FPGAs tailored for low temperatures
Xilinx Virtex-5 and Virtex-7: In some physics experiments, such as particle detectors at CERN or in space systems, Xilinx (now AMD) Virtex-5 and Virtex-7 series FPGAs have been used at very low temperatures. These FPGAs have been subjected to temperature testing, and although they were not specifically designed to operate at cryogenic temperatures, they have demonstrated stability in cryogenic temperature ranges. Some additional configurations may be necessary to ensure their performance.

Intel (formerly Altera) Stratix IV: This series of FPGAs has also been used in low-temperature physics applications. As with Xilinx FPGAs, they are not specifically designed for cryogenic operation, but some versions have been tested in low-temperature conditions in certain experimental projects.

Adaptations and Limitations
Component Revision and Modification: Although these FPGAs can operate in cold environments, they require careful adaptation of their supporting components, such as capacitors and resistors, which must also be able to operate in these temperatures without deterioration.

Performance and Reliability Issues: Low temperatures can affect the charge retention characteristics and operation of transistors. Hardware developers often work closely with manufacturers to fine-tune parameters and ensure that circuits operate correctly.

Vivado Programming: There are no specific tools in Vivado to configure FPGAs at cryogenic temperatures, so design and programming in VHDL or Verilog is typically performed. However, the Vivado development environment allows you to optimize the circuit design to improve the FPGAs tolerance to different temperature conditions.

Alternatives for Low-Temperature Testing
Another option is to use FPGAs in low-temperature test systems, which will allow you to assess whether the hardware can withstand cryogenic conditions. This is typically done in advanced physics experiments and requires specific cooling equipment.

The Xilinx Artix-7 is not specifically designed to operate in cryogenic conditions, and its low-temperature performance is not guaranteed. The Artix-7 family is optimized for applications requiring power efficiency and good performance over a temperature range typical for industrial and commercial environments, but is not rated for operation in extremely low-temperature environments, such as those in cryogenic applications.

Limitations of the Artix-7 for low-temperature use
Temperature range: Artix-7s are typically rated to operate in commercial (-40°C to 100°C) or industrial (-40°C to 125°C) temperature ranges, which are wide but not cryogenic levels (below -150°C, as would be required in many particle physics experiments or space environments).

Electronic component behavior: At very low temperatures, internal FPGA components such as transistors and capacitors can experience adverse effects such as changes in capacitance and charge carrier mobility. This impacts the reliability and speed of circuits within the FPGA.

Cryogenic performance: In some controlled experiments, certain FPGA models (such as Virtex, Stratix, etc.) have shown some performance in low-temperature conditions, but the Artix-7 has not been widely tested in such environments. It is not recommended for applications that demand reliable performance in cryogenic conditions without extensive testing and adaptation.

Alternatives for low-temperature experiments
For projects that require hardware in cryogenic environments, more robust FPGAs are typically used, such as the Xilinx Virtex-5 and Virtex-7 series or even Kintex UltraScale, which have shown some resilience in lower-temperature testing, albeit with limitations. The use of additional specialized components to withstand the cryogenic environment and ensure system stability is also considered.

There are no ready-made code examples for direct dark matter detection, as such experiments and designs are extremely specialized and experimental.

Simulation of a Cryogenic Quantum System in Qiskit: In this case, we can simulate a quantum system that represents a theoretical detection of low-energy particles under controlled conditions. This does not simulate an FPGA itself, but rather a basic quantum system in Qiskit that can be used to explore low-energy interactions, similar to what you would look for with a cryogenic detector.

VHDL Code Example to Implement a Processing System in FPGA: Although you cannot do a "dark matter detection" in VHDL (as it requires very specific hardware and advanced detection infrastructure), I can help you create a basic template in VHDL that allows you to simulate a signal input that could be processed in an experimental setting.

1. Simulation of a Quantum System in Qiskit
This Qiskit code creates a simple quantum circuit that allows you to simulate interactions of low-energy particles through superposition states. This is not a cryogenic simulation per se, but you could adapt and expand the code to tune parameters that model a low-energy environment.

```
from qiskit import QuantumCircuit, Aer, transpile, assemble,
execute
from qiskit.visualization import plot_histogram
from qiskit.providers.aer import AerSimulator

# Crear un circuito cuántico de 2 qubits para simular un sistema de
baja energía
qc = QuantumCircuit(2)

# Colocar los qubits en superposición para simular fluctuaciones de
baja energía
qc.h(0)  # Puerta Hadamard en el qubit 0
qc.cx(0, 1)  # CNOT para entrelazar qubits
```

```
# Medir los qubits
qc.measure_all()

# Configuración del simulador y ejecución
simulator = AerSimulator()
compiled_circuit = transpile(qc, simulator)
job = simulator.run(compiled_circuit)
result = job.result()

# Visualizar los resultados
counts = result.get_counts(qc)
print("Resultados de la simulación de baja energía:", counts)
plot_histogram(counts)
```

Superposition and entanglement: By applying a Hadamard gate and then a CNOT gate, we put the qubits into a superposition and entanglement state. This simulates a low-energy configuration.

Simulation and Measurement: Measurement allows you to see how quantum states can fluctuate in a theoretical low-energy system, similar to experiments that would try to detect low-interacting particles. This code does not detect dark matter, but it does allow you to simulate low-energy phenomena that could help you model some detection properties.

2. VHDL Code for Signal Processing on an FPGA
Here is some basic VHDL code that simulates the capture and processing of a digital signal. In an experimental application, you could adjust this to simulate receiving data from a sensor on an FPGA system.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Signal_Processor is
    Port ( clk       : in STD_LOGIC;
           reset     : in STD_LOGIC;
           signal_in : in STD_LOGIC;
           signal_out: out STD_LOGIC);
end Signal_Processor;

architecture Behavioral of Signal_Processor is
    signal temp_signal : STD_LOGIC := '0';
begin
    process(clk, reset)
    begin
```

```
        if reset = '1' then
            temp_signal <= '0';
            signal_out <= '0';
        elsif rising_edge(clk) then
            temp_signal <= signal_in;
            signal_out <= temp_signal;
        end if;
    end process;
end Behavioral;
```

Code explanation:
Signal input and output: The code receives a signal (signal_in) to be processed and then sends it to the output (signal_out).

Timing: Processing occurs on each rising edge of the clock (clk), allowing for a basic simulation of signal processing on an FPGA.

Reset: If reset is high, the system is initialized.

Adaptation for Dark Matter Detection Simulation:

This code is very basic, but you could adapt it for an experimental environment where a simulated signal represents the presence of particles or interactions. If you are using FPGA simulators, such as Vivado's, you could create an input signal that simulates low-energy events and analyze how the processor responds in VHDL.

Handling Input and Output Signals: At low temperatures, circuits may experience fluctuations in voltage levels and switching times. Therefore, it is important to introduce delay times or buffers to stabilize the reading of the signals.

Input Filters and Debounce: Systems in low temperature conditions may present noise due to variations in the physical properties of the components. Incorporating basic filters or debouncing at the input can help reduce erroneous readings.

Calibration of Sensors and Signals: As the temperature drops, analog and digital input values may fluctuate, so it is ideal to have calibration mechanisms to compensate for possible variations.

VHDL Code Example with Adjustments for Low Temperature Simulations
Below is an example of how to adapt the VHDL code to take into account some of these strategies.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Low_Temperature_Signal_Processor is
    Port ( clk       : in STD_LOGIC;
           reset     : in STD_LOGIC;
           signal_in : in STD_LOGIC;
           signal_out: out STD_LOGIC;
           stable_signal: out STD_LOGIC);  -- Señal estabilizada
para salida
end Low_Temperature_Signal_Processor;

architecture Behavioral of Low_Temperature_Signal_Processor is
    signal temp_signal : STD_LOGIC := '0';
    signal debounce_counter : INTEGER := 0;
    constant debounce_limit : INTEGER := 100000;  -- Límite para el
debounce
begin
    -- Proceso principal con reducción de frecuencia de reloj
    process(clk, reset)
    begin
        if reset = '1' then
            temp_signal <= '0';
            signal_out <= '0';
            debounce_counter <= 0;
        elsif rising_edge(clk) then
            -- Debounce: Aumentar el contador solo si la señal es
estable
            if signal_in = temp_signal then
                if debounce_counter < debounce_limit then
                    debounce_counter <= debounce_counter + 1;
                else
                    stable_signal <= temp_signal;
                end if;
            else
                debounce_counter <= 0;
            end if;

            -- Procesamiento de la señal de salida con retardo
adicional
            temp_signal <= signal_in;
            signal_out <= stable_signal;  -- Salida estabilizada
        end if;
    end process;
end Behavioral;
```

