```python
import datetime
import unittest

class DNSRecord:
    """
    Represents a DNS record with basic attributes.
    """
    def __init__(self, record_id, domain, record_type, value):
        self.record_id = record_id
        self.domain = domain
        self.record_type = record_type
        self.value = value

class DNSManager:
    """
    Manages DNS records, supporting CRUD operations.
    """
    def __init__(self):
        self.records = {}

    def create_record(self, record_id, domain, record_type, value):
        """
        Creates a new DNS record. Raises an error if the record_id already
exists.
        """
        if record_id in self.records:
            raise ValueError("Record ID already exists.")
        self.records[record_id] = DNSRecord(record_id, domain, record_type,
value)

    def read_record(self, record_id):
        """
        Reads a DNS record by its ID. Raises an error if the record is not found.
        """
        if record_id not in self.records:
            raise ValueError("Record not found.")
        return self.records[record_id]

    def update_record(self, record_id, domain=None, record_type=None,
value=None):
        """
        Updates an existing DNS record with new values.
        """
        if record_id not in self.records:
            raise ValueError("Record not found.")
        record = self.records[record_id]
        if domain:
            record.domain = domain
        if record_type:
            record.record_type = record_type
        if value:
            record.value = value

    def delete_record(self, record_id):
        """
        Deletes a DNS record by its ID. Raises an error if the record is not
found.
        """
        if record_id not in self.records:
            raise ValueError("Record not found.")
        del self.records[record_id]

class Monitor:
    """
    Monitors and logs updates to DNS records.
    """
    def __init__(self):
        self.log = []

    def log_update(self, record_id, update_details):
        """
        Records a log entry for a DNS record update with a timestamp.
        """
        timestamp = datetime.datetime.now()
        log_entry = f"Timestamp: {timestamp}, Record ID: {record_id}, Update:
{update_details}"
        self.log.append(log_entry)
        return log_entry

class TestDNSManagement(unittest.TestCase):
    """
    Test cases for DNS management operations.
    """
    def test_create_and_read_record(self):
        manager = DNSManager()
        manager.create_record("001", "example.com", "A", "192.168.1.1")
        record = manager.read_record("001")
        self.assertEqual(record.domain, "example.com")
        self.assertEqual(record.value, "192.168.1.1")
```

```python
    def test_update_record(self):
        manager = DNSManager()
        manager.create_record("002", "example.org", "A", "192.168.2.1")
        manager.update_record("002", value="192.168.2.2")
        record = manager.read_record("002")
        self.assertEqual(record.value, "192.168.2.2")

    def test_delete_record(self):
        manager = DNSManager()
        manager.create_record("003", "example.net", "A", "192.168.3.1")
        manager.delete_record("003")
        with self.assertRaises(ValueError):
            manager.read_record("003")

if __name__ == "__main__":
    unittest.main()
```