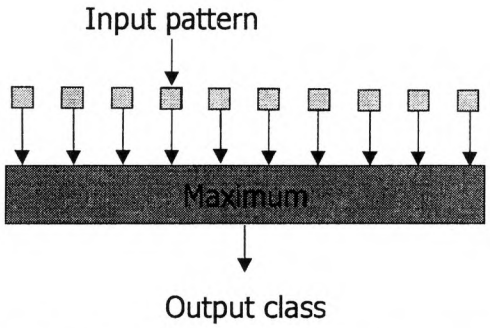


Figure 23. Exhaustive sequential search architecture.

This strategy consists of the sequential implementation of 10 neural networks, each of which is again trained to recognize a particular digit. We can have two variations of this strategy. The one is exhaustive search (**figure 23**): when a digit is input to the system all 10 neural networks are implemented sequentially by reconfiguring the OPGA, and their outputs are kept in a buffer. Finally all outputs are compared and the highest output determines the class that is output by the system.

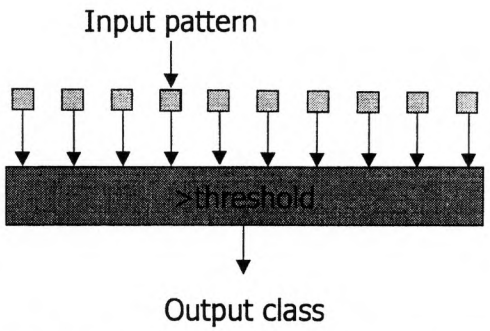


Figure 24. Non-exhaustive sequential search architecture.

The second one is non-exhaustive search (**figure 24**): when a digit is input to the system the neural networks are implemented one by one. When the output of a network exceeds a certain threshold the system outputs the corresponding class and the search is aborted.

The exhaustive search has the same correct classification performance with the fully parallel search and can be though as a sequential implementation of the winner-take-all network described in the previous section. It requires minimal hardware resources but is the slowest strategy. The required reconfiguration memory is 10 pages.

The non-exhaustive search has inferior correct classification performance, but requires on average half the time, assuming that all classes are equally populated. It requires the same minimal hardware resources and 10 reconfiguration pages.

5. 4. Tree search

The two previous cases can be thought of as extreme cases of tradeoff between hardware resources and time. We present now an intermediate strategy that is based on a tree structure:

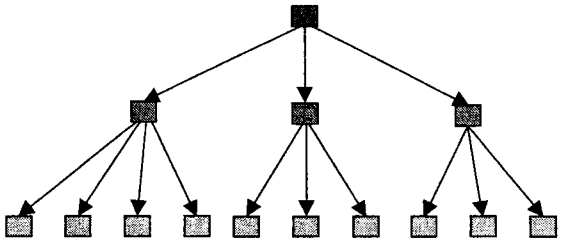


Figure 25. Tree search architecture.

At each node we use a neural net to classify the current digit in one of the subcategories of that node. The OPGA reconfigures itself to implement the neural network in the next node, as dictated by the classification in the current node. At the root node we divide the digits in three categories, namely {1,2,3,8},{0,5,9} and {4,6,7}. This grouping was made as an attempt to keep maximally correlated digits within the same group, in order to facilitate the classification task of the first layer. At the next level the current digit is classified in one of 3 or 4 categories and again this classification dictates the next classifier to be implemented. The networks of the final layer are used to confirm or overturn the previous classifications. They are trained to recognize only one digit, and a result their correct classification rate is very high (typically above 99%). If the neural network at the leaf gives a positive result (it's output exceeds a certain threshold) then we assume that the classification is correct, and the tree outputs this result. If not we move back to the previous level and follow the path dictated by the next largest outcome of that node. If all leaves of a node of the second level turn out to overturn the classification then we move back to the root and again follow the path dictated by the next largest outcome. If all output nodes overturn their respective hypotheses we conclude that the input is not a digit.