Code Explanation:

Reducing the Clock Rate: Although the clock rate is not set here directly, this design allows you to reduce the speed at the hardware level or from the settings in Vivado or the simulation environment you use. This

should be done in the final synthesis project, by setting a low clock rate to improve stability at low temperatures.

Debounce Implementation:

The debounce_counter signal is used to check the stability of the input signal. Only if signal_in remains constant for a time (defined by debounce_limit), it is considered "stable".
This technique helps mitigate noise and rapid changes that can occur at low temperatures.
Signal Buffering and Stabilization:

The stable_signal signal is used to buffer the stabilized signal after the debounce. Then, signal_out takes the value of this stabilized signal, allowing for a more reliable output.
Considerations for Implementation in Vivado

Clock Configuration: In the design environment, reduce the clock frequency to the minimum that your system can support. This is crucial in cryogenic conditions to avoid timing and transient issues in components.

Simulation Testing: You can simulate component behavior under extreme conditions using simulation tools that allow you to tune temperature conditions. This will help you predict how design elements will behave.

Timing Analysis: Run timing analysis in Vivado to verify that setup and hold times are viable at low frequencies. This also allows you to identify critical paths that could be affected by the cryogenic environment.

Quantum Simulation in Qiskit as an Add-on
If your goal is to simulate phenomena at the quantum level, Qiskit will allow you to model particle interaction at low energies experimentally. You can create test circuits that model how fluctuations might be detected in a hypothetical quantum system, although this is not directly applicable to FPGA hardware.

Example 1: Simulating Low-Energy Particles with Qiskit
In this example, we will use Qiskit to simulate a two-particle system in an entangled quantum state that could serve as a model for detecting small energy variations, simulating subtle changes that could be interpreted as low-energy signals.

```
from qiskit import QuantumCircuit, Aer, execute
from qiskit.visualization import plot_histogram
from qiskit.quantum_info import partial_trace

# Crear un circuito cuántico con 2 qubits y 1 bit clásico
qc = QuantumCircuit(2, 2)
```

```
# Preparar un estado entrelazado
qc.h(0)  # Puerta Hadamard en el primer qubit
qc.cx(0, 1)  # Puerta CNOT para entrelazar los qubits

# Medir ambos qubits
qc.measure([0, 1], [0, 1])

# Ejecutar la simulación en el simulador Aer
backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1000)
result = job.result()

# Mostrar los resultados
counts = result.get_counts(qc)
print("Detección de partículas a baja energía:", counts)
plot_histogram(counts)
```

Explanation:

Entangled States: Quantum entanglement allows for observing correlations between particles, even in a low-energy system. These correlations can help detect variations in the system that could be indicative of rare events, such as the presence of dark matter.

Measurements: This model measures the states of particles, resulting in a distribution that can show hints of fluctuations at extremely low energies.

Example 2: Detecting Noise Signals in Low-Energy Conditions with SciPy

In this example, we will create a model to analyze a signal that simulates a very low-energy frequency, using a filter to extract patterns that could be hidden by thermal noise in a low-temperature environment.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter, welch

# Configuración de parámetros de la señal
fs = 1000  # Frecuencia de muestreo
t = np.arange(0, 10, 1/fs)  # Vector de tiempo
low_energy_freq = 5  # Frecuencia de señal de baja energía (5 Hz)
signal = 0.5 * np.sin(2 * np.pi * low_energy_freq * t)

# Agregar ruido para simular condiciones de baja temperatura
noise = np.random.normal(0, 0.2, len(t))
noisy_signal = signal + noise

# Filtro de paso bajo para aislar la frecuencia de baja energía
def low_pass_filter(data, cutoff, fs, order=4):
```

```
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = lfilter(b, a, data)
    return y

# Aplicar filtro a la señal ruidosa
filtered_signal = low_pass_filter(noisy_signal, cutoff=10, fs=fs)

# Visualización de la señal y del filtro aplicado
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(t, noisy_signal, label="Señal ruidosa")
plt.plot(t, signal, label="Señal original", linestyle='--')
plt.title("Señal de baja energía con ruido térmico")
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(t, filtered_signal, label="Señal filtrada", color="red")
plt.title("Señal de baja energía después de aplicar el filtro")
plt.legend()
plt.show()
```

Explanation:
Low-Pass Filter: This filter reduces high-frequency noise that can occur due to thermal fluctuations in cryogenic environments, highlighting the low frequencies that represent the desired energy.

Spectral Analysis: To improve the detection of low-energy variations, you can use spectral analysis, such as the Welch method, to observe how different frequencies behave under low-temperature conditions.

Example 3: Detection of Quantum Energy Variations in Qiskit (Hamiltonian Model)
This example simulates a low-energy Hamiltonian, allowing changes in the quantum state to be observed due to a small perturbation in energy. This model is relevant if you are looking to detect interactions that minimally alter the system, such as those that dark matter could induce.

```
from qiskit import Aer, transpile
from qiskit.opflow import X, Z, I
from qiskit.opflow.primitive_ops import PauliSumOp
from qiskit.algorithms import NumPyEigensolver
import numpy as np

# Definir el Hamiltoniano de baja energía: H = a*Z + b*X
a, b = 0.1, 0.05  # Coeficientes de baja energía
hamiltonian = a * Z ^ I + b * X ^ I

# Resolver los valores propios para obtener los niveles de energía
```

```
eigensolver = NumPyEigensolver()
result = eigensolver.compute_eigenvalues(hamiltonian)

# Mostrar los niveles de energía calculados
print("Niveles de energía de baja energía (eigenvalues):")
print(result.eigenvalues.real)
```

Explanation:
Hamiltonian with Pauli Operators: Here, we use *Z*
Z and *X* operators in the Hamiltonian to model the small energy variations in the system.

Energy Eigenvalues: The obtained energy levels indicate the possible quantum states, which could be relevant if you are looking to detect changes in these levels as a consequence of particles that interact weakly with the system.

Example 4: Analysis of Temperature Fluctuations and Signal in SciPy
For a system operating in low-temperature environments, you can use SciPy to simulate how temperature fluctuations affect the signal. This example uses a white noise function with different intensity levels.

```
from scipy.signal import welch

# Generar señal de baja energía simulada
low_energy_freq = 1.5  # Frecuencia baja para simular energía
mínima
signal = 0.1 * np.sin(2 * np.pi * low_energy_freq * t)

# Añadir ruido blanco para simular fluctuaciones térmicas
thermal_noise = np.random.normal(0, 0.02, len(t))
thermal_signal = signal + thermal_noise

# Aplicar un análisis espectral para observar las variaciones de
frecuencia
f, Pxx = welch(thermal_signal, fs=fs, nperseg=1024)

# Visualizar
plt.figure(figsize=(10, 4))
plt.semilogy(f, Pxx, label="Espectro de Potencia con Ruido
Térmico")
plt.title("Análisis de Fluctuaciones de Temperatura en Señales de
Baja Energía")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Densidad de Potencia")
plt.legend()
plt.show()
```

Explanation:

White Noise for Simulating Temperature: This noise represents the random behavior of particles at different energies, ideal for emulating the low-temperature environment.
Power Analysis: By looking at the power spectrum, you can identify whether there are signals at specific frequencies that might match the energy of unknown particles.

Summary
These examples combine Qiskit for quantum simulation of low-energy states and SciPy for processing signals affected by thermal noise. Each can be adapted and used to detect small variations in energy or frequency, which could prove useful in experimental models for indirect detection of particles such as dark matter under low-temperature conditions.