5. 5. Classification performance

In order to evaluate the different strategies we simulate them. The results of our simulations can be summarized as follows:

	Fully Parallel	Tree	Sequential	Exhaustive Sequential
Correct classification for the training set	0.98378	0.97253	0.97332	0.98378
Correct classification for the test set	0.93656	0.91096	0.89705	0.93656
Average number of reconfigurations for the training set	1	3.5124	5.3421	10
Average number of reconfigurations for the test set	1	3.9104	5.0534	10
Required memory	10	14	10	10
Required area	10	1	1	1

We now examine the performance and requirements of each strategy:

- Regarding the correct classification performance the fully parallel and exhaustive sequential strategies are the best. Their superiority is easily interpreted by the fact that they can rely on the output of all the 10 single-digit nets to make a decision, while the other two strategies produce their answer as soon as possible, and therefore are forced to base it on less information.

- Regarding hardware/area consumption the fully parallel strategy is the most expensive, since it requires the implementation of all 10 neural nets simultaneously. All other strategies implement only one neural net at a time.

- Regarding the classification speed (number of reconfigurations required to reach a final decision) the fully parallel strategy is obviously the best, and the exhaustive sequential search the worse. The point of interest is the comparison of the tree search and the non-exhaustive sequential search. As we can see the two strategies have comparable correct classification performance, but the tree search produces a result faster that the non-exhaustive sequential search. This trend should be intensified when the problem involves more classification categories.

- Regarding memory the tree search strategy is the most expensive (order of $n \log n$, where n the number of classes), but this should not be a problem since in OPGAs the reconfiguration storage space is ample.

The conclusions drawn from this simple problem can be extended to more complex situations. Clearly as the number of possible classifications of the input data increases the hardware requirements for the parallel search and the number of reconfigurations for the sequential search grow in proportion. The tree search strategy allows us to decrease the

number of steps at the expense of larger memory requirements. The requirements for time, computational hardware, and memory of the different strategies for complex recognition tasks is a crucial issue that needs to addressed in future work.

Namely using a tree classifier we progressively formulate a hypothesis. At each level of the tree a new component is added to the hypothesis. The current node determines what the next question to be asked is. If a complete hypothesis is rejected we can backtrack and follow a new path, leading to a different complete hypothesis (the rejection is fed back to the previous layer). In more complex classification problems we could use preliminary tests that are more intuitively appealing and use a variety of classifiers in subsequent stages, each of them fine-tuned to make a classification based on the conclusions drawn up to that point.

6. CONCLUSSIONS

In this paper we describe that a reconfigurable processor can much more efficiently perform a number of applications, like pattern recognition, by allowing the device to reprogram its hardware resources.

An architecture for the OPGA module has been presented. The module can be made very compact because of the technique to multiplex the holograms in the memory. The characterization of the VCSEL array revealed that VCSELs are suitable for holographic recording and a good choice for the OPGA.

We have applied the OPGA to digit classification with a variety of strategies. This flexibility allows us to tradeoff between the required area, time and performance. Based on simulation results we further conclude that in order to perform classification using a limited amount of area in a reasonable time the tree search strategy should be preferred, provided that reconfiguration memory is not a scarce asset, as in the case for OPGAs.

ACKNOWLEDGEMENTS

The authors want to acknowledge Eric Fossum and Sandor Barna from Photobit Corp. for their contribution to the design and testing of the OPGA chip, and also Arrigo Benedetti for helpful discussions on FPGA in computation. The research is funded by DARPA through Contract F30601-98-1-0199 and by the National Science Foundation Engineering Research Center grant to Caltech.

REFERENCES

1. S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, Norwell, 1992.

2. G. Barbastathis, M. Levene, and D. Psaltis, "Shift multiplexing with spherical reference waves", Applied Optics, Vol. 35 No. 14, pp. 2403-2417, 1996.

3. G. J. Steckman, I. Solomatine, G. Zhou, and D. Psaltis, "Characterization of phenanthrenequinone-doped poly(methyl methacrylate) for holographic memory", Optics Letters, Vol. 23 No. 16, pp. 1310-1312, 1998.

4. M. Motomura, Y. Aimoto, A. Shibayama, Y. Yabe, and M. Yamashina, "An Embedded DRAM-FPGA Chip with Instantaneous Logic Reconfiguration", *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, K. L. Pocek, J. M. Arnold, pp. 264-266, IEEE Computer Society, Los Alamitos, 1998